UNIVERSIDAD AUSTRAL | INGENIERÍA

Product Quality Evaluation Method (PQEM): Medición de la Calidad de un Producto de Software

Tesis presentada para obtener el título de Doctora en Ingeniería de la Universidad Austral

Ing. Mariana Falco

Directora: Dra. Gabriela Robiolo

Laboratorio de Innovación, Desarrollo y Transferencia de la Universidad Austral (LIDTUA)

CONICET

2022

Pilar, Buenos Aires, Argentina

Resumen

La creciente complejidad de los sistemas de software significa que existen varios desafíos dentro del diseño, desarrollo y logro de la calidad del software. Los propietarios de productos, líderes de proyectos o profesionales deben comprender el nivel de calidad del producto, de una manera sintética e intuitiva para facilitar la decisión de aceptar o rechazar la iteración. La presente tesis introduce una novedosa solución denominada Método de Evaluación de la Calidad del Producto (abreviado en inglés, PQEM) para evaluar las características de calidad de un producto de software, utilizando el enfoque Goal-Question-Metric, ISO/IEC 25010, y la extensión de la cobertura de prueba aplicada a cada característica de calidad. Asimismo, la extensión del PQEM a través de la suma de pesos asociados a cada característica de calidad para permitir que cada uno de los grupos de interés establezca su importancia de cada característica de calidad. El resultado de PQEM es un valor único que representa la calidad por cada iteración de un producto, como una medida agregada. Se llevó a cabo un conjunto de aplicaciones ilustrativas aplicando PQEM a tres iteraciones de una aplicación web y móvil, dentro del entorno sanitario. La investigación sobre el campo de la operacionalización de la calidad en algunos indicadores numéricos es útil para los profesionales. Como tal, PQEM permite a los stakeholders obtener un valor agregado multidimensional que representa el nivel de calidad obtenido por iteración de un producto de software.

Palabras claves: medición de calidad, ciclo de vida de un producto software, evaluación de la calidad.

Product Quality Evaluation Method (PQEM): Measurement of the Quality of a Software Product

Abstract

The increasing complexity of software systems means that there are several challenges within the design, development, and achievement of software quality. Product owners, project leaders or practitioners need to comprehend the product quality level, in a synthetic and intuitive way to facilitate the decision of accepting or rejecting the iteration. This thesis presents a novel solution called Product Quality Evaluation Method (PQEM) to evaluate the quality characteristics of a software product, using the Goal-Question-Metric approach, ISO/IEC 25010, and the extension made of test coverage applied to each quality characteristic. Likewise, the extension of PQEM through the addition of weights associated with each quality characteristic to allow each stakeholders to set their importance of each quality characteristic. The outcome of PQEM is a single value representing the quality per each iteration of a product, as an aggregate measure. A set of illustrative applications was carried out by applying PQEM to three iterations of a web and mobile application, within the healthcare environment. Research on the field of operationalizing the quality in some numeric indicators is useful for practitioners. As such, PQEM allows the stakeholders to obtain a multidimensional aggregated value that represents the obtained quality level per iteration of a software product.

Keywords: quality measurement, software product life cycle, quality evaluation.

Agradecimientos

A mamá, que siempre estuvo.

A Harry y Betún. A Maquefá.

A mi directora, Gabriela Robiolo, por acompañarme, enseñarme y guiarme durante todo este tiempo; por escuchar mis propuestas y el potenciar el trabajo colaborando y creciendo.

A la Universidad Austral, y en particular, a la Facultad de Ingeniería por abrirme las puertas para, no solo realizar mi Doctorado, si no también para potenciar mi formación como profesional y persona.

Al CONICET, como sustento para la promoción de la ciencia y la tecnología en nuestro país, y en creer en mi plan de tesis para haber obtenido la beca doctoral cofinanciada para llevarla adelante.

A mis colegas, compañeros, y docentes, que siempre apoyaron y motivaron el progreso de la tesis y mi crecimiento como persona.

Cuando la gratitud es absoluta, las palabras sobran

Índice General

Índice de Figuras y Tablas

1. Introducción

1.1. Motivación

Los distintos avances tecnológicos han impactado la sociedad en gran medida, y en particular, el software ha cobrado una gran importancia y popularidad en la vida cotidiana, transformando actividades regulares como la gestión de pagos de servicios por aplicaciones, la lectura de códigos QRs en los restaurantes, entre otros. La calidad del software es un factor vital relacionado con el éxito empresarial y la seguridad humana, que describe el grado de conformidad con los requisitos y expectativas explícitos o implícitos; lo que motiva el interés en dicha área en un contexto de dominios variados y dinámicos (Fenton & Bieman, 2014).

Existen dos enfoques que se pueden seguir para garantizar la calidad del software. Uno se centra en una especificación y evaluación directa de la calidad del producto de software, mientras que el otro es asegurar la calidad del proceso cuando el producto está subdesarrollado (Zhao et al., 2017). Particularmente, se considera el primer enfoque para garantizar la calidad del software, y la evaluación de la calidad tiene como objetivo el examen sistemático del grado en que una entidad es capaz de cumplir con los requisitos especificados.

La calidad de un sistema es extremadamente importante y las métricas del software se convirtieron en una parte esencial para comprender si la calidad del software corresponde a aquello que los *stakeholders* necesitan (Mordal et al., 2018).

Estas necesidades son incluidas dentro de la norma ISO/IEC 25010:2011, como un conjunto de características y subcaracterísticas de calidad. Considerando los diferentes stakeholders que participan en los proyectos de software como desarrolladores, gerentes, usuarios, entre otros; la calidad debe evaluarse con diferentes niveles de detalle.

Con base en lo anterior, se han propuesto varias métricas diferentes, pero la aplicación práctica de las mismas se ve desafiada, por un lado, por la necesidad de combinar diferentes métricas según lo recomendado por diferentes métodos de modelos de calidad como el enfoque *Goal-Question-Metric* (Basili et al., 1994) y *Factor-Criteria-Metric* (McCall et al., 1977)

y, por otro lado, por la necesidad de conocer la calidad de todo el producto de software, con base en los valores métricos obtenidos para elementos de software como métodos y clases.

En consecuencia, una evaluación de calidad significativa debe combinar los resultados de varios métodos para responder preguntas específicas, combinando, por ejemplo, la complejidad ciclomática con la cobertura de la prueba (Mordal et al., 2018). Como tal, los *project managers* y los profesionales de calidad tienen diferentes complicaciones cuando necesitan comprender el nivel de calidad del producto, de una manera sintética e intuitiva para permitir la identificación del estado en cada iteración del producto. Por ejemplo, cuando el *project manager* debe tomar la decisión de aceptar o rechazar el desarrollo realizado en una iteración, evaluar el trabajo de los desarrolladores, decidir un pago o negociar una extensión de presupuesto.

En este contexto, se hace necesario entender y comprender en profundidad no sólo qué es lo que se desarrolla para cumplir con los requerimientos funcionales de un producto, si no también cómo se lleva a cabo ese desarrollo, con qué nivel de calidad se logra cada requerimiento. Un entendimiento integral de cada uno de ellos es lo que verdaderamente permite vislumbrar el producto resultante. De la misma manera, y si bien, existen diversos métodos que buscan medir la calidad del producto, no todos obtienen una medida per sé sino más bien una aproximación de esa calidad resultante. Pero es con una medida con la que es viable tomar una decisión para continuar con el desarrollo y con otra iteración.

1.2. Objetivos

1.2.1 Objetivo general

La presente tesis posee el siguiente objetivo general:

- Construir un método que permita lograr un análisis de calidad de un producto de software, a partir del estudio y medición de un conjunto de características de calidad en cada una de las iteraciones de dicho producto, lo que permitirá obtener un único número multidimensional representativo de la calidad.

1.2.2 Objetivos específicos

Los objetivos específicos asociados se listan a continuación:

- Definir un método que:
    - permita obtener el nivel de calidad de un producto software,
    - posibilite el análisis y la medición de un conjunto de características de calidad, teniendo como base el enfoque Goal-Question-Metric,
    - extienda el concepto de cobertura para lograr calcular el nivel de calidad por característica de calidad individual y en conjunto,
    - pueda ser aplicado en cada una de las iteraciones dentro del ciclo de vida del software, pudiendo evidenciar la evolución del nivel de calidad del producto a lo largo su ciclo de vida.
- Diseñar y ejecutar casos ilustrativos con stakeholders reales que permitan analizar la aplicabilidad del método y extraer hallazgos y conclusiones.
- Definir el uso de pesos como complemento en la especificación del método, para esquematizar una jerarquía entre características de calidad.
- Construir un catálogo de métricas aplicadas en la industria.
- Desarrollar una aplicación software que permita aplicar el método PQEM.

1.3. Marco de referencia

El método conecta la evaluación de la calidad del software y los requisitos no funcionales, dos áreas de investigación que tienen una larga trayectoria contribuyendo al desarrollo de cada una. Además, el método se basa en logros sólidos y conocidos.

En primer lugar, el enfoque ***Goal-Question-Metric*** (Basili, 1992; Caldiera et al., 1994) que ha sido validado empíricamente en muchos estudios de caso y demostrado su valor en estudios sobre requisitos.

También, en la **Teoría Representacional de la Medición** (Fenton & Bieman, 2014) la cual sostiene que la medición, en el sentido más amplio, consiste en la asignación de números a objetos o fenómenos de acuerdo con reglas. Dicha teoría busca formalizar la intuición sobre la forma en que funciona el mundo; es decir, los datos que se obtienen como medidas deben representar atributos de las entidades que se observan, y la manipulación de los datos deben preservar las relaciones que observamos entre las entidades.

Otro concepto importante abordado dentro del método PQEM construido es la **cobertura** (Garai y Adamkó, 2017), la cual permite asociar un valor con un conjunto de pruebas para un programa en particular, y es este valor el que indica la completitud de esas pruebas del programa. Fenton y Bieman (2014) también abordan la cobertura de las pruebas como un indicador del proceso de calidad de las pruebas.

Partiendo de este concepto de cobertura, y extendiéndolo a las características de calidad, el método PQEM buscó lograr las definiciones y fórmulas necesarias para entender el nivel alcanzado por el producto para las características de calidad y la ISO/IEC 25010:2011, a la par de definir los criterios de aceptación para cada característica de calidad.

El análisis de la cobertura de calidad de un producto define una medida clara de la calidad del producto, logrando también una evaluación confiable de la calidad general del software, como lo proponen Horgan et al. (1994) con respecto al testing.

Los **estándares** también son importantes para PQEM, porque su principal objetivo es que brindan una base de comprensión para las personas y las organizaciones, y representan una herramienta que facilita la comunicación y la medición. En este contexto, aseguran que los diferentes productos, componentes y servicios de diferentes personas, empresas y organizaciones sean compatibles; a la par de promover la difusión del conocimiento en las industrias. De esta manera, puede verse como una forma de cooperación voluntaria entre la industria, los usuarios, los gobiernos y los investigadores en busca de consenso. Consecuentemente, la familia de estándares ISO/IEC 25000 son parte esencial dentro de la construcción del método PQEM.

1.4. Contribuciones

Las principales contribuciones de la tesis se presentan a continuación:

1. Reconociendo la existencia de otros métodos y metodologías que abordan el mismo objetivo, la presente tesis describe el diseño de un Método de Evaluación y Medición de Productos Software (en inglés, *Product Quality Evaluation Method* - **PQEM**) que incluye características de calidad definidas por el estándar ISO/IEC 25010:2011, a partir de un análisis completo y profundo de cada característica de calidad en la iteración.

    a. PQEM incluye una **medida agregada** que permite un análisis detallado de los resultados por característica de calidad, logrando además la síntesis de los requisitos funcionales y no funcionales en un número único multidimensional que representa el nivel de calidad de un producto software,

    b. Dicha medida se logró a través del empleo de la **cobertura como un concepto de escala porcentual**, que permite definir la cobertura de las características de calidad y la cobertura de los productos de software en cada iteración,

    c. PQEM emplea **GQM** y el estándar **ISO/IEC 25010:2011** como la base metodológica a partir de la cual se derivan los QARs, ampliando luego los criterios de aceptación para requisitos funcionales y no funcionales, lo que permite de cierta manera abordar las falencias que han sido sugeridas en la literatura, sobre los mismos

        i. Lo que implica que el practitioner puede definir los objetivos, los QARs, las métricas y los criterios de aceptación que el producto a medir debe cumplimentar

    d. PQEM resulta en una herramienta que posibilita la **toma de decisiones** de los profesionales y stakeholders, a partir del nivel de calidad

resultante; siendo capaz además de obtener la historia de calidad de cada producto.

e. Un aspecto no positivo de PQEM es que, en sí, el proceso de medición requiere **tiempo** y dedicación; pero ese esfuerzo dedicado permite obtener ese único número para una toma de decisión efectiva y significativa.

2. La idea subyacente de PQEM fue operacionalizada de la siguiente forma:

   a. En una **herramienta**, denominada Quality Tracker Tool (QTT) que permite ejecutar el método y visualizar los resultados en tendencias.

   b. Una **Tesis de Maestría** en la que se desarrolló Product Quality Dashboard (PQD), el cual es una aplicación open-source que brinda un panorama general del nivel de calidad en el release a través de su uso en el pipeline que activa la tool en el proceso integrado con SonarQube[1]

   c. Una **Tesina de Grado** en la que se extendió PQD para integrar Jenkins, considerando además métricas para medir la calidad del software relacionado con la integración y el deployment continuo, lo que posibilita al Product Manager analizar la calidad del producto[2]

3. Se ha efectuado, a la par, la realización de un **análisis de la literatura** para comprender las métricas aplicadas en la industria. De dicho análisis, surgió la construcción de un catálogo que fue incluido en QTT.

   a. Dicho **catálogo** posibilita que el practitioner pueda seleccionar métricas existentes, además de ser capaz de crear nuevas o específicas para el producto que se busca analizar.

4. Las **aplicaciones ilustrativas** fueron posibles a partir de la interacción con una asignatura de la carrera de Ingeniería en Informática la cual aborda el desarrollo de aplicaciones con stakeholders reales. En particular, se trabajó con stakeholders del área de medicina, quienes participaron activamente en la

[1] MSc Thesis, Kert Prink, Master in Computer Science, University of Tartu, Estonia
[2] Tesina de Grado, Franz Soto Leal, Ingeniería en Informática, Facultad de Ingeniería, Universidad Austral

definición del alcance y la medición, brindando su perspectiva en cuanto a la correlación y completitud con su necesidad en el ámbito médico.

a. Dichas aplicaciones, y su correlativa medición, permitieron abordar no solo tests que respondan al estado y el comportamiento de la misma, sino también conocer otros aspectos que hacen a la calidad que no están incluidos en una cobertura de testing.

b. Actualmente, el método es compatible con las metodologías ágiles. En particular, para las aplicaciones ilustrativas se consideró como medición una versión completa con todas las funcionalidades, que fueron logradas luego de varios sprints. Consecuentemente, PQEM fue aplicado considerando una iteración como un conjunto de sprints.

5. Como una forma de clarificar el alcance, es viable mencionar que PQEM no es un método de desarrollo. Tampoco es un modelo de calidad, porque justamente necesita de un modelo que lo sustente como los estándares de calidad empleados. Es un método de medición de la calidad que aborda iteraciones, y que es capaz de acompañar cualquier ciclo de desarrollo que contenga iteraciones, independientemente de la cantidad de fases que existan.

Publicaciones

Battolla, T. F., Fuentes, S., Illi, J. I., Nacht, J., Falco, M., Pezzuchi, G., & Robiolo, G. “Sistema dinámico y adaptativo para el control del tráfico de una intersección de calles: modelación y simulación de un sistema multi-agente”, Simposio Argentino de Inteligencia Artificial (ASAI), 47 Jornadas Argentinas de Informática (47 JAIIO), 2018, pp. 20 – 33, ISSN: 2451-7585.

J. Nacht, M. Falco, & G. Robiolo. “Modelado y Simulación de una Intersección de Calles en un Contexto Multi-Agente”,In: Revista elektron, Departamento de Electrónica, Facultad de Ingeniería, Universidad de Buenos Aires, Vol. 2, No. 2, pp. 83-94 (2018), http://elektron.fi.uba.ar/index.php/elektron/article/view/59/47.

Falco, M., Bauret, L., and Robiolo, G., "*HeartCare: an Agent Oriented Architecture Implemented with Actors*", In: XXV Congreso Argentino de Ciencias de la Computación (CACIC 2019). Universidad Nacional de Río Cuarto, 14 al 18 de Octubre de 2019. Río Cuarto, Córdoba, Argentina.

Falco, M., Robiolo, G. "A Systematic Literature Review in Multi-Agent Systems: Patterns and Trends", In: XLV Conferencia Latinoamericana de Informática, Centro Latinoamericano de Estudios de Informática (CLEI), Panamá. 30 Septiembre - 4 Octubre 2019, IEEE.

Falco, M., and Robiolo, G., "*A Unique Value that Synthesizes the Quality Level of a Product Architecture: outcome of a Quality Attributes Requirements Evaluation Method*", 3rd International Workshop Series on Managing Quality in Agile and Rapid Software Development Processes (QuASD 2019), Barcelona, Spain, Nov 27th, 2019, Springer.

Falco, M., Robiolo, G., "Tendencies in Multi-Agent Systems: A Systematic Literature Review", CLEI Electronic Journal (CLEIej), April 2020 Vol 23, Number 1, paper #1. https://doi.org/10.19153/cleiej.23.1.1

Falco, M., Robiolo, G., and Salaberri, M., "*Plataforma Life+: Un Entorno Flexible para la Rehabilitación de Pacientes Cardíacos*", Congreso Argentino de Informática y Salud (CAIS 2020) en las 49 Jornadas Argentinas de Informática (49 JAIIO), Facultad de Ingeniería - UBA, Ciudad Autónoma de Buenos Aires, AR. Octubre 2020.

Falco, M., Scott, E., and 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.

Falco, M., and Robiolo, G. *Building a Catalogue of ISO/IEC 25010 Quality Measures Applied in an Industrial Context*. In: Journal of Physics: Conference Series (Vol. 1828, No. 1, p. 012077). 2020 International Symposium on Automation, Information and Computing (ISAIC 2020) 2-4 December 2020, Beijing, China.

Falco, M., and Robiolo, G., *PQEM: Product Quality Evaluation Method*, 7th International Conference on Software Engineering (SOFT 2021), July 24-25, 2021, London, United Kingdom.

Falco,, M. *Product Quality Evaluation Method and QualityTracker Tool Validation: a Case Study Plan*. XXIV Ibero-american Conference on Software Engineering. Doctoral Symposium. August 3rd - September 3rd, 2021, San José, Costa Rica.

Falco, M. and Robiolo, G., *Product Quality Evaluation Method (PQEM): To Understand the Evolution of Quality Through the Iterations of a Software Product* (October 7, 2021). International Journal of Software Engineering and Applications (IJSEA) Vol. 12, No. 5, September 2021, Available at SSRN: https://ssrn.com/abstract=3937996

Falco, M., and Robiolo, G. *Trends and Findings in Measuring Software Quality Metrics in the Industry*. In 2022 IEEE Congreso Bienal de Argentina (ARGENCON). IEEE.

Falco, M. and Robiolo, G. PQEMw, *Applying Weighted Sums to Software Quality Measurement*. In 51 Jornadas Argentinas de Informática (51 JAIIO), Universidad Abierta Interamericana, Ciudad Autónoma de Buenos Aires, AR. Octubre 2022.

1.5. Organización de la tesis

La presente tesis se estructura de la siguiente manera:

- Luego del capítulo introductorio, el **Capítulo 2** introduce las bases de la calidad del software, describiendo los modelos y los requerimientos asociados, junto con los estándares de calidad y las características de calidad.
- El **Capítulo 3** presenta la Teoría Representacional de la Medición, el Enfoque Goal-Question-Metric y la noción de cobertura; como sustento teórico al método definido en la tesis.
- El **Capítulo 4** resume el estado del arte, en cuanto a literatura existente y el catálogo creado a partir de un estudio de la literatura en pos de obtener el conjunto de métricas que han sido aplicadas en la industria.

- El **Capítulo 5** describe la metodología aplicada en la presente tesis, como una base conceptual de lo realizado a partir del paradigma de la ciencia del diseño.

- El **Capítulo 6** introduce y caracteriza al método definido como parte del Doctorado en Ingeniería, denominado *Product Quality Evaluation Method* (PQEM). Dicho capítulo presenta su definición, la caracterización de los pasos, y las ecuaciones definidas para el cálculo de la cobertura individual por característica de calidad y el nivel de calidad por iteración. De la misma manera, se presenta la herramienta que se desarrolló para ser capaz de ejecutar el método.

- El **Capítulo 7** presenta la extensión del método PQEM para incluir pesos, a la hora de calcular la cobertura de las características de calidad, definiendo una idea de jerarquía entre el conjunto de características.

- El **Capítulo 8** describe los ejemplos ilustrativos del método PQEM con respecto a tres iteraciones, vislumbrando la aplicación del mismo en cada uno de sus pasos.

- El **Capítulo 9** puntualiza las amenazas a la validez del método PQEM y su aplicación.

- El **Capítulo 10** describe las conclusiones y las líneas de trabajo futuro.

- Finalmente, en **Anexos** se mencionan los documentos que contienen los análisis de las iteraciones con el método PQEM.

2. Introducción a la Calidad de Software

2.1 Calidad

La calidad debe definirse y medirse si es que efectivamente se busca lograr una mejora. Sin embargo, un problema en la ingeniería y la gestión de la calidad es que el término calidad tiende a ser ambiguo, por lo que comúnmente puede ser malinterpretado o resultar subjetivo. Una visión popular de la calidad es que es un rasgo intangible: se puede discutir, sentir y juzgar, pero no se puede pesar ni medir. Esta visión refleja el hecho de que las personas perciben e interpretan la calidad de diferentes maneras (Krasner, 2021).

Los conceptos erróneos y la vaguedad de las opiniones populares no ayudan al esfuerzo de mejora de la calidad en las industrias. Con ese fin, la calidad debe describirse a partir de una definición viable. Crosby (1979) define la calidad como conformidad con los requerimientos (del inglés, *conformance to requirements*) y Juran y Gryna (1970) la definen como aptitud para el uso (del inglés, *fitness for use)*.

La conformidad con los requerimientos implica que los mismos deben establecerse claramente de manera que no se de lugar a una mala interpretación. Luego, en el proceso de desarrollo y producción, se toman mediciones regularmente para determinar la conformidad con esos requerimientos. Las no conformidades se consideran defectos, y representan una ausencia de calidad (Fleming, 2016).

La definición de la aptitud para el uso o *fitness for use* considera los requerimientos y expectativas de los clientes, que implican si los productos o servicios se ajustan a sus usos. Dado que diversos clientes pueden usar los productos de diferentes maneras, significa que los productos deben poseer múltiples elementos de aptitud para su uso. Según Juran y Gryna (1970), cada uno de estos elementos es una característica de calidad y todos ellos pueden clasificarse en categorías conocidas como parámetros de aptitud para el uso.

Finalmente, puede decirse que las dos definiciones anteriores se encuentran relacionadas, son consistentes y similares, y la diferencia es que el concepto de idoneidad para el uso implica un papel más importante para los requisitos y expectativas de los clientes.

2.1.1 Calidad de Software

En software, el sentido más estrecho de la calidad del producto se reconoce comúnmente como la falta de errores o bugs en el producto. También, es el significado más básico de conformidad con los requerimientos, porque si el software contiene demasiados defectos funcionales, no se cumple el requerimiento básico de proporcionar la función deseada (Kan, 2003). Esta definición, usualmente, se expresa de dos maneras:

- **Tasa de defectos** (del inglés, *Defect Rate*) como por ejemplo: número de defectos por millón de líneas de código fuente, por punto de función u otra unidad.

- **Confiabilidad** (del inglés, *Reliability*) como por ejemplo: número de fallas por n horas de operación, tiempo medio de falla, o la probabilidad de funcionamiento sin fallos en un tiempo especificado.

Generalmente, la satisfacción del *stakeholder* se mide por el porcentaje satisfecho o insatisfecho a partir de un set de encuestas, donde además de la satisfacción general del cliente con el producto de software, también se mide la satisfacción con respecto a atributos específicos. Para aumentar la satisfacción general del *stakeholder*, así como la satisfacción con varios atributos de calidad, éstos últimos deben considerarse en la planificación y el diseño del software. Sin embargo, estos atributos de calidad no siempre son congruentes entre sí. Por ejemplo, cuanto mayor sea la complejidad funcional del software, más difícil será lograr la mantenibilidad (Kan, 2003).

2.1.2 Modelos de Ciclos de Vida de Software

Las métricas y los modelos de calidad no se pueden discutir en el vacío; deben estar referenciados al proceso de desarrollo de software. A lo largo del tiempo, han surgido diversos modelos de software que se describirán brevemente. El **proceso en cascada** (en inglés, *waterfall process*) ha sido probado en el tiempo y es más adecuado para el desarrollo de software de sistema complejo con numerosas interdependencias. Este proceso produce entregables intermedios claramente definidos y permite un fuerte control del proyecto.

El **enfoque de creación de prototipos** (en inglés, *prototyping approach*) permite al equipo de desarrollo y a los clientes aclarar los requisitos y su interpretación al principio del ciclo de desarrollo. No es un proceso per se; se puede utilizar con varios modelos de proceso. Se ha vuelto muy utilizado en el desarrollo de aplicaciones. También se puede utilizar con subsistemas de software de sistemas cuando intervienen interfaces externas.

El **proceso iterativo** y el **modelo en espiral** (en inglés, *spiral model*) se han utilizado ampliamente en los últimos años, especialmente en el desarrollo de aplicaciones. Junto con la gestión de riesgos y la creación de prototipos, estos nuevos procesos aumentan la probabilidad de que el producto final satisfaga los requisitos del usuario y faciliten la reducción del tiempo de ciclo.

El desarrollo del software ha sido impactado por las necesidades de una sociedad cambiante y dinámica, donde las exigencias y los requerimientos de las empresas y los consumidores han llevado a requerir ciclos de vida y metodologías diferentes a las que implementaban aquellos modelos originales de software. Agile, como metodología de desarrollo, surgió como una nueva forma de desarrollar software incorporando como valor a los individuos y las interacciones por sobre los procesos y las herramientas, a la colaboración del usuario por sobre la negociación de un contrato y al hecho de responder al cambio por sobre seguir un plan. Es un proceso adaptativo, orientado a las personas, a través una entrega temprana y continua de un software que aporte valor al stakeholder y al usuario final (Shore y Warden, 2021). Con respecto a PQEM (Falco et al., 2019), y en base a sus características, es posible aplicarlo a cualquier ciclo de vida de desarrollo que contenga iteraciones, porque los conceptos embebidos como la cobertura y la identificación de los QARs pueden ser efectivizados en cada iteración.

2.1.2 Requerimientos

Los requerimientos para un sistema vienen en una variedad de formas: requisitos textuales, maquetas, sistemas existentes, casos de uso, historias de usuarios y más (Bass et al., 2003; Bass et al., 2012). Independientemente de la fuente, todos los requisitos abarcan las siguientes categorías:

1. **Requerimientos funcionales**. Establecen lo que debe hacer el sistema y cómo debe comportarse o reaccionar a los estímulos en tiempo de ejecución.

2. **Requerimientos de atributos de calidad** (en inglés, *Quality Attribute Requirements*). Estos requisitos son calificaciones de los requisitos funcionales o del producto en general. Una calificación de un requisito funcional es un elemento como la rapidez con la que se debe realizar la función o la resistencia que debe tener a la entrada errónea. Una calificación del producto en general es un elemento como el tiempo para implementar el producto o una limitación en los costos operativos.

3. **Restricciones** (en inglés, *constraints*). Una restricción es una decisión de diseño con cero grados de libertad. Es decir, es una decisión de diseño que ya se tomó. Los ejemplos incluyen el requisito de usar un cierto lenguaje de programación o de reutilizar un cierto módulo existente, o una orden de administración para hacer que su sistema esté orientado al servicio. Podría decirse que estas opciones están dentro del ámbito del arquitecto, pero factores externos (como no poder capacitar al personal en un nuevo idioma, o tener un acuerdo comercial con un proveedor de software, o impulsar los objetivos comerciales de interoperabilidad de servicios) han llevado a esos en poder de dictar estos resultados de diseño.

Ahora bien, para cada uno de estos requerimientos, la respuesta de la arquitectura es la siguiente (Bass et al., 2012):

(a) los **requerimientos funcionales** se satisfacen asignando una secuencia apropiada de responsabilidades a lo largo del diseño,

(b) los **requerimientos de atributos de calidad** se satisfacen mediante las diversas estructuras diseñadas en la arquitectura y los comportamientos e interacciones de los elementos que pueblan esas estructuras y,

(c) las **restricciones** se satisfacen al aceptar la decisión de diseño y conciliar con otras decisiones de diseño afectadas.

En el caso de PQEM, se ha hecho foco en los requerimientos de atributos de calidad debido a que, en conjunto con el enfoque GQM, es viable definir ese conjunto de necesidades para cada atributo de calidad medido a partir de preguntas.

2.3 Estándares de Calidad

Los estándares de calidad de software son parte de la Ingeniería de Software, la utilización de estándares y metodologías para el diseño, programación, prueba y análisis del software desarrollado, con el objetivo de ofrecer una mayor confiabilidad y mantenibilidad en concordancia con los requerimientos exigidos. Es posible, entonces, elevar la productividad y el control en la calidad de software, a partir de la gestión de la calidad para mejorar su eficacia y eficiencia (Bass et al., 2003).

En un escenario en el que los sistemas de software se desarrollan y construyen, en su mayoría, por terceros, el contratante del servicio, debe confiar en el buen hacer del proveedor seleccionado, especialmente si no dispone de los medios apropiados para auditar la entrega y argumentar defectos en el proceso de desarrollo, si es que existieran.

En general, una vez validado que el sistema responde a los principales requerimientos funcionales especificados, el usuario realizará las pruebas de aceptación, corrigiendo los errores encontrados y deployando en el entorno de producción, la versión final. Sin embargo, no siempre se validan de manera rigurosa los requerimientos tanto funcionales como no funcionales, o se ejecutan validaciones que aseguren que el sistema es lo suficientemente robusto y estable como para pasar a un entorno productivo.

Un atributo de calidad (del inglés, *quality attribute*) es una propiedad medible o comprobable de un sistema que se utiliza para indicar qué tan bien el sistema satisface las necesidades de los stakeholders. Se puede considerar a un atributo de calidad como una medida de la bondad de un producto en alguna dimensión de interés para un stakeholder (Bass et al., 2003; Bass et al., 2012).

2.3.1 Norma ISO/IEC 25000

ISO 25000:2011[3] (SQuaRE, del inglés: *Software Quality Requirements and Evaluation*) es una nueva serie de normas que se basa en ISO 9126 y en ISO 14598 (Evaluación del software). Uno de los principales objetivos de la serie SQuaRE es la coordinación y

[3] ISO/IEC 25000, https://iso25000.com/index.php/en/iso-25000-standards

armonización del contenido de la ISO 9126 y de la ISO 15939:2002 (*Measurement Information Model*). SQuaRE está formada por las siguientes normas:

- **ISO/IEC 2500n**. División de gestión de calidad. Los estándares que forman esta división definen todos los modelos comunes, términos y referencias a los que se alude en las demás divisiones de SQuaRE.
- **ISO/IEC 2501n**. División del modelo de calidad. El estándar que conforma esta división presenta un modelo de calidad detallado, incluyendo características para la calidad interna, externa y en uso.
- **ISO/IEC 2502n**. División de mediciones de calidad. Los estándares pertenecientes a esta división incluyen un modelo de referencia de calidad del producto software, definiciones matemáticas de las métricas de calidad y una guía práctica para su aplicación.
- **ISO/IEC 2503n**. División de requisitos de calidad. Los estándares que forman parte de esta división ayudan a especificar los requerimientos de calidad. Estos requisitos pueden ser usados en el proceso de especificación de requerimientos de calidad para un producto software que va a ser desarrollado o como entrada para un proceso de evaluación. El proceso de definición de requerimientos se guía por el establecido en la norma.
- **ISO/IEC 15288 (ISO, 2003). ISO/IEC 2504n**. División de evaluación de la calidad. Estos estándares proporcionan requerimientos, recomendaciones y guías para la evaluación de un producto software, tanto si la llevan a cabo evaluadores, como clientes o desarrolladores.
- **ISO/IEC 25050–25099**. Estándares de extensión SQuaRE. Incluyen requisitos para la calidad de productos de software *Off-The-Self* y para el formato común de la industria (CIF) para informes de usabilidad.

ISO/IEC 25010:2011

ISO/IEC 25010:2011[4] describe el modelo de calidad como la piedra angular de un sistema de evaluación de la calidad de un producto, y esta norma determina qué características de calidad se tendrán en cuenta al evaluar un producto de software.

La calidad de un sistema es el grado en que el sistema satisface las necesidades declaradas e implícitas de sus diversos stakeholders y, por lo tanto, proporciona valor. Las necesidades de esos *stakeholders* (funcionalidad, rendimiento, seguridad, mantenibilidad, entre otras) son precisamente lo que se representa en el modelo de calidad, que categoriza la calidad del producto en características y subcaracterísticas.

Figura 1. Características y subcaracterísticas de calidad del estándar ISO/IEC 25010:2011. Fuente: Elaboración propia.

El modelo de calidad del producto (ver Figura 1) comprende las siguientes características de calidad:

[4] ISO/IEC 25010:2011, https://iso25000.com/index.php/en/iso-25000-standards/iso-25010

A. **Idoneidad Funcional** (en inglés, *Functional Suitability*)**:** representa el grado en que un producto o sistema proporciona funciones que satisfacen las necesidades declaradas e implícitas cuando se utiliza en condiciones específicas. Esta característica se compone de tres subcaracterísticas: *Functional completeness*, *Functional correctness*, *y Functional appropriateness***.**

B. **Eficiencia de Rendimiento** (en inglés, *Performance Efficiency*): concierne el desempeño en relación con la cantidad de recursos utilizados en las condiciones establecidas. Esta característica se compone de las siguientes subcaracterísticas: *Time behaviour*, *Resource utilization*, *y Capacity*.

C. **Compatibilidad** (en inglés, *Compatibility*): es el grado en el que un producto, sistema o componente puede intercambiar información con otros productos, sistemas o componentes y / o realizar sus funciones requeridas mientras comparte el mismo entorno de hardware o software. Esta característica se compone de las siguientes subcaracterísticas: *Co-existence, y Interoperability*.

D. **Usabilidad** (en inglés, *Usability*): es el grado en el que usuarios específicos pueden utilizar un producto o sistema para lograr objetivos específicos con eficacia, eficiencia y satisfacción en un contexto de uso específico. Esta característica se compone de las siguientes subcaracterísticas: *Appropriateness recognizability*, *Learnability*, *Operability*, *User error protection*, *User interface aesthetics*, *y Accessibility*.

E. **Confiabilidad** (en inglés, *Reliability*): es el grado en el que un sistema, producto o componente realiza funciones específicas en condiciones específicas durante un período de tiempo específico. Esta característica se compone de las siguientes subcaracterísticas: *Maturity*, *Availability*, *Fault tolerance*, *y Recoverability*.

F. **Seguridad** (en inglés, *Security*): es el grado en el que un producto o sistema protege la información y los datos para que las personas u otros productos o sistemas tengan el grado de acceso a los datos adecuado a sus tipos y niveles de autorización. Esta característica se compone de las siguientes subcaracterísticas: *Confidentiality*, *Integrity*, *Non-repudiation*, *Accountability*, *y Authenticity*.

G. **Mantenibilidad** (en inglés, *Maintainability*): esta característica representa el grado de eficacia y eficiencia con el que se puede modificar un producto o sistema para mejorarlo, corregirlo o adaptarlo a cambios en el entorno y en los requisitos. Esta característica se compone de las siguientes subcaracterísticas: *Modularity*, *Reusability*, *Analysability*, *Modifiability*, *y Testability*.

H. **Portabilidad** (en inglés, *Portability*): es el grado de efectividad y eficiencia con el que un sistema, producto o componente se puede transferir de un hardware, software u otro entorno operativo o de uso a otro. Esta característica se compone de las siguientes subcaracterísticas: *Adaptability*, *Installability*, *y Replaceability*.

Las características y subcaracterísticas que se mencionan en la Figura 1 son incluidas en el método PQEM como una base, a partir de la cual, es posible derivar los Quality Attribute Requirements (QARs) y ejecutar luego las coberturas por cada característica de calidad.

A modo de resumen, el capítulo abordó la descripción del concepto de la calidad de software, puntualizando la importancia de llevar a cabo un proceso de medición fehaciente durante el ciclo de vida de un producto. También, se comentaron la evolución de los modelos de software y los requerimientos de calidad. De la misma manera, se describieron los estándares de calidad como un marco de referencia implementado dentro del método PQEM.

3. Introducción a la Medición de la Calidad de Software

3.1 Teoría Representacional de la Medición

La Teoría Representacional de la Medición busca formalizar las ideas intuitivas sobre el funcionamiento del mundo, es decir: aquellos datos que se obtienen como mediciones deben representar atributos de las entidades que se observan, mientras que, la manipulación de los datos debe preservar las relaciones observadas entre las entidades (Fenton & Bieman, 2014).

Es viable partir de relaciones sencillas con una comprensión inicial de un evento o tópico (como por ejemplo, la medición de la temperatura) y, luego, avanzar a mediciones sofisticadas, precisas y con herramientas particulares. Formalmente, es viable definir la **medición** (del inglés, *measurement*) como el mapeo del mundo empírico al mundo formal y relacional. En consecuencia, una **medida** (del inglés, *measure*) es el número o símbolo asignado a una entidad por este mapeo para caracterizar un atributo.

3.1.1 Las Reglas del Mapeo

Es posible emplear una medida para caracterizar un atributo. En el mundo real, se estudia una entidad, buscando ampliar el conocimiento sobre ella. Por ende, el dominio del mapeo es el mundo real mientras que el rango es el mundo matemático. Cuando se asigna el atributo a un sistema matemático, se tiene una gran cantidad de opciones para el mapeo y el rango, debido a que pueden utilizarse números reales, enteros o incluso un conjunto de símbolos no numéricos.

Para medir la altura de una persona, no basta con especificar un número. Si se mide la altura en pulgadas, entonces se está definiendo un mapeo del conjunto de personas en pulgadas; si se mide la altura en centímetros, entonces se tiene un mapa diferente. Además, incluso cuando el dominio y el rango son los mismos, la definición de mapeo puede ser diferente. Es decir, puede haber muchas asignaciones diferentes y diversas formas de medir,

según las convenciones que se adopten. Consecuentemente, una medida debe especificar el dominio y el rango, así como la regla para realizar el mapeo (Fenton & Bieman, 2014).

3.1.2 La Condición de Representación de la Medición

Por definición, cada relación en el sistema relacional empírico corresponde a través de la medición a un elemento en un sistema numérico. Se busca que las propiedades de las medidas en el sistema numérico sean las mismas que las de los elementos correspondientes en el mundo real, de modo que al estudiar los números se pueda aprender sobre el mundo real; buscando que el mapeo preserve la relación, regla denominada como condición de representación (Fenton & Bieman, 2014).

La **condición de representación** afirma que un mapeo de medición M debe mapear entidades en números y relaciones empíricas en relaciones numéricas, de tal manera que las relaciones empíricas se preserven y sean preservadas por las relaciones numéricas. Por ejemplo, la relación empírica “más alto que” es mapeada a una relación numérica “>”, en particular: `A es más alto que B si y sólo si M (A) > M (B).`

El mapeo considerado como medida (del inglés, *measure*) también conocido como representación u homomorfismo, porque la medida representa el atributo en el mundo numérico. En este contexto, los pasos del proceso de medición formal son los siguientes:

1. Identificar el atributo para entidades del mundo real
2. Identificar relaciones empíricas para el atributo
3. Identificar relaciones numéricas correspondientes a cada relación empírica
4. Definir el mapeo de las entidades del mundo real a los números
5. Verificar que las relaciones numéricas se preservan y son preservadas por las relaciones empíricas (condición de representación)

En general, pueden existir diversas formas de asignar números que satisfagan la condición de representación. La naturaleza de las diferentes asignaciones determina el tipo de escala del atributo. Existen cinco tipos de escala bien conocidos: nominal, ordinal, de intervalo,

de razón y absoluta. El tipo de escala para una medida determina qué tipo de declaraciones es factible realizar de manera significativa utilizando la medida.

De lo anterior, pueden extraerse algunas conclusiones. Se utiliza la noción de representación para definir la validez: cualquier medida que satisfaga la condición de representación es una medida válida. Luego, cuanto más rico es el sistema de relaciones empíricas, menor cantidad de medidas válidas. Se considera que un sistema relacional es rico si tiene un gran número de relaciones que se pueden definir. Pero, a medida que aumenta el número de relaciones empíricas, se incrementa el número de condiciones que un mapeo de medición debe satisfacer en la condición de representación (Fenton & Bieman, 2014).

3.1.3 Literatura existente

La aplicación de la teoría en la medición de las propiedades relacionadas con la calidad del software significa que tenemos una herramienta poderosa para comprender lo que medimos, cuáles son las operaciones significativas en los valores medidos y cómo interpretar los resultados (Jørgensen, 1999). La literatura denota la aplicación de dicha teoría en enfoques axiomáticos para la medición del software, junto con una aplicación de ejemplo: la definición de una medida de acoplamiento de agregación para clases de objetos (Poels y Dedene, 2000)..

También, en un framework para el ciclo de vida de medición de software con un enfoque particular en la fase de diseño de una medida de software, para así derivar criterios de verificación relacionados con dichos diseños de métodos de medición. El uso previsto de dicho framework es proporcionar un marco de análisis estructurado para comprender y evaluar los métodos de medición existentes documentados en la literatura y en uso en la industria (Habra et al, 2008).

Por otro lado, Benedicenti et al. (2001) buscaron medir la calidad de los agentes de software escritos en Java usando programación extrema, donde seleccionaron un factoring de calidad de acuerdo con las propiedades y limitaciones del tipo de productos y método de desarrollo considerado; y lograron las medidas cuantitativas a través de la Teoría Representacional de la Medición.

3.2 El Enfoque Goal Question Metric

3.2.1 La medición

El desarrollo de productos software requiere de un mecanismo que posibilite la medición, en pos de la retroalimentación y la evaluación, que permita responder a una variedad de preguntas relacionadas con la implementación de cualquier proceso de software (Basili et al., 1994). La medición da sustento a la planificación del proyecto, permite determinar fortalezas y debilidades, proporciona una justificación tanto sea para adaptar como para perfeccionar técnicas, y permite evaluar la calidad de los procesos y los productos específicos.

En el ciclo de vida del proyecto, a través de la evaluación, es posible evaluar el progreso, aplicar medidas correctivas basadas en dicha evaluación y comprender el impacto de dicha acción. Para que sea eficaz y como plantean Basili, Caldera y Rombach (1994), la medición debe ser:

1. Centrada en objetivos específicos;
2. Aplicada a todos los productos, procesos y recursos del ciclo de vida;
3. Interpretada en base a la caracterización y comprensión del contexto organizacional, ambiente y metas.

Lo anterior significa que la medición debe definirse de arriba hacia abajo, debe estar enfocada, basado en metas y modelos.

3.2.2 El enfoque

El enfoque *Goal Question Metric* (GQM) se basa en la suposición de que, para que una organización logre una medición bien orientada, debe primero especificar los objetivos propios y los de sus proyectos, para luego integrar esos objetivos con los datos que están orientados a definir dicho objetivos operacionalmente y, finalmente, proporcionar un marco para interpretar los datos. GQM se definió originalmente para evaluar defectos en un conjunto de proyectos

dentro del ámbito de *NASA Goddard Space Flight Center*[5], y si bien, el enfoque se empleó para definir y evaluar objetivos de un proyecto en un dominio particular, su uso se ha expandido a un contexto más amplio; como un paso de establecimiento de objetivos en un paradigma evolutivo de mejora de la calidad (Basili et al., 1994).

3.2.2.1 Niveles del Modelo de Medición

El resultado de la aplicación de GQM es, en sí, la especificación de un sistema de medición dirigido a un conjunto particular de problemas y un conjunto de reglas para la interpretación de los datos de medición. El modelo de medición resultante tiene tres niveles: conceptual, operacional y cuantitativo (Basili et al., 1994).

Nivel conceptual (Objetivo, Meta o *Goal*)

Un objetivo se define para un objeto, con distintas razones, con respecto a varios modelos de calidad, desde varios puntos de vista, en relación con un entorno particular. Los objetos de medición son:

- **Productos**: artefactos, entregables y documentos que se producen durante el ciclo de vida del sistema; como por ejemplo, especificaciones, diseños, programas, conjuntos de pruebas, entre otros.
- **Procesos**: actividades relacionadas con el software normalmente asociadas con el tiempo; como por ejemplo, especificar, diseñar, probar, entrevistar, entre otros.
- **Recursos**: elementos utilizados por los procesos para producir sus productos; como por ejemplo, personal, hardware, software, espacio de oficina, entre otros.

Nivel operacional (Pregunta o *Question*)

Se utiliza un conjunto de preguntas para caracterizar la forma en que se realizará la evaluación o el logro de una meta específica en base a algún modelo. Las preguntas intentan caracterizar el objeto de medición (que puede ser un producto, un proceso, un recurso) con

[5] **NASA Goddard Space Flight Center**: nombrado por el pionero de los cohetes Dr. Robert H. Goddard, la NASA estableció el centro como su primer complejo de vuelos espaciales en 1959. Estudia la Tierra, el sol, el sistema solar y el universo.

respecto a un tema de calidad seleccionado y determinar su calidad desde el punto de vista seleccionado.

Nivel cuantitativo (Métrica o *metric*)

A cada pregunta se le asocia un conjunto de datos para poder responderla de forma cuantitativa. Los datos pueden ser objetivos o subjetivos, como se describen a continuación:

- **Objetivo**: dependen únicamente del objeto que se está midiendo y no del punto de vista desde el que se toman. Por ejemplo: número de versiones de un documento, horas del personal dedicadas a una tarea, tamaño de un programa, entre otros.
- **Subjetivos**: dependen tanto del objeto que se está midiendo como del punto de vista desde el que se toman. Por ejemplo: legibilidad de un texto, nivel de satisfacción del usuario, entre otros.

3.2.2.2 El Modelo como Estructura Jeráquica

El modelo GQM es una estructura jerárquica que comienza con una meta u objetivo; que especifica el propósito de la medición, el objeto que se va a medir, el problema que se va a medir y el punto de vista desde el que se toma la medida. Dicho objetivo se refina en varias preguntas, como la del ejemplo, que generalmente desglosan el problema en sus componentes principales. Luego, cada pregunta se refina en métricas, sean en base a datos objetivos o subjetivos. Se puede utilizar la misma métrica para responder diferentes preguntas bajo el mismo objetivo (Basili et al., 1994).

3.2.2.3 Un Ejemplo de Aplicación

Para dar un ejemplo de aplicación del enfoque GQM, se puede suponer el caso en que se quiere mejorar la puntualidad del procesamiento de solicitudes de cambio, durante la fase de mantenimiento del ciclo de vida de un sistema. El objetivo resultante especificará un

propósito (mejorar), un proceso (procesamiento de solicitud de cambio), un punto de vista (project manager) y un problema de calidad (puntualidad).

Este objetivo se puede refinar a una serie de preguntas, por ejemplo, sobre el tiempo de respuesta y los recursos utilizados. Estas preguntas pueden responderse mediante métricas que comparen tiempos de respuesta específicos con los promedios. El modelo completo de GQM se muestra a continuación.

Goal	Propósito	mejorar
	Problema	la puntualidad de
	Objeto (proceso)	procesamiento de solicitud de cambio
	Punto de vista	del project manager
Question		¿Cuál es la velocidad de procesamiento de la solicitud de cambio actual?
Metrics		Tiempo de ciclo medio (en inglés, *Average Cycle Time*)
		Desviación estándar (en inglés, *Standard Deviation*)
		Porcentaje de casos fuera del límite superior
Question		¿Está mejorando el rendimiento del proceso?
Metrics		$\frac{current\ average\ cycle\ time}{baseline\ average\ cycle\ time} * 100$
		Calificación subjetiva de la satisfacción del gerente

En resumen, el enfoque GQM es un mecanismo que permite definir e interpretar el software operativo de una forma comprensible pero a la vez medible. Puede utilizarse de forma aislada o, mejor, dentro del contexto de un enfoque más general para la mejora de la calidad del software.

En este último caso, el desarrollo de modelos GQM es una tarea realizada por la diversidad de experiencias que utilizará como insumos al proceso los objetivos impulsados por el negocio proporcionados por la gestión corporativa y las características del entorno proporcionadas por el equipo del proyecto. Vale destacar que la sección 4 de Estudios Relacionados contiene ejemplos de aplicación de GQM.

El proceso de elicitación en el método PQEM fue realizado a través del enfoque GQM, por lo que el **Paso 2.1** de dicho método (seleccionar características y subcaracterísticas de calidad) abordará el nivel conceptual con la definición de metas (considerando la estructura definida en (Basili et al., 1994), compuesta por: propósito, asunto, objeto y punto de vista); El **Paso 2.2** (especificar los QARs) incluirá el nivel operativo con la especificación de las preguntas por objetivo y, finalmente, el **Paso 2.3** (definir métricas y criterios de aceptación para cada QAR) especificará el nivel cuantitativo, definiendo las métricas por pregunta.

3.3 Cobertura

En ciertas industrias donde el software es un componente clave del producto, la satisfacción del cliente puede estar correlacionada con la calidad del software. En industrias como la fabricación de automóviles o aviones, los defectos del software pueden producir consecuencias financieras negativas e incluso poner en peligro la vida humana. Por ende, la mejora en la calidad del software y la recopilación de evidencia cuantitativa de la mejora, es fundamental en dichas industrias (Yang et al, 2009).

El testing de software es una práctica que se utiliza a menudo para indicar la calidad del software, pero es un proceso que requiere un nivel intermedio/alto de mano de obra y recursos, y comprender dicho requerimiento implica un compromiso entre presupuesto, tiempo y calidad. En un desarrollo de software sistemático, las empresas buscan medidas de integridad y bondad de las pruebas para establecer los criterios de finalización de las mismas, y la cobertura (del inglés, *coverage*) es una de ellas.

El propósito del testing de software es detectar errores (Myers et al., 2011), por lo que un set de testeos exhaustivos requieren de las funcionalidades implementadas y de la

cobertura; y por ello es posible asegurar que un programa cumple con sus especificaciones al ejercitar las características descritas en la especificación (Horgan et al., 1994).

Los testeos basados en cobertura (del inglés, *coverage-based testing)* miden el porcentaje de software que se analiza durante el proceso de testing (Yang et al, 2009). Es aplicable a cualquier etapa de testing, incluidos los testeos unitarios, los testeos de integración o los testeos del sistema. Los testeos de cobertura identifican construcciones en la codificación del programa que no se han ejercido durante los testeos funcionales. Los mismos guían el test de construcciones de software importantes y proporcionan una lista de verificación clara de la completitud para cada test (Horgan et al., 1994). Existen varias medidas de cobertura de pruebas unitarias que se utilizan para cuantificar la suficiencia de los testeos, como la cobertura de declaraciones, decisiones y funciones (Myers et al., 2011).

Debido a que la cobertura basada en especificaciones funcionales se basa en la disponibilidad de especificaciones, la cobertura basada en la estructura se emplea más comúnmente. Dichos testeos pueden medir la cobertura en varias granularidades, incluidas declaraciones, líneas, bloques, condiciones, métodos y clases. Proporciona una forma de cuantificar el grado de minuciosidad de los testeos de caja blanca. Los testeos de cobertura tienen las siguientes ventajas y desventajas (Yang et al., 2009).

1. La fiabilidad parece aumentar con la cobertura de las pruebas.
2. La cobertura del código proporciona una cuantificación del progreso del testeo.
    a. Seleccionar testeos que proporcionen la mayor ganancia incremental en cobertura es una forma de priorizar el testing.
    b. Las herramientas de cobertura permiten con frecuencia la detección de casos de prueba redundantes, que son candidatos para ser eliminados de un conjunto de pruebas porque ejecutarlos requiere tiempo y recursos sin mejorar de manera efectiva la detección de defectos.
3. Según observaciones en la industria (Yang et al., 2009), aumentar la cobertura del código se convierte en una motivación para mejorar las pruebas. La cobertura proporciona una medida cuantitativa que se puede utilizar para informar el progreso de las pruebas. Los desarrolladores u organizaciones de

aseguramiento de la calidad están entonces más motivados a mejorar el proceso de prueba utilizando la guía de la cobertura de prueba.

4. Intuitivamente, la mayoría de los desarrolladores sienten que aumentar la cobertura aumenta la detección de defectos. A pocos desarrolladores les gusta entregar código con, digamos, un 20% de cobertura. Desafortunadamente, no es posible cuantificar cuántos defectos más se pueden encontrar al aumentar la cobertura.

    a. La cobertura total (100%) no garantiza la ausencia de defectos. Siempre hay un equilibrio entre usabilidad y minuciosidad al elegir una medida.

        i. La cobertura de declaraciones es simple pero insensible a las condiciones.

        ii. Tampoco se conoce todavía un buen mapeo entre el nivel de cobertura y el esfuerzo de prueba.

    b. Una alta cobertura requiere un esfuerzo considerable y la relación entre la cobertura obtenida y el esfuerzo de prueba parece no lineal.

Esta idea de cobertura de testing fue extendida al método PQEM para poder calcular los valores de cobertura para cada una de las características de calidad, a partir de un cociente entre los QARs exitosos y el número total de QARs para cada característica; y el valor de cobertura total de la iteración que representa el nivel de calidad alcanzado.

A modo de resumen, el Capítulo describió los conceptos básicos de la medición de la calidad de software, para lo cual se describieron los puntos más importantes de la Teoría Representacional de la Medición como la base formal para las reglas de mapeo o representación, considerando atributos y relaciones. Luego, se introdujo el enfoque GQM el cual permite implementar un modelo de medición para el cual se definen objetivos, preguntas y métricas. Finalmente, se esbozó la idea de cobertura en pos de establecer la base que sustenta la extensión de dicho concepto para ser aplicado dentro del método y así poder calcular el valor de calidad para la iteración. El capítulo correspondiente de Estudios Relevantes contiene ejemplos relevantes en la literatura de los tópicos caracterizados.

4. Estudios Relacionados

PQEM es un método que permite medir el nivel de calidad alcanzado en cada iteración dentro del ciclo de vida de un producto software, y en función de ello, el presente capítulo ahonda en describir los estudios relacionados para el método en sí y los tópicos relacionados al mismo: la arquitectura de un producto, el modelo y las características de calidad, la utilización de pesos como ponderación de dichas características y indicadores agregados y, además, se describirán los resultados obtenidos de la realización de un mapeo sistemático que permitió identificar el conjunto de métricas de calidad que han sido aplicadas en la industria y publicadas.

4.1 Arquitectura

Tanto la definición de la arquitectura de un producto como la especificación de características de calidad y QARs son decisiones que no deben tomarse a la ligera porque tienen un alto impacto en el estado del producto final. Aunque las técnicas de evaluación arquitectónica basadas en escenarios (Kazman et al., 2000) son un enfoque bien establecido para realizar evaluaciones estructuradas de diseños arquitectónicos, estas técnicas no se utilizan ampliamente en la industria.

Un análisis de la literatura permitió identificar aquellos estudios relevantes en base a su importancia y similitud con los objetivos principales de PQEM. E. Woods (2012) creó un método de revisión arquitectónica llamado *Tiny Architectural Review Approach* (TARA) el cual se enfoca en qué tan bien una arquitectura particular admite un conjunto de requisitos clave, al contrario de lo que hacen la mayoría de los métodos basados en escenarios como ATAM (Kazman et al., 2000).

TARA permite la situación en la que el sistema ya se ha implementado, pero PQEM se puede aplicar mientras el software está en desarrollo. PQEM requiere cinco pasos por iteración, mientras que TARA se define con siete pasos. Considerando los siete pasos, una de las principales diferencias con PQEM es que no incluye métricas para analizar las características de calidad, pero en el Paso 3, analizan las métricas de producción del sistema.

Para tener éxito, TARA necesita un patrocinador (responsable del sistema) y un asesor (con conocimiento organizacional y de dominio). PQEM incluye al stakeholder en la definición

de las características de calidad, la definición del nivel de calidad esperado por iteración y por producto, y la comunicación de los resultados por iteración y por producto; así como el equipo de desarrollo. Tanto PQEM como TARA quieren comprender el contexto en el que existe el sistema y los requisitos funcionales clave que debe cumplir, pero PQEM lo hace a través de un proceso de obtención de requisitos.

TARA no define un nivel de calidad que deba cumplirse o compararse. TARA y PQEM comparten los conceptos de estructuras y dependencias de módulos, caracterización de código (acoplamiento, por ejemplo). TARA aborda solo la cobertura de prueba después de ejecutar todas las pruebas automatizadas disponibles, mientras que PQEM amplía este concepto para analizar la cobertura de todas las características de calidad por iteración de un producto, definiendo varias ecuaciones para calcular estos valores. A diferencia de TARA, PQEM llega a obtener hallazgos y conclusiones por iteración a través del nivel de calidad TOC_i, que es un número entre 0 y 1; y este número puede mostrar qué tan cerca estuvo la implementación de los criterios de aceptación definidos.

Posteriormente, algunos autores coincidieron en que gestionar la evolución rentable de los sistemas de software industrial representa un desafío debido a su complejidad y larga vida útil. Como tal, Koziolek et al. (2012) aplicó varios enfoques de última generación para combinarlos en un método ligero holístico llamado MORPHOSIS, que facilita arquitecturas de software sostenibles. En consecuencia, su enfoque principal es la sustentabilidad, mientras que el principal objetivo de PQEM es lograr un nivel adecuado de calidad que repercutirá no solo en la sustentabilidad sino en el conjunto de características de calidad incluidas. Este método incluye tres fases: análisis de escenarios de evolución, cumplimiento de la arquitectura y seguimiento de métricas a nivel de arquitectura.

En la primera fase, los autores realizaron un análisis de escenarios de evolución de acuerdo con una versión extendida del método ALMA (Bengtsson et al., 2004), a partir del cual pudieron realizar una elicitación de escenarios combinados de arriba hacia abajo y de abajo hacia arriba. Esta es una diferencia con PQEM, porque además de no estar basado en escenarios, el proceso de elicitación no se basa en ALMA, sino que se implementa el enfoque GQM. La segunda fase permite tratar las dependencias entre capas de módulos y, finalmente, dentro de la tercera fase han encontrado varias métricas de código a nivel de arquitectura que miden diferentes aspectos de la sostenibilidad.

El conjunto de métricas mide la calidad de la modularización de un sistema de software no orientado a objetos, y los autores emplean la noción de API como base para las métricas (Sarkar et al., 2006). Han utilizado la notación de estructuración de objetivos para desglosar la mantenibilidad según ISO/IEC 25010:2011, consecuentemente, no se han centrado en la totalidad de las características de calidad definidas por e estándar, como lo hace PQEM en el proceso de obtención.

Varios autores mencionaron que es importante comprender las consecuencias de las decisiones sobre los diversos artefactos de la ingeniería de software, como código, casos de prueba, deployments, entre otros; al analizar el impacto de una solicitud de cambio. Como tal, Rostami et al. (2015) presentan una herramienta llamada Predicción de mantenibilidad arquitectónica de Karlsruhe (del inglés, *Karlsruhe Architectural Maintainability Prediction* o KAMP) para analizar la propagación de cambios causada por una solicitud de cambio en un sistema de software basado en el modelo de arquitectura. KAMP se ocupa de todas las fases del ciclo de vida del software. Permite a los miembros del proyecto evaluar el efecto de las solicitudes de cambio en las áreas de trabajo técnicas y organizacionales durante el ciclo de vida del software.

KAMP brinda a los arquitectos de software la posibilidad de modelar la arquitectura inicial con información de contexto anotada que involucra tareas técnicas y organizativas, denominada arquitectura base, y la arquitectura después de implementar la solicitud de cambio, denominada arquitectura de destino. Luego, la herramienta calcula las diferencias entre el modelo de arquitectura base y de destino y genera listas de tareas aplicando reglas de derivación e interpretación en el modelo de arquitectura. Además de la propagación estructural de los cambios en la arquitectura, KAMP calcula los impactos en todas las áreas de trabajo principales que involucran tareas como el desarrollo y la ejecución de pruebas, la configuración de la compilación, la implementación y sus artefactos correspondientes que deben procesarse durante el ciclo de vida del software.

En resumen, en PQEM, la elicitación se basa en el método GQM, para especificar las necesidades de las partes interesadas en forma de objetivos, preguntas, métricas y criterios de aceptación para cada pregunta. Ninguno de los estudios propone alguna forma de síntesis del análisis, por lo que proponemos la definición y cálculo de valores de cobertura para cada característica de calidad seleccionada, y para todo el producto, lo que conduce a la consecución de un número multidimensional como valor resumen. del nivel de calidad alcanzado como

resultado final. Finalmente, el enfoque de este método está orientado a la medición de características de calidad.

4.2 Modelo y características de calidad

A la hora de evaluar las distintas propiedades que cumple un software, es el modelo de calidad el que define y especifica cuáles son esas características de calidad que serán consideradas (Kim et al., 2014, Alenezi, 2016, Mishra y Otaiwi, 2020). En cuanto al modelo de calidad, Ortega et al. (2003) diseñaron un prototipo con enfoque sistémico que contiene características de calidad (Eficiencia, Confiabilidad, Funcionalidad, Mantenibilidad, Portabilidad y Usabilidad) para analizar un producto y que, al aplicarlo, es posible no solo estudiar las debilidades y fortalezas, sino también discernir su cumplimiento con los estándares.

Varios atributos como el tamaño, la complejidad y el acoplamiento permiten abordar las propiedades de un producto de software, y se han propuesto varias métricas para medir estos atributos. Morasca y Briand (1997) construyeron un framework axiomático para definir medidas coherentes para un atributo en diferentes niveles de medida.

Posteriormente, Habra et al. (2008) propuso un framework para el ciclo de vida de la medición de software, con un enfoque particular en la fase de diseño de una medida de software. Dicho framework incluye las definiciones de los criterios de verificación que se pueden utilizar para comprender las etapas del diseño de medición de software, y también integra las diferentes perspectivas de los enfoques de medición existentes.

Siavvas et al. (2021) introdujeron un modelo un modelo de evaluación de seguridad jerárquica (del inglés, *Hierarchical Security Assessment Model*), capaz de evaluar el nivel de seguridad interna de los productos de software en función de indicadores de bajo nivel, es decir, alertas de análisis estático y métricas de software relevantes para la seguridad. El modelo, siguiendo las pautas de ISO/IEC 25010 y basado en un conjunto de umbrales y pesos, agrega sistemáticamente estos indicadores de bajo nivel para producir una puntuación de seguridad de alto nivel que refleje el nivel de seguridad interna del software analizado.

A nivel organizacional, Staron et al. (2009) introdujeron un framework para desarrollar sistemas de medición personalizados, en particular, aplicado en una unidad de desarrollo de software dentro de Ericsson. Han demostrado que con la ayuda de las normas ISO/IEC, los sistemas de medición se pueden utilizar de manera efectiva en la industria del software y que el marco presentado mejora la forma de trabajar con métricas.

Considerando también entornos organizacionales híbridos donde se utilizan simultáneamente metodologías de desarrollo más antiguas (p. ej., waterfall) y más nuevas (p. ej., Agile y Continuous Integration Continuous Delivery [CI/CD]) en diferentes partes de la empresa, Pradhan y Nanniyur (2021) describen un framework general desarrollado e implementado en Cisco Systems para transformar el sistema de gestión de calidad heredado para organizaciones en cascada en un sistema moderno de Calidad 4.0. El marco consta de los siguientes seis componentes: estandarización de métricas, estandarización de procesos, medición, informes, análisis de calidad, y cultura y liderazgo,

La calidad del software en uso (del inglés, *quality in use* o QinU) es la percepción del software en su contexto de uso (Atoum y Bong, 2015). QinU ha sido aplicado en dominios como aplicaciones web y mobile, ingeniería de software y desarrollo de procesos comerciales. La medición de la calidad permite a los proveedores de software mejorar sus estrategias de gestión y actividades de apoyo a la toma de decisiones, correlacionando la satisfacción de los usuarios y los ingresos de los proveedores (Atom, 2020).

Actualmente, existen dos enfoques para medir el software QinU: los modelos de calidad ISO/IEC 25010:2011 y modelos personalizados. I. Atom (2020) propuso un framework para medir las revisiones de software que consumen QinU de manera competente. Dentro del componente de predicción, se mapean las oraciones de revisión de software a sus respectivas características QinU del modelo ISO/IEC 25010:2011 basado en una medida de similitud de texto.

4.3 Uso de sumas ponderadas e indicadores de agregación

4.3.1 Uso de sumas ponderadas

Las sumas ponderadas (del inglés, *weighted sums*) se utilizan para cuantificar una serie de atributos de software, lo que permite utilizar una suma ponderada para combinar varias

medidas de nivel inferior para construir una única medida de nivel superior que puede cuantificar diferentes aspectos de un atributo dado al mismo tiempo o incluso un solo aspecto de un atributo, basado en una serie de medidas para él (Morasca, 2010). La literatura refleja ejemplos de su uso, como por ejemplo Hovorushchenko (2017) quien desarrolló un método para evaluar los pesos de las medidas e indicadores de calidad del software, basado en el conjunto de características de calidad. El análisis de ISO/IEC 25010:2011 arroja la conclusión de que existen medidas, que afectan a más de una sub-característica y características de la calidad del software, es decir, existe la correlación de subcaracterísticas y características por medidas (las subcaracterísticas y características dependen de 203 medidas, pero solo en 138 medidas diferentes).

4.3.2 Uso de indicadores de agregación

El método PQEM incluye sumas ponderadas como una forma de medir cada característica de calidad que está compuesta por un subconjunto de requisitos de atributos de calidad (QARs), que conducen a obtener una única medida por característica de calidad. Además, se obtiene una medida de la calidad general como una suma agregada de las medidas obtenidas por cada característica de calidad.

Ivan et al., (2015) describieron que el desarrollo de aplicaciones de software involucra conceptos tales como nivel de calidad estimado (usado cuando la aplicación aún no está disponible), nivel de calidad planificado (obtenido al estudiar la calidad de aplicaciones de software de la misma clase disponibles en el mercado) y el nivel real de calidad (debe establecerse para garantizar que se alcancen los objetivos de la aplicación).

Construyeron un indicador para cada criterio de evaluación que elige productos de calidad. Del conjunto de indicadores asociados a las características de calidad, se extrae un indicador para cada característica de modo que los indicadores seleccionados cumplan simultáneamente los mismos requisitos en cuanto a las propiedades de los indicadores como la sensibilidad. Estos mismos autores han desarrollado dos indicadores agregados que incluyen todas las características de calidad del estándar, uno es la agregación usando el operador de suma y el otro es la agregación usando el operador de producto. Sus principales variables son las siguientes: el número de características, el nivel de calidad y el peso asociado a una característica.

Dos de los primeros modelos describieron la calidad utilizando un enfoque de descomposición, uno es de McCall et al. (1977) y el otro de Boehm et al. (1978). Estos modelos se enfocan en el producto final e identifican atributos clave de calidad desde la perspectiva del usuario, llamados factores de calidad, que normalmente son atributos externos de alto nivel como confiabilidad, usabilidad y facilidad de mantenimiento; y asumen que los factores de calidad todavía se encuentran en un nivel demasiado alto para poder medirse directamente, por lo que se descomponen aún más en subfactores de calidad (Fenton & Bieman, 2014).

En el modelo de McCall (McCall et al., 1977), la confiabilidad del factor se compone de los criterios (o sub-factores) consistencia, precisión, tolerancia a errores y simplicidad. A veces, los criterios de calidad son atributos internos, como estructura y modularidad, lo que refleja la creencia de los desarrolladores de que los atributos internos tienen un efecto sobre los atributos de calidad externos.

Consideraron un conjunto de condiciones, que es sí (1) o no (0) según se cumpla o no; y pueden calcular un factor de calidad a través de la suma de cada condición, dividida por la cantidad total de condiciones por factor de calidad. Y es posible utilizar diferentes ponderaciones, para que las ponderaciones reflejen importancia, costo, entre otros (Fenton & Bieman, 2014). Además, definen la calidad del software en términos de atributos externos del software, que, a su vez, se definen en términos de sub-atributos del software. Cada sub-atributo puede medirse mediante un conjunto de medidas, y conceptualmente, las sumas ponderadas se han utilizado para agregar las medidas asociadas con un sub-atributo para obtener un valor único para medir el sub-atributo.

A su vez, todas las medidas que cuantifican los sub-atributos relacionados con un atributo pueden agregarse con una suma ponderada en una medida de ese atributo. Finalmente, se puede obtener una medida para la calidad global como una suma ponderada de las medidas obtenidas para los atributos (Morasca, 2010). La literatura muestra enfoques que abordan la medición de diferentes aspectos dentro del ciclo de vida de un producto de software. Por ejemplo, se definió un conjunto de medidas para cuantificar el tamaño de un producto al comienzo del desarrollo en función del conjunto de requisitos funcionales, y se denominan métodos de medición del tamaño funcional (del inglés, *Functional Size Measurement* o FSM).

El análisis de puntos de función (del inglés, *Function Points Analysis* o FPA) es la primera propuesta para FSM, proporciona una medida del tamaño en puntos de función y requiere un análisis manual de documentos informales como los requisitos de software y el tamaño de una aplicación antes de que se desarrolle (Abran y Robillard, 1996; Lavazza et al, 2013). En este contexto, y considerando que el proceso de identificación y ponderación de Componentes Funcionales Básicos (BFC) - en FPA, los requisitos funcionales del usuario se modelan como un conjunto de BFC - es difícilmente automatizable, Lavazza et al. (2013) definen que la observación de que la medición de FPA se basa en la medida de BFC y la posterior ponderación y agregación de las medidas de BFC.

Llegaron a la conclusión de que los procesos de definición y medición de los FPA se pueden simplificar drásticamente si se tiene en cuenta un subconjunto de los BFC utilizados en la definición original de la medida, lo que permite ahorros sustanciales en el esfuerzo de medición, sin sacrificar la precisión de las estimaciones del esfuerzo de desarrollo de software. Algunos estudios han utilizado sumas ponderadas como formas de cuantificar los atributos de calidad, por lo que PQEM incluye esta idea de suma ponderada como una forma de combinar medidas que son parte de cada atributo de calidad analizado en una iteración.

4.4. Catálogo de métricas en la industria

La calidad del software describe los atributos deseables de los productos de software. En este contexto, se realizó una evaluación de la existencia y disponibilidad de medidas de calidad en términos de características de calidad que se han aplicado a la industria, con el fin de determinar la viabilidad de aplicar las medidas de calidad existentes en la industria en el contexto del desarrollo de software.

Se efectivizó una revisión bibliográfica (Falco & Robiolo, 2021) en bases de datos científicas con respecto al concepto de características de calidad. Se aplicó el guideline de Kitchenham et al., (2007) para realizar el análisis, la evaluación y la interpretación de estudios relevantes, para responder a las preguntas de investigación que se enumeran a continuación.

RQ1: ¿Existen medidas de calidad para evaluar la calidad del producto de software en términos de características de calidad que se hayan aplicado a la industria?,

RQ2: ¿Qué métodos se utilizan para definir las medidas de calidad?,

RQ3: ¿Existe alguna herramienta automatizada y/o medida de calidad de las aplicadas a la industria?,

RQ4: ¿Cuáles son los principales dominios de aplicación dentro de la industria? ¿Existe una relación entre los dominios y las características de calidad?; y

RQ5: ¿Qué criterios de aceptación se han utilizado para determinar las características de calidad?

Con respecto al proceso de búsqueda, se consideraron estudios pertenecientes a bases de datos científicas (Science Direct, ACM, Springer e IEEE). Además de estos, también se utilizaron varios libros, fuentes en línea y estudios secundarios para realizar el trabajo de revisión sistemática de manera integral y para contrastar los resultados obtenidos (ACM, Elsevier, Google Scholar, IEEE Xplore, ScienceDirect, Springer, Taylor and Francis, Wiley biblioteca en línea).

Para determinar la existencia en la literatura de requisitos para atributos de calidad y atributos de calidad en general, se consideraron consultas que incluían "requisitos de atributos de calidad", "atributos de calidad", "métrica de pregunta de meta", "calidad del software", "ISO 25010 ", "medidas de calidad", "métricas de calidad", "criterios de aceptación" y diferentes combinaciones para cada característica de calidad (Idoneidad funcional, Eficiencia de rendimiento, Compatibilidad, Usabilidad, Confiabilidad, Seguridad, Mantenibilidad y Portabilidad) dentro de la norma ISO/IEC 25010: 2011\footnote{ISO/IEC 25010:2011, https://iso25000.com/index.php/en/iso-25000-standards/iso-25010}.

Estas consultas dieron lugar a un grupo inicial de 27.989 artículos, que se filtró aún más en las siguientes etapas. Una vez que se eliminaron los estudios duplicados, filtramos y aplicamos los criterios de exclusión; y luego aplicamos los criterios de inclusión a los artículos restantes, quedando 789 artículos para la elegibilidad. Luego del assessment de calidad, fue posible reducir la totalidad de artículos de más de miles a 27, los cuales fueron analizados individualmente para construir el catálogo para observar el flujo de información. Si bien el

conjunto de artículos puede parecer compuesto por una cantidad baja de artículos, pensamos que esto puede suceder debido a la falta de caracterización de las aplicaciones prácticas dentro de la industria.

Lo anterior permitió construir un catálogo de medidas de calidad que ya han sido utilizadas o aplicadas en la industria, permitiendo entender cuáles son los métodos utilizados para definir esas medidas de calidad y los principales dominios de aplicación, en pos de identificar si existen herramienta automatizadas y qué criterios de aceptación han sido utilizados para determinar las características de calidad.

El catálogo como tal proporciona a los diferentes stakeholders y usuarios, una lista completa de medidas de calidad listas para aplicar, las cuales permiten medir la calidad en cada iteración dentro del ciclo de vida del software.

Vale mencionar que se han encontrado un breve conjunto de contribuciones, que se esperaba por la especificidad del tema, pero se considera que aún así es representativo porque está compuesto por un conjunto de contribuciones que describen medidas de calidad en términos de las siguientes características de calidad: Funcional Idoneidad, Eficiencia de Rendimiento, Compatibilidad, Usabilidad, Confiabilidad, Tolerancia a Fallas, Seguridad, Mantenibilidad y Portabilidad.

4.4.1 Evaluar la calidad en términos de características de calidad

Siguiendo la primera pregunta (RQ1): **¿existen medidas de calidad para evaluar la calidad del producto de software en términos de características de calidad que se hayan aplicado a la industria?**, la misma fue respondida por el conjunto completo de artículos, y con base en el análisis es posible resumir los resultados en dos grandes grupos: el primero, describe cómo los autores describieron las medidas de calidad en términos de las siguientes características de calidad: Fault Tolerance, Functional Suitability y Usabilidad sin una metodología explícita, mientras que el segundo describe la aplicación de GQM para definir métricas de características de calidad.

4.4.1.1 Existencia de medidas de calidad

De acuerdo a la existencia de medidas de calidad (del inglés, *quality measures*) en términos de características de calidad, se encontraron contribuciones para las siguientes características: Fault Tolerance (Oinas, 2000) (Debnath et al., 2019), Functional Suitability (Debnath et al., 2019), y Usability (Schramme y Macías, 2019).

Uno de los trabajos a destacar, es el de Schramme y Macías (2019) quienes han propuesto un framework para analizar y medir las medidas de calidad de usabilidad durante la fase de implementación, a lo largo del código fuente. Estas anotaciones son interpretadas por un procesador de anotaciones para obtener información valiosa y calcular automáticamente las medidas de calidad de usabilidad en el momento de la compilación. Su enfoque incluye medidas de calidad internas, es decir, aquellas relacionadas con los aspectos de implementación del producto de software; inspirado en las medidas de calidad de usabilidad internas incluidas en ISO/IEC 25023.

4.4.1.2 Medidas de calidad en términos de GQM

Como se discutió anteriormente, la mayoría de los artículos analizados han utilizado GQM como el enfoque para definir un conjunto de métricas en términos de las siguientes características de calidad: Safety (Michael et al., 2010), Performance (Tan et al., 2013), Usability (Tan et al., 2013; Schrettner et al., 2012), Reliability (Gencel et al., 2013), Maintainability (Gencel et al., 2013), Security (Philippou et al., 2020), Testability (Beer y Felderer, 2018), y Evolvability (Tsuda et al., 2018).

Michael et al. (2010) definieron un framework de métricas de validación que utiliza el análisis de hazards y los requisitos de software derivados para mitigar dichos hazards, como indicadores para medir la suficiencia de los requisitos de seguridad del software en las primeras etapas del proceso de desarrollo del software.

Con respecto a la usabilidad, dos autores diferentes midieron esta característica de calidad. Por un lado, Tan et al. (2013) introdujo un framework para respaldar la industria de los móviles, que incluye una taxonomía de atributos y sub-atributos. Al utilizar GQM, identificaron un conjunto de preguntas y medidas para cada atributo. También, combinaron las medidas de calidad con el rendimiento.

Por otro lado, Schrettner et al. (2012) presentaron un enfoque para modelar, recopilar, almacenar y evaluar medidas de software. Su enfoque se basa en GQM y permite la definición de varias cuestiones que involucran diferentes tipos de medidas de calidad.

Gencel et al. (2013) propuso un marco de apoyo a la toma de decisiones para la selección de métricas (DSFMS) que se basa en el enfoque GQM. El núcleo del framework incluye un proceso iterativo de selección de métricas basado en objetivos que incorpora mecanismos de toma de decisiones en la selección de métricas, un repositorio de atributos / métricas predefinido y un modelo de trazabilidad entre los elementos de GQM.

Además de Reliability, otro objetivo de Gencel et al. (2013) fue mejorar la Mantenibilidad, que son sus respectivos sub-objetivos (disminuir los errores que causan fallas en la operación, mejorar el mantenimiento del proyecto y mejorar la mantenibilidad de los requisitos).

Philippou et al. (2020) describen que las métricas de seguridad sufren la falta de adaptabilidad a los contextos organizacionales particulares. Estos autores propusieron una metodología llamada SYMBIOSIS (*SecuritY Metrics and BusIness ObjectiveS, Integrated and Synchronised*) que define un proceso de obtención y refinamiento de metas que mapea los objetivos de negocio con las metas de medición de seguridad mediante el uso de plantillas sistemáticas que capturan elementos de contexto relevantes (objetivos de negocio, propósito, stakeholders, alcance del sistema).

Beer y Felderer (2018) informan sobre un estudio de caso en el que miden y mejoran la Testabilidad sobre la base del enfoque GQM.

Los defectos de capacidad de evolución son estados no comprensibles y no modificables que no producen directamente fallas de comportamiento en tiempo de ejecución. Un defecto de capacidad de evolución es: "*un defecto en el código que hace que el código sea menos compatible con los estándares, más propenso a errores o más difícil de modificar, ampliar o comprender*", como se define en (Mäntylä y Lassenius, 2006).

Tsuda et al. (2018) introdujeron un método para gestionar los defectos de Evolvability a través de un algoritmo de aprendizaje de árbol de clasificación que puede manejar relaciones no ortogonales entre métricas. Llevaron a cabo un estudio de caso utilizando sistemas

integrados desarrollados por la empresa Komatsu (Japón), que es un fabricante de máquinas en general.

4.4.2 Métodos para definir medidas de calidad

La pregunta RQ2: **¿qué métodos se utilizan para definir las medidas de calidad?** permitió comprender los métodos más utilizados para definir adecuadamente un conjunto de medidas de calidad. Una serie de modelos, herramientas y prácticas se han utilizado y reportado en la literatura, como GQM (Basili et al, 1994), GQM+ (Basili et al, 2014), Goal Argument Metrics (GAM) (Cyra et al, 2007), entre otros; y extensiones o mejoras para la formulación original de GQM. Como resultado del análisis, se identificaron métodos basados puramente en estándares como ISO/IEC, GQM, extensiones de GQM, y un estudio que, además de GQM, emplea *Goal Structuring Notation* (GSN). GQM resultó como el mecanismo más utilizado para definir e interpretar software operacional y medible (Zhao et al., 2015), con 15 artículos de los 27 analizados.

A partir de lo anterior es viable comprender que GQM todavía se usa para obtener un conjunto de métricas para medir la calidad del software, es interesante ver un ejemplo de cómo esas contribuciones operativas han incluido GQM, para mostrar la viabilidad e inspirar investigadores a continuar e innovar en ese punto. De la misma manera, algunos autores introdujeron una extensión o enfoque complementario para GQM, para obtener las métricas. Por ejemplo, para reducir la subjetividad de la aplicación de GQM para desarrollar un programa de métricas de calidad, Pradhan y Nanniyur (2021) han desarrollado un marco de Prevención-Identificación-Evaluación-Remoción (PIER) para la medición de la calidad.

4.4.3 Existencia de herramientas automatizadas

Para la pregunta RQ3: **¿existe alguna herramienta automatizada y/o medida de calidad de las aplicadas a la industria?**, se puntualiza, a continuación, el resumen de los enfoques de automatización encontrados.

- **Usability:** un procesador de anotaciones que se encarga de procesar automáticamente las anotaciones en el código y calcular las métricas en el momento de la compilación (Schramme y Macías, 2019).

- **Maintainability**: construyó una herramienta que proporciona medidas automáticas para las medidas relacionadas con la mantenibilidad del estándar IT-CISQ (Plöesch et al., 2015).

- **Testability**: utilizó la herramienta de análisis de calidad del software VizzAnalyzer (Wingkvist et al., 2010).

- **Evolvability**: aprendizaje automático con algoritmo C5.0 (Tsuda et al., 2018).

- **General**: operacionalización del modelo de calidad con respecto a las evaluaciones de calidad semiautomáticas para obtener declaraciones de calidad interpretables en niveles de abstracción más altos, y adaptó el kit de herramientas ConQAT para hacer frente a los requisitos que surgieron de su modelo de evaluación (Mayr et al., 2012); desarrollo de la aplicación *Unified Quality Monitoring* (UQM) (Schrettner et al., 2012); y desarrollo de un algoritmo k-means mejorado, y uso de FineReport para generar informes de métricas (Zhao et al., 2015).

4.4.4 Dominios de aplicación en la industria

Al responder la pregunta RQ4, **¿cuáles son los principales dominios de aplicación dentro de la industria? ¿Existe una relación entre los dominios y las características de calidad?**, se extrajeron el siguiente conjunto de dominios de aplicación donde se llevó a cabo la medición de la calidad: seguro social, tecnología aplicada, software y móvil, ingeniería, comunicaciones, financiero y sanitario. Aunque no es una lista extensa de dominios, estos son dinámicos y en continuo movimiento, lo que permite entender que si estos dominios mostraban aplicaciones industriales de evaluación de la calidad, entonces la necesidad de automatización es necesaria.

La mayor cantidad de características de calidad aplicadas se encontraban en los dominios de tecnología aplicada, software, dispositivos móviles y de ingeniería, que es de acuerdo a la era actual donde esos son dominios verticales que pueden funcionar y hacer una

integración con otro. Además, Security estudió las finanzas y la salud, un tema importante dentro de estos dominios.

4.4.5 Criterios de aceptación empleados

La pregunta RQ5: **¿qué criterios de aceptación se han utilizado para determinar las características de calidad?** permitió identificar si las contribuciones habían o no utilizado o definido criterios de aceptación para las medidas de calidad. En este contexto, un criterio de aceptación es un conjunto de condiciones o reglas comerciales aceptadas que la funcionalidad o característica debe satisfacer y cumplir para ser aceptada por el propietario del producto o las partes interesadas (Software Testing Help, 2020).

En otras palabras, un criterio de aceptación es un punto en el que se debe tomar la decisión de aceptar o rechazar, por lo que la aplicación de este concepto a cada medida de calidad establece una base común sobre cómo identificar y determinar el nivel en el que se acepta lo aceptable. Dentro de los estudios analizados, hemos encontrado cuatro autores que explícitamente definieron y mencionaron criterios de aceptación para las medidas de calidad: Philippou et al. (2020), Biscoglio y Marchetti (2014), Zhao et al. (2015) y Schramme y Macías (2019).

Philippou et al. (2020) propusieron una metodología que promueve la integración de los objetivos comerciales, el contexto organizacional y los riesgos de seguridad en el proceso de desarrollo de las medidas de calidad de la seguridad. En este contexto, definieron un conjunto de medidas de calidad a través del enfoque GQM, y el criterio de aceptación es lo que llamaron "interpretación de la medida".

Por ejemplo, para el objetivo: “determinar la efectividad del proceso implementado para brindar capacitación a los nuevos empleados en términos de su finalización exitosa”, la pregunta es: “¿Cómo evaluaríamos si la capacitación es asistida / completada por todos los que deberían asistir / completarla?”, la interpretación de la medición es la siguiente: 1) 0-60% - se requiere intervención; es necesario determinar lo que dificulta la efectividad del proceso, 2) 61-90% - es necesario vigilar la métrica de cerca, ya que podría indicar que algo necesita ser alterado, 3) 91-100% - no hay necesidad de cambio.

Biscoglio y Marchetti (2014), dentro del dominio de preservación de medios audiovisuales, han adoptado el estándar ISO/IEC 25010:2011 y la han utilizado como base para la definición de un modelo de calidad personalizado de acuerdo con las necesidades de la tarea de agregación de almacenamiento. Han definido un conjunto de funciones de medición de características basadas en las características de calidad de ISO/IEC 25010:2011 por ejemplo: *Functional Suitability (FS) = (Functional Completeness + Functional Suitability) / 2*, por lo que definieron un criterio de aceptación del formulario "*El valor FS más cercano a 1 es mejor*". Se siguió el mismo procedimiento para la eficiencia del rendimiento, la compatibilidad, la usabilidad, la confiabilidad, la seguridad, el mantenimiento y la portabilidad.

Posteriormente, Zhao et al. (2015) definió un framework de métricas de software híbrido que respalda la toma de decisiones para desarrollar software de comercio electrónico de alta calidad. Dentro del entorno de computación en la nube, han definido un conjunto de cuatro niveles de falla, cada uno de los cuales tiene un tiempo de respuesta (como <= 3 horas) y un tiempo fijo (como <= 1 día laborable). Además, Schramme y Macías (2019) dentro de su análisis y medición de métricas de usabilidad interna a través de anotaciones de código, han definido un conjunto de criterios de aceptación como valor óptimo.

4.4.6 Conclusiones del catálogo

La medición de la calidad del software permite a los profesionales, gerentes de calidad y gerentes de proyectos comprender qué tan bien el producto de software cumple con sus requisitos funcionales y no funcionales. Con base en el análisis y considerando RQ1, es factible mencionar que se especificaron medidas de calidad para evaluar la calidad del producto para las siguientes características de calidad: Confiabilidad, Idoneidad funcional y Usabilidad.

Ahora bien, con respecto a RQ2, la mayoría de los autores aplicaron el enfoque GQM con las siguientes características de calidad: Seguridad, Usabilidad, Rendimiento, Confiabilidad, Mantenibilidad, Seguridad, Testabilidad y Evolvability.

Si bien no existe una herramienta de automatización que aborde un análisis integral en términos de todas las características de calidad ni resuma el nivel de calidad, con RQ3 fue posible identificar un conjunto de contribuciones a la automatización: un marco que presenta anotaciones de código, un marco que soporta la recopilación automática de medidas de calidad, una herramienta que proporciona medidas automáticas para la mantenibilidad y un

modelo de calidad que cubre los requisitos de calidad para el código fuente que son específicos para el software de sistemas integrados. La automatización es necesaria en un contexto donde el dinamismo es parte de las actividades del día a día, y donde no se debe perder tiempo en medir manualmente.

Con respecto a RQ4 y a partir de los artículos analizados, hemos extraído de los estudios el siguiente conjunto de dominios de aplicación donde se llevó a cabo la medición de la calidad: seguridad social, tecnología aplicada, software y móvil, ingeniería, comunicaciones, finanzas y salud. Aunque no hay una lista larga de dominios, estos son dominios dinámicos y en continuo movimiento, por lo que nos permite entender que si estos dominios mostraron aplicaciones industriales de evaluación de la calidad, entonces se necesita y se requiere la necesidad de automatización.

En base a RQ5, cuatro autores definieron explícitamente el uso de criterios de aceptación para las medidas de calidad (Biscoglio y Marchetti, 2014; Zhao et al., 2015; Schramme y Macías, 2019; Phillippou et al., 2020), mientras que otros definieron umbrales o en inglés, *threshold*s, que pueden definirse como el valor mínimo o máximo (establecido para un atributo, característica o parámetro) que sirve como punto de referencia para comparación u orientación y cuyo incumplimiento puede requerir una revisión completa. revisión de la situación o el rediseño de un sistema (Tsuda et al., 2018).

Si bien la muestra es pequeña en cuanto al total de artículos estudiados, se puede decir que no es una práctica común definirlos, lo cual es un punto a considerar debido a la importancia de definir un criterio de aceptación de una métrica para para lograr confiabilidad y eficiencia al concluir si esa métrica se cumple o no. Del mismo modo, cabe mencionar que no existía una forma sistemática de definición.

En esta línea, donde es extremadamente importante para la industria obtener transparencia y trazabilidad, una herramienta de automatización así como un catálogo de medidas de calidad proporcionarán no sólo una guía de las medidas de calidad aplicadas sino también un medio para identificar para cada producto la calidad características que son más importantes para medir en cada sprint o iteración del producto de software. Se considera que el catálogo de medidas de calidad debe ser útil para cualquier persona que desee consultar o entender qué medidas de calidad se han aplicado en un entorno industrial.

De la misma manera, la definición de PQEM se sustenta en la necesidad de especificar un método que permita analizar la calidad de forma integral, comprendiéndola desde el estudio de cada característica de calidad en cada iteración; sin limitar la cantidad de características ni de requerimientos.

4.5. Resumen de Gaps

Teniendo como base la literatura analizada, es posible resumir en los siguientes ítems los gaps de investigación existentes en el área:

- La literatura denota una variada cantidad de contribuciones que dan cuenta de diversas formas de medición de la calidad, pero las mismas no abordan un estudio completo de todas las características de calidad, como así tampoco describen o proponen una forma de síntesis de dicho análisis y medición. De la misma manera, aún en un contexto digital en crecimiento, no todas las propuestas contienen algún grado de operacionalización; y aquellas que las poseen realizan la medición de unas pocas características calidad, o solo una.

- Los estándares que definen las actividades y los procesos técnicos se han redactado para ayudar a detectar errores lo antes posible en el proceso de desarrollo. Sin embargo, la implementación de dichas pautas de estándares por parte de la industria generalmente se la considera lenta y costosa, lo que no permite un uso y aplicación generalizada.

- La recopilación y el análisis de métricas requieren un esfuerzo significativo para presentar de manera efectiva los resultados de los procesos de medición. Se requieren formas de optimizar los procesos de medición, que aboguen por datos realistas y procesos claros, transparentes y simples.

- Si bien en las últimas décadas, los sistemas de gestión de la calidad han mejorado significativamente, las brechas significativas dentro de la infraestructura, el proceso y la cultura organizacional impiden la implementación de un sistema completo de gestión de calidad basado en datos. Se debe posicionar el foco en lograr programas de métricas consistentes para las

empresas que reduzcan e incluso eliminen las métricas deficientes, que representan un obstáculo importante para lograr los objetivos de calidad.

- Los métodos ágiles se centran en los valores humanos, un aspecto crucial en el desarrollo de software de seguridad, que se basa en la confianza. Estas iniciativas merecen ser incentivadas, posicionándolas y adaptándolas para que cumplan con los requisitos de certificación.

5. Metodología

5.1 Base Conceptual del Método

El paradigma de la ciencia del diseño (del inglés, *Design Science*) tiene sus raíces en la ingeniería y las ciencias de la inteligencia artificial (Hevner y Chatterjee, 2010). Es, fundamentalmente, un paradigma de resolución de problemas, y busca crear innovaciones que definan las ideas, las prácticas, las capacidades técnicas y los productos a través de los cuales el análisis, el diseño, la implementación, la gestión y el uso de los sistemas de información se pueden realizar de manera eficaz y eficiente (Denning, 1997; Tsichritzis, 1998).

Tales artefactos no están exentos de leyes naturales o teorías del comportamiento. Por el contrario, su creación se basa en teorías de kernel existentes que se aplican, prueban, modifican y amplían a través de la experiencia, la creatividad, la intuición y las capacidades de resolución de problemas del investigador (Hevner et al., 2004).

Dentro de la investigación de sistemas de información, el diseño es tanto un proceso (conjunto de actividades) como un producto (artefacto), un verbo y un sustantivo (Walls et al., 1992). Describe el mundo como actuado (procesos) y el mundo como sentido (artefactos). Esta visión platónica del diseño respalda un paradigma de resolución de problemas que cambia continuamente la perspectiva entre los procesos de diseño y los artefactos diseñados para el mismo problema complejo.

El proceso de diseño es una secuencia de actividades de expertos que producen un producto innovador (es decir, el artefacto de diseño). La evaluación del artefacto proporciona información de retroalimentación y una mejor comprensión del problema para mejorar tanto la calidad del producto como el proceso de diseño.

March y Smith (1995) identifican dos procesos de diseño y cuatro artefactos de diseño producidos por la investigación de la ciencia del diseño en sistemas de información. Los dos procesos se construyen y evalúan. Los artefactos son restricciones (del inglés, *constructs*), modelos, métodos e instancias. Los artefactos con un propósito se construyen para abordar problemas no resueltos hasta ahora. Se evalúan con respecto a la utilidad proporcionada para resolver esos problemas.

Los *constructs* proporcionan el lenguaje en el que se definen y comunican los problemas y las soluciones. Los modelos utilizan construcciones para representar una situación del mundo real: el problema de diseño y su espacio de solución. Los métodos definen procesos. Proporcionan orientación sobre cómo resolver problemas, es decir, cómo buscar el espacio de solución. Estos pueden variar desde algoritmos matemáticos formales que definen explícitamente el proceso de búsqueda hasta descripciones textuales informales de enfoques de mejores prácticas, o alguna combinación.

Las instancias muestran que las restricciones, modelos o métodos se pueden implementar en un sistema de trabajo. Demuestran viabilidad, lo que permite una evaluación concreta de la idoneidad de un artefacto para su propósito previsto. También permiten a los investigadores aprender sobre el mundo real, cómo el artefacto lo afecta y cómo los usuarios se lo apropian.

La ciencia del diseño es inherentemente un proceso de resolución de problemas. El principio fundamental de la investigación en ciencias del diseño del que se derivan las siete pautas es que el conocimiento y la comprensión de un problema de diseño y su solución se adquieren en la construcción y aplicación de un artefacto. Las siete pautas, directrices o *guidelines* son:

1. El diseño como artefacto (del inglés, *Design as an artifact*)
2. Relevancia del problema (del inglés, *Problem relevance*)
3. Evaluación de diseño (del inglés, *Design Evaluation*)
4. Contribuciones de investigación (del inglés, *Research Contributions*)
5. Rigor de la investigación (del inglés, *Research Rigor*)
6. El diseño como proceso de búsqueda (del inglés, *Design as a Search Process*)
7. Comunicación de la Investigación (del inglés, *Communication of Research*)

Es decir, la investigación en ciencias del diseño requiere la creación de un artefacto innovador y con un propósito (Pauta, Directriz o *Guideline* 1) para un dominio de problema específico (*Guideline* 2).

Debido a que el artefacto tiene un propósito, debe generar utilidad para el problema especificado. Por tanto, es fundamental una evaluación exhaustiva del artefacto (*Guideline* 3). La novedad es igualmente crucial, ya que el artefacto debe ser innovador, resolviendo un problema no resuelto hasta ahora o un problema conocido de una manera más eficaz o

eficiente (*Guideline* 4). De esta forma, el artefacto en sí debe estar rigurosamente definido, representado formalmente, ser coherente e internamente consistente (*Guideline* 5).

El proceso mediante el cual se crea, y a menudo el artefacto en sí, incorpora o habilita un proceso de búsqueda mediante el cual se construye un espacio problemático y se plantea o promulga un mecanismo para encontrar una solución eficaz (*Guideline* 6).

Finalmente, los resultados de la investigación en ciencia del diseño deben comunicarse de manera efectiva (*Guideline* 7) tanto a una audiencia técnica (investigadores que los extenderán y profesionales que los implementarán) como a una audiencia gerencial (investigadores que los estudiarán en contexto y profesionales que decidirán si deben implementarse dentro de sus organizaciones).

5.2 Método de Investigación

El método de investigación se encuentra embebido dentro del paradigma de la ciencia del diseño (Hevner et al., 2004). Hevner, March, Park, y Ram definieron un conjunto de pautas para ayudar a la comunidad a comprender los requisitos y las necesidades de una investigación científica mediante un diseño eficaz.

Teniendo en cuenta que el paradigma comprende la conceptualización de problemas, el diseño de soluciones y la validación, Runeson y otros (2020) especificaron que puede posicionarse como un marco para la investigación empírica en Ingeniería de Software, con el objetivo de proporcionar conocimientos teóricos para soluciones prácticas relacionadas con los desafíos de la Ingeniería de Software del mundo real. Las siguientes subsecciones contienen la formulación de los guidelines definidos en (Hevner et al., 2004). Vale destacar que el guideline 4 (contribuciones de la investigación) no se describe debajo porque ha sido conceptualizado en la Sección 1.

5.2.1. Diseño como Artefacto

Basado en los desafíos anteriores, PQEM es un método de cinco pasos para cada iteración, que permite a los gerentes y profesionales de calidad estudiar, medir y comprender el

nivel de calidad de un producto de software. El resultado final es un valor único numérico, que puede tomar valores entre 0 y 1, y que representa el nivel de calidad del producto.

Este valor de calidad TOC_i puede pensarse como una medida agregada debido a que se obtiene a través de la cobertura de calidad de cada característica de calidad medida, así como un valor multidimensional, ya que dicho valor TOC_i sintetiza el nivel de calidad alcanzado por cada característica o sub característica de calidad en ese número único.

El componente multidimensional del método PQEM puede entenderse como el grado en que el producto de software cubre el conjunto de requisitos de atributos de calidad (Bass et al., 2003). Cabe mencionar que este proceso se repite después de cada iteración.

5.2.2 Relevancia del Problema

La calidad juega un papel importante para los usuarios finales, debido a que es una confirmación de que todos los requisitos fueron diseñados y desarrollados de acuerdo con sus necesidades (Jain et al., 2018; Neri & Travassos, 2018). Una evaluación de la calidad significativa debe combinar los resultados de varios métodos para responder preguntas específicas, uniendo, por ejemplo, la complejidad ciclomática[6] con la cobertura de testing[7] (Mordal et al., 2018) y, además, la evaluación tiene que poder definir un modelo, desglosado en diferentes características y subcaracterísticas de calidad.

Los gerentes de proyecto y los profesionales de la calidad se enfrentan a diferentes dificultades cuando necesitan comprender el nivel de calidad del producto en cada iteración y del producto completo. Esta comprensión puede ocurrir cuando el gerente de proyecto debe tomar la decisión de aceptar o rechazar los problemas o tareas realizadas como parte de una

[6] **Complejidad Ciclomática** (del inglés, *Cyclomatic Complexity*): es una métrica que proporciona una medición cuantitativa de la complejidad lógica de un programa. El resultado obtenido en el cálculo de la complejidad ciclomática define el número de caminos independientes dentro de un fragmento de código y determina la cota superior del número de pruebas que se deben realizar para asegurar que se ejecuta cada sentencia al menos una vez.

[7] **Cobertura de testing** (del inglés, *test coverage*): es una métrica que mide la cantidad de tests realizados por un conjunto de tests. Incluirá la recopilación de información sobre qué partes de un programa se ejecutan cuando se ejecuta el conjunto de test para determinar qué ramas de declaraciones condicionales se han tomado.

iteración o incluso de la versión completa (Caldiera et al., 1994), cuando debe evaluar el trabajo de los desarrolladores, decidir un pago o negociar una extensión de presupuesto.

Diversos autores han marcado la necesidad de que solo con medidas es posible tomar decisiones de gestión de proyectos sobre una base bien fundamentada (Islam y Falcarin, 2011). En este contexto, PQEM permite monitorear la evolución del nivel de calidad dentro del ciclo de vida del producto, lo cual es un punto diferencial a otros métodos y herramientas existentes. Conjuntamente, se definió un método nuevo por esta necesidad de obtener la evaluación de calidad de un producto software en cada iteración del ciclo de vida, y que resuma la calidad en un número multidimensional.

5.2.3. Evaluación del Diseño

La evaluación del método se realiza a través de un ejemplo ilustrativo de un problema real en el ámbito de la salud, desarrollado en un ámbito académico, aplicándolo a una aplicación web y *mobile* en tres iteraciones. El mismo incluyó la aplicación de PQEM en tres iteraciones, la interacción y retroalimentación continua con los stakeholders, y el análisis de calidad para cada iteración. Detalles completos de los mismos pueden encontrarse en la Sección 8, de Aplicaciones Ilustrativas. Como trabajo futuro, se prevé realizar un caso de estudio dentro de un entorno industrial, con el fin de obtener información sobre el nivel de la aceptación y la utilidad del método PQEM por parte de profesionales y empresas.

5.2.4. Diseño como Proceso de Búsqueda

Se ha diseñado y definido el método PQEM incorporando comentarios de revisores de prestigiosos congresos y journals internacionales, y se ha realizado la validación con aplicaciones móviles y web. Cada uno de los pasos realizados genera una retroalimentación útil para mejorar y optimizar el método propuesto.

5.2.5. Rigor de la Investigación

5.2.6. Comunicación de la Investigación

Las audiencias orientadas a la tecnología reciben detalles suficientes para que puedan replicar cada paso del método PQEM. Además, las audiencias orientadas a la gestión pueden comprender si los recursos de la organización deben comprometerse a utilizar el método dentro de su contexto organizativo específico. Como un objetivo general, por fuera del alcance explícito en la tesis, se busca promover el proyecto de software libre, así como los estudios de casos en la industria.

5.3 Resumen del Proceso

Considerando como base los guidelines caracterizados anteriormente, se describen a continuación los pasos y las actividades comprendidas en el modelo del proceso de la metodología., abordando el entorno completo de la tesis a lo largo del tiempo, abordando tres iteraciones con sus actividades.

Iteración 1

- **Actividad 1.** Identificación del problema y motivación
    - Se buscó conocer el estado del arte de los sistemas multi-agentes (SMA), para conocer los beneficios de este paradigma en el contexto del desarrollo de aplicaciones altamente complejas.
    - Análisis de la literatura en dicho contexto, abarcando medición, métricas de producto y proceso para los sistemas.
- **Actividad 2**. Definir los objetivos para una solución.
    - Los objetivos que se definieron fueron:
        - Conocer la literatura representativa de SMA
        - Definir e implementar métricas de producto de los SMA, abarcando atributos internos (tamaño, complejidad) y atributos externos (disponibilidad, interoperabilidad, modificabilidad, performance, seguridad, testabilidad, usabilidad)

        - Evaluar la calidad de SMA
        - Definir e implementar métricas de proceso de los SMA

- **Actividad 3.** Diseño y desarrollo.
    - Realización de un estudio sistemático de la literatura.
    - Desarrollo aplicaciones en entorno de simulación de SMA
    - Colección de aplicaciones open source de SMA
    - Codificación de métricas
    - Medición de aplicaciones orientadas a agentes
    - Evaluación de la calidad de SMA
- **Actividad 4**. Demostración.
    - Análisis de la literatura entre 2009 y 2017, en el contexto académico e industrial, donde se analizaron 279 artículos, a partir de los cuales es viable mencionar que el área no logró una adopción generalizada para apoyar un cambio de paradigma, y que un SMA puede ser estructurado en base al paradigma orientado a objetos. Una característica clave del campo de SMA es la dispersión, basada en la gran cantidad de revistas, conferencias y dominios de aplicación.
    - En pos de profundizar en la comprensión del paradigma SMA, se han desarrollado dos aplicaciones en entornos de simulación abordando la intersección de calles y la evacuación de un edificio. La definición, implementación y medición de las métricas se realizó en contextos simulados pero reales, aplicando GQM para la definición de las mismas.
    - Con respecto al primer entorno de simulación, se planteó un algoritmo cuyo fin era reducir el tiempo de espera de los conductores en una intersección en NetLogo. La validación fue llevada a cabo en tres escenarios posibles, definidos por medio de la variación de la frecuencia de autos, y se estudió el impacto del cambio de luces comparando semáforos de tiempos fijos con un control

inteligente mediante agentes. Este algoritmo fue extendido a través de la incorporación al modelo de peatones, ambulancias, la colisión entre los autos y la posibilidad de que los autos puedan girar a la izquierda.

    - En base al segundo entorno de simulación, se abordó la influencia específica del cambio espacial en el rendimiento de la evacuación dentro de un museo, explorando la dinámica que afecta a los visitantes cuyo impacto está relacionado con los riesgos del procedimiento de salida del museo, la protección de las obras museográficas, la seguridad y el comportamiento de los visitantes, los cuellos de botella y la situación de alarma de incendio. El simulador consistió en un modelo peatonal de agentes que entran y salen de un museo con una dinámica normal y una dinámica de emergencia.

    - Dichas aplicaciones se midieron a través de la aplicación de GQM.

- **Actividad 5**. Evaluación.

    - El trabajo realizado en el contexto de SMA demostró que es un área de conocimiento con un crecimiento lento, y cuya innovación no presentaba grandes ventanas de oportunidad. Por lo cual, a partir de las publicaciones realizadas y las no aceptadas, fue posible comprender que era necesario ajustar el alcance y abordar el análisis de la calidad no para SMA en particular, sino considerando otras arquitecturas también.

- **Actividad 6.** Comunicación.

    - Submisión de artículos y publicación de resultados.

Iteración 2

- **Actividad 1.** Identificación del problema y motivación

    - Al momento de analizar las aplicaciones disponibles en SMA, se detectó que eHealth era una temática creciente. En correlación con este punto, y en el contacto con especialistas del área de cardiología, se arribó a la problemática relacionada con la falta de organización y la falta de tecnología adecuada para la

asistencia más eficiente a un mayor número de personas, el número de las muertes por trastornos cardíacos aumentan.

- **Actividad 2**. Definir los objetivos para una solución.
    - Desarrollo de un aplicación denominada HeartCare con una arquitectura orientada a agentes implementada con Actores, para seguimiento extrahospitalario de los pacientes cardíacos que deben realizar una rehabilitación.
    - Definición de un framework de atributos de calidad, y su aplicación en HeartCare.
- **Actividad 3.** Diseño y desarrollo.
    - Diseño e implementación de HeartCare.
    - Definición del framework de atributos de calidad
    - Medición de HeartCare con el framework.
- **Actividad 4**. Demostración.
    - Un sistema multiagente fue diseñado y desarrollado para permitir a los pacientes con enfermedades del corazón realizar su rehabilitación fuera del hospital y a sus médicos dar una herramienta web para su seguimiento. Se llegó a desarrollar y evaluar una primera versión de la arquitectura de la aplicación, en la cual se emplearon tecnologías actuales como Akka y Scala para permitir la representación de los agentes a través del modelo del actor. Fue una buena decisión el uso del paradigma orientado al agente, dado que a futuro permitirá escalabilidad de la aplicación, pero no para las altas, bajas y modificaciones.
    - Paralelamente al desarrollo, se definió un Framework de Atributos de Calidad y se aplicó en HeartCare. El alcance del framework es el siguiente:
        - Atributos de Calidad: Availability, Interoperability, Performance, Security, Usability, Modifiability, y Functional Suitability.

        - Estándares: Emergency Department Information Systems (EDIS) - Functional Profile - HL7; HL7 Version 3 Standard: Privacy, Access and Security Services (PASS); Access Control, User-Managed Access (UMA) Profile of OAuth 2.0.; Foundation for Intelligent Physical Agents (FIPA), considerando a FIPA ACL SC00061G (ACLMSS), FIPA Request IP specification (SC00026H), FIPA Ontology Service Specification (XC00086C) (OSS), FIPA Communicative Act Library Specification (SC00037J).

    - La medición de la arquitectura se realizó mediante la construcción de una hoja de cálculo organizada como una hoja por QA, y una hoja única que incluía el resumen de la medición. Cada hoja tenía la lista de ítems donde cada uno de ellos se caracterizó a través de una identificación, una descripción, el control de calidad correspondiente, el tema que trata, el estándar si corresponde, los posibles valores [0 (aprobado) o 1 (fallido)], el resultado obtenido [0 (aprobado) o 1 (fallido)], y una columna final que incluye una breve descripción de cómo se aprobó el elemento. Se analizaron los siguientes ítems: 12 (availability), 31 (interoperability), 18 (performance), 19 (security), 27 (usability), 16 (modifiability), 15 (functional suitability), dando un total de 138 ítems totales.

    - La hoja de cálculo funcionó como una lista de verificación para el control de calidad, donde una vez completada con todos los ítems por control de calidad, fue posible analizar cada elemento para determinar su presencia o ausencia en la arquitectura. En consecuencia, la hoja de cálculo completa ayudó con la organización de los elementos y las respuestas, para abordar más adelante los cálculos de las ecuaciones de cobertura. El Framework de Atributos de Calidad fue una herramienta para cuantificar el nivel de calidad alcanzado en la versión del primer prototipo de arquitectura de HeartCare y una guía para la planificación de versiones posteriores. De la misma manera, un punto a destacar es que se ha podido obtener, con respecto a un criterio de aceptación definido, un número multidimensional que permite conocer el nivel de calidad de la arquitectura.

- **Actividad 5**. Evaluación.

    - En base a diversas devoluciones de reviewers de conferencias internacionales, fue posible comprender que el alcance de un framework de atributos de calidad para sistemas multiagentes era pequeño para un dominio con poco crecimiento y que era necesario un método que permita analizar la calidad de sistemas complejos en general, y no propio para ese dominio en particular.

- **Actividad 6.** Comunicación.
    - Submisión de artículos y publicación de resultados.
    - Comunicación a colegas en jornadas de investigación, en pos de obtener feedback sobre el framework de calidad.

Iteración 3

- **Actividad 1.** Identificación del problema y motivación
    - En base a devoluciones, se comprendió que era posible hacer crecer el alcance abordando a partir del framework, la definición de un método de evaluación de la calidad de un producto.
- **Actividad 2**. Definir los objetivos para una solución.
    - Definición de un método de evaluación de la calidad de un producto, a partir del framework de atributos de calidad.
    - Aplicación del método a aplicaciones.
- **Actividad 3.** Diseño y desarrollo.
    - Definición del alcance y las funcionalidades del método
    - Definición de la estructura, los pasos y los subpasos
    - Definición de la secuencia y las estructuras iterativas
    - Definición de los artefactos necesarios para la realización de la medición en forma manual

- **Actividad 4**. Demostración.
    - El framework permitió definir PQEM (Product Quality Evaluation Method) o Método de Evaluación de la Calidad del Producto. Dicho método se compone de cinco pasos por iteración, cuyo objetivo principal es analizar, estudiar, medir y evaluar el nivel de calidad de las diferentes iteraciones dentro de un producto de software, y que produce un solo valor entre 0 y 1 como resultado final que representa el nivel de calidad del producto. Este es básicamente el grado en que el producto de software cubre / cumple sus requisitos de atributos de calidad.
    - Interacción con stakeholders, y definición de los ejemplos ilustrativos: alcance, funcionalidades esperadas, equipos de trabajo (Desarrolladores, Technical Leaders, Quality Analyst),
    - Desarrollo de aplicaciones web y mobile.
        - Luego de HeartCare, se desarrolló Life+ en la cual se potenció el diseño y usabilidad de la aplicación, pensando en que los usuarios (médicos y pacientes) tengan una menor curva de aprendizaje a la hora de utilizar la aplicación. Life+ poseía dos vistas, una web que utiliza el médico y otra mobile que utiliza el paciente en conjunto con el sensor.
        - A posteriori, se desarrolló la tercera iteración, denominada Corazón Valiente, la cual buscó incrementar la cantidad de funcionalidades, aumentando la concurrencia, e incorporando smart wearables como otra opción en vez de un sensor cardíaco puro.
    - Ejecución de la aplicación del método, habiendo establecido los canales de comunicación con los stakeholders y los equipos
    - Análisis de resultados de medición, con la incorporación de feedback de stakeholders y equipos.
        - Se empleó un artefacto (hoja de cálculo) que funcionó como una lista de verificación para el control de calidad, donde una vez completada con todos los ítems por control de calidad, fue posible analizar cada QAR para determinar su presencia o ausencia en el desarrollo. En

consecuencia, la hoja de cálculo completa ayudó con la organización de los elementos y las respuestas, para abordar más adelante los cálculos de las ecuaciones de cobertura.

    - Creación de un catálogo de métricas, mediante la realización de un estudio sistemático, que contiene el conjunto de métricas construidas a partir de GQM a partir de los artículos que definían los goals, las questions y las metrics; junto con las métricas y sus descripciones obtenidas de otro subset de los artículos analizados.

- **Actividad 5**. Evaluación.
    - Comparación de los objetivos del método con los resultados reales observados en la aplicación del método, en el contexto de las aplicaciones ilustrativas
    - Análisis y decisión relativa a tratar de mejorar la eficacia de la aplicación del método
        - En particular, entre las iteraciones y mediciones, se entendió la necesidad de contar con un set de métricas predefinidas que permita al practitioner disponer de una base sobre la cual seleccionar QARs y métricas; además de poder crear las propias, lo que potenció la creación del catálogo de métricas.
- **Actividad 6.** Comunicación.
    - Realización de publicaciones en conferencias y journals
        - A partir del feedback de reviewers, se trabajó en la mejora del método
    - Comunicación a colegas en jornadas de investigación, en pos de obtener feedback sobre PQEM

6. Product Quality Evaluation Method (PQEM)

6.1 Descripción

Product Quality Evaluation Method (PQEM) es un método de cinco pasos por iteración (ver Figura 2), cuyo objetivo principal es realizar una evaluación de calidad cuidadosa de las diferentes iteraciones dentro de un producto de software, y que produce un valor único entre 0 y 1 como resultado final el cual representa el nivel de calidad del producto, siendo 1 el valor que representa que todos los requerimientos de calidad se han cumplimentado.

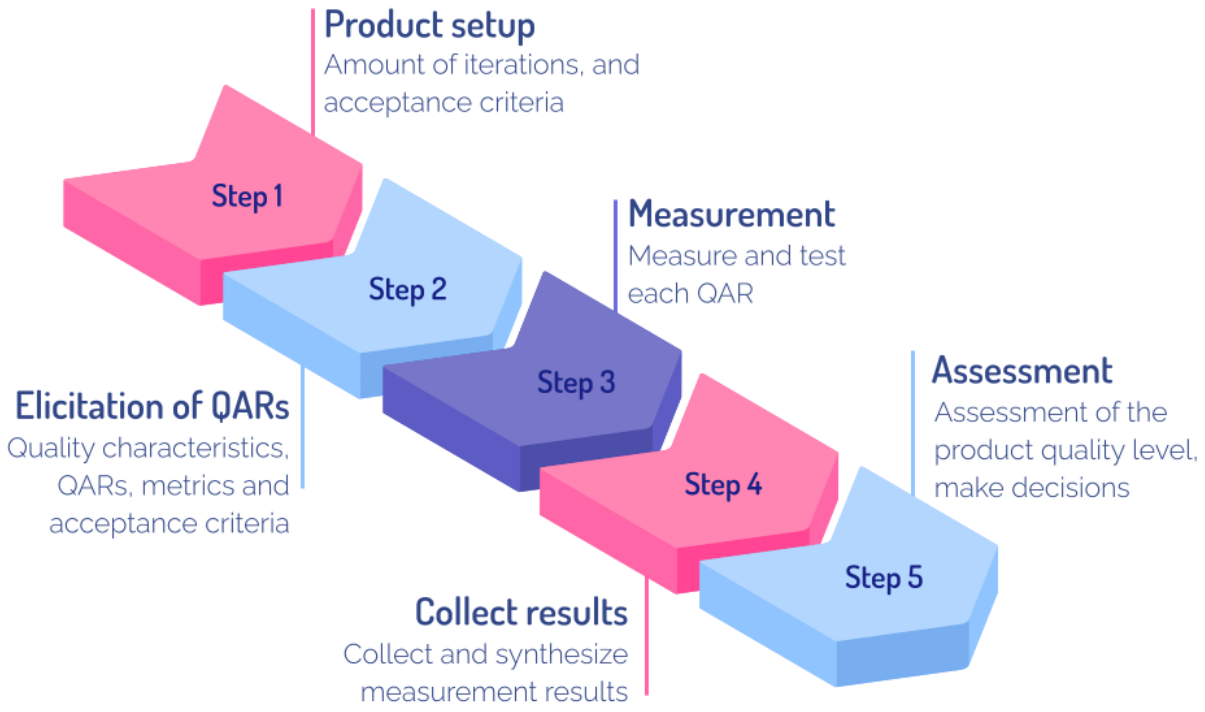

Figura 2. Resumen de los pasos de PQEM para una iteración dentro del ciclo de vida de un producto software. Fuente: Elaboración propia.

Los cinco pasos dentro del método son los siguientes, como se observa en la Figura 2.

1. **Setup del Producto**
2. **Elicitación de los Quality Attributes Requirements (QARs)**
    a. Seleccionar características y subcaracterísticas de calidad
    b. Especificar Quality Attributes Requirements (QARs)
    c. Definir métricas de cada Quality Attribute Requirement (QAR)
    d. Definir los criterios de aceptación de cada Quality Attribute Requirement (QAR)
3. **Medición y testeo de cada Quality Attribute Requirement (QAR)**
4. **Recopilar y sintetizar los resultados**
5. **Evaluación del nivel de calidad del producto**
    a. Recolectar medidas
    b. Decisión y Control

Vale la pena mencionar que el primer paso debe realizarse solo una vez, pero los pasos 2 a 5 deben repetirse en cada iteración para cualquier producto de software. Esto último conduciría a un producto o aplicación de software completamente funcional que se desplegaría para los clientes (Falco & Robiolo, 2019; Falco & Robiolo, 2021).

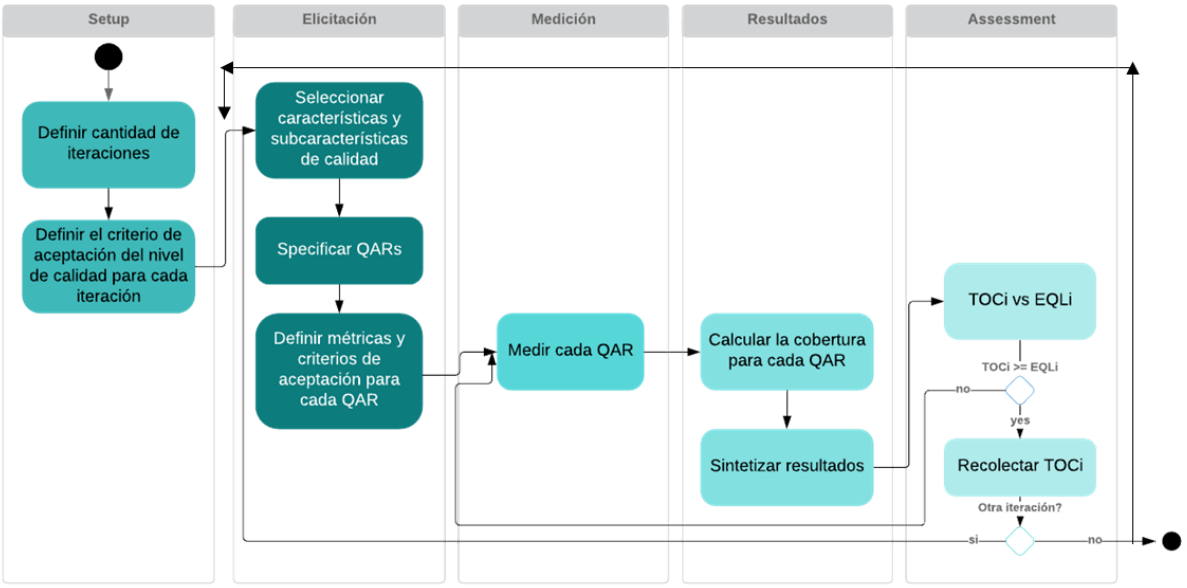

Figura 3. Describiendo PQEM a través de un diagrama de actividades. Fuente: Elaboración propia.

A partir de la Figura 3, se listan las entradas y salidas de cada fase, que serán caracterizadas en profundidad en la subsección siguiente.

1. **Setup del Producto**
    a. Outputs: cantidad de iteraciones, nivel de calidad esperado por iteración
2. **Elicitación de los Quality Attributes Requirements (QARs)**
    a. Inputs: cantidad de iteraciones, nivel de calidad esperado por iteración
    b. Outputs: características y subcaracterísticas de calidad, conjunto de Quality Attributes Requirements (QARs), conjunto de métricas para cara QAR, conjunto de criterios de aceptación para cada métrica definida.
3. **Medición y testeo de cada Quality Attribute Requirement (QAR)**
    a. Inputs: características y subcaracterísticas de calidad, conjunto de Quality Attributes Requirements (QARs), conjunto de métricas para cara QAR, conjunto de criterios de aceptación para cada métrica definida.

b. Outputs: medición de cada QAR resultando en un valor de “pasó-no pasó”

4. **Recopilar y sintetizar los resultados**

    a. Inputs: medición de cada QAR resultando en un valor de “pasó-no pasó”

    b. Outputs: cobertura de calidad para cada característica, nivel de calidad para la iteración

5. **Evaluación del nivel de calidad del producto**

    a. Inputs: nivel de calidad para la iteración

    b. Output: toma de decisión con respecto al nivel de calidad alcanzado

6.2 Caracterización

La presente sección contiene la descripción de cada uno de los pasos que llevan a cabo, como parte de la aplicación del método PQEM.

6.2.1 Paso 1: Setup del Producto

El primer paso de PQEM incluye dos entradas: la definición de la cantidad de iteraciones que se espera que logre el producto de software, así como la caracterización y la justificación de los criterios de aceptación para el nivel de calidad esperado por iteración EQL_i, como se muestra en la Figura 3, y que puede ser diferente e incremental del primero al último (Segue, s.f.).

Este último es un punto clave porque los criterios de aceptación abogan por comprender qué tan bien se logra la calidad para cada objetivo, lo que permite vislumbrar la calidad total del producto, dentro de cada iteración.

De esta manera, el criterio de aceptación lo definen los stakeholders, y es un número positivo que puede tomar cualquier valor entre 0 y 1. Por ejemplo, si se consideran tres iteraciones para un producto software, entonces se puede definir un criterio de aceptación de 0.70, 0.80 y 0.90, respectivamente, para cada una de las tres iteraciones.

Con base en los valores anteriores es posible ver que, en este ejemplo, se espera lograr una mejora en la calidad a medida que el producto crece en funcionalidad. Posteriormente, es factible comprender que 1 es el mejor y más estricto valor de los criterios de aceptación, lo que significa que todos los requisitos de atributos de calidad han pasado la medición; mientras que el 0 equivale a que todos los QARs no resultaron exitosos.

Otro punto a considerar es que PQEM es un método de cinco pasos, pero por iteración sólo se repiten cuatro pasos (del Paso 2 al Paso 5); ya que el Paso 1 define la base de cantidad de iteraciones y criterio de aceptación de cada una de ellas. En cada iteración, al llegar al paso 5 y en base al resultado obtenido del nivel de calidad, se puede retornar al paso 2 (porque es necesario realizar algún ajuste en los QARs definidos o comienza una nueva iteración) o al paso 3 (donde se debe volver a medir aquellos QARs que no pasaron la evaluación).

6.2.2 Paso 2: Elicitación de los Quality Attributes Requirements (QARs)

Hoy en día, la Ingeniería de Sistemas es crucial en la industria, y la Ingeniería de Requisitos es una etapa importante de ese proceso general, donde un proceso adecuado puede generar no solo de manera eficiente sino rápida nuevos productos (Dick et al., 2017). En este contexto, se ha llevado a cabo este proceso de elicitación a través del enfoque *Goal-Question-Metric* (Basili, 1992).

De esta manera:

- el **Paso 2.1** se acercará al nivel conceptual con la definición de metas (considerando la estructura definida en (Caldiera et al., 1994), compuesta de propósito, tema, objeto, y punto de vista - del inglés, *purpose*, *issue*, *object*, *viewpoint*);

- el **Paso 2.2** incluirá el nivel operativo con la especificación de las preguntas por objetivo;
- el **Paso 2.3** especificará el nivel cuantitativo, definiendo las métricas por pregunta.

Es necesario tener en cuenta que los stakeholders deben validar el Paso 2.

6.2.2.1. Paso 2.1: Seleccionar características y subcaracterísticas de calidad

Como explican Estdale & Georgiadou (2018), la norma ISO/IEC 25010:2011 proporciona una gran contribución para establecer la performance del delivery de los diferentes procesos de software. En cuanto al método PQEM, todas las características y subcaracterísticas de calidad pueden ser seleccionadas por los stakeholders; y considerando que cada característica está mapeada con un *goal*, se define por el propósito, tema, objeto y punto de vista para cada característica seleccionada (Basili et al., 1994).

Al crear una instancia del enfoque GQM, puede definirse por ejemplo para *Reliability - Fault Tolerance*:

Goal	Objetivo	Analizar el producto entregado y el proceso de desarrollo
	Problema	Confiabilidad y sus causas
	Proceso	desarrollo
	Punto de vista	del project manager y el usuario

6.2.2.2. Paso 2.2: Especificar Quality Attributes Requirements (QARs)

Bass et al. (2003) explicaron que los requisitos de un sistema se originan en diferentes fuentes y formas, como atributos funcionales y de calidad. Con respecto al Paso 2.1, los stakeholders y el equipo de desarrollo especifican los QARs para cada una de las características de calidad.

Estos QARs son las preguntas definidas para cada uno de los objetivos, y por ejemplo, en el contexto de Disponibilidad un QAR puede ser definido como: “¿El sistema permite mostrar tendencias de signos vitales? (en inglés, *Does the system allows to display trends of vital signs?)* o para Interoperabilidad: "¿El sistema permite capturar y mostrar datos de sensores inalámbricos?" (en inglés, *Does the system allow you to capture and display data from wireless sensors?*) (Jiménez-Fernández et al., 2013).

6.2.2.3. Paso 2.3: Definir Métricas para cada Quality Attribute Requirement (QAR)

En este paso, se definen las métricas que proporcionarán la información necesaria para responder las preguntas definidas en el Paso 2.2. Pueden verse ejemplos de métricas en la Tabla 1, donde para la pregunta: “Are the amount of crashes under control?” la métrica es *Number of crashes*.

Tabla 1. Artefacto para almacenar datos. Ejemplo de un subgrupo de QARs para Tolerancia a Fallos (en inglés, Fault Tolerance) de la segunda iteración de la aplicación.

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

6.2.2.4. Paso 2.4: Definir Criterios de Aceptación para cada Quality Attribute Requirement (QAR)

Al determinar cuándo una historia está completa y funcionando como se esperaba, es posible especificar un criterio de aceptación que contenga las condiciones que un producto de software debe cumplir para ser aceptado por el stakeholder (van Solingen et al., 1999).

Los criterios de aceptación también se tratan al definir qué requisitos deben cumplirse en cada versión incremental de un producto software (Bass et al., 2003). En este contexto, se buscó extender estos conceptos para cada una de las medidas de calidad (del inglés, *quality measures*) con el fin de determinar si este valor medido se cumplió o no, abordando no solo las funcionalidades, sino también las características de calidad.

Como tal, los criterios de aceptación poseen una importancia única debido a que, a través de su definición, es posible conocer objetivamente si cada QAR está presente o no, así como obtener el valor de TOC_i en base a la cobertura de QARs, el cual puede ser desagregado por característica de calidad. Cada medida de calidad requiere de ciertos criterios de aceptación para ser útil y completa. Pueden verse ejemplos de criterios de aceptación en la Tabla 1.

6.2.3. Paso 3: Medición y Testeo de cada Quality Attribute Requirement (QAR)

Este paso implica la medición de cada pregunta, la ejecución de la medida de calidad definida y la descripción de si se cumplieron o no los criterios de aceptación (1 = “pasó”, 0 = “no pasó”). En el caso de la Usabilidad, la medición une las respuestas del número de usuarios que formaron parte del test de Usabilidad. El valor final de cada pregunta de la prueba se obtendrá de la aplicación de la Ecuación (6), que promueve la unificación del número total de respuestas por encuestado, para cada una de las preguntas definidas; permitiendo posteriormente calcular el valor total de calidad.

6.2.4. Paso 4: Recolectar y Sintetizar los Resultados

Con respecto al método de evaluación, es posible mencionar que se han basado las ecuaciones que se presentan a continuación, en el concepto de cobertura de prueba para derivar la cobertura de las diferentes características de calidad y la cobertura total de QAR por iteración. Con base en lo anterior, las Ecuaciones (1) a (6) describen los cálculos necesarios para calcular el nivel de calidad de un producto de software en cada iteración.

$$OC_{qi} = \frac{NpQARqi}{NQARqi} \quad (1)$$

$$EC_{qi} = \frac{NQARqi}{TNQARi} \quad (2)$$

$$O_vC_{qi} = \frac{NpQARqi}{TNQARi} \quad (3)$$

$$TEC_i = \sum_{q=1}^{n} ECqi \quad (4)$$

$$TOC_i = \sum_{q=1}^{n} OvCqi \quad (5)$$

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

donde:

- *q* identifica cada característica de calidad,
- *i* identifica cada iteración,
- *n* es el número total de características de calidad seleccionadas,
- OC_{qi} es la cobertura obtenida para cada característica de calidad en particular por iteración,
- $NpQAR_{qi}$ es el número de QARs que pasaron por característica de calidad, en cada iteración,
- $NQAR_{qi}$ es el número de QAR por característica de calidad para cada iteración,
- EC_{qi} es la cobertura esperada por característica de calidad por iteración,
- $TNQAR_i$ es el número total de QARs por iteración,
- OvC_{qi} es la cobertura general por característica de calidad por iteración,
- TEC_i es la cobertura total esperada por iteración (cuyo valor máximo es 1),
- TOC_i es la cobertura total obtenida de QAR por iteración,
- UC_x es la cobertura para cada QAR de Usabilidad,
- pa_x es el número de respuestas que pasaron para cada QAR,
- re_x es el número de encuestados.

TOC_i es un valor multidimensional porque resume el nivel de calidad obtenido de todas las características y subcaracterísticas de calidad, y es en sí, un indicador de agregación debido a que combina valores de un conjunto de indicadores (los resultados de las coberturas individuales) en una única medida de resumen compuesta o agregada.

Para cada QAR correspondiente a Usabilidad, se obtendrán respuestas *z* según el número de participantes que realicen el test de Usabilidad. Con respecto a esta característica de calidad, cada QAR se analiza de la siguiente manera:

A) Es necesario unificar las respuestas *z* del test de usabilidad que fueron diferentes de 0 y 1 para convertirse en 0 o 1, por ejemplo, aquellas que tengan un valor cualitativo como bajo, muy bajo, medio o alto se pueden unificar definiendo un criterio que todas aquellas respuestas con baja y muy baja se considerarán aprobadas (1) y medias y altas como que no pasaron (0).

B) Luego, se resumen todos los valores (0 y 1) de cada QAR y se obtiene el número que representa los QAR pasados.

C) Posteriormente, la cobertura por QAR se calcula con la Ecuación (6), donde *x* es cada QAR.

Si el valor obtenido con la Ecuación (6) es menor que 0.5 entonces se considera fallido, pasado en caso contrario, obteniendo el valor pq_i para Usabilidad; como la suma de los valores pasados.

D) Finalmente, una vez calculado el valor p_{qi}, es posible calcular la cobertura para la Usabilidad con la Ecuación (1) y (3); y continuar con los cálculos para obtener el valor TOC_i, a través de la Ecuación (5).

6.2.4.1 Análisis matemático

A partir de la medición de la calidad en las normas de la serie ISO/IEC 2502n, se cuantifica la medición mediante funciones matemáticas basados en los modelos especificados en ISO / IEC 25010. En base a las medidas de calidad especificadas en la ISO/IEC 25021, la cual establece las especificaciones de formato y ejemplos para los elementos que permiten construir medidas de calidad.

Las funciones de medición en ISO/IEC 25022[8], ISO/IEC 25023[9] e ISO/IEC 25024[10] describen cómo se combinan los elementos de medición de la calidad para producir la medición de la calidad. Por ejemplo, la cláusula 8.2.1 en la ISO/IEC 25023 define la función de medición de las medidas de completitud de la función con cobertura funcional: $x = \frac{A}{B}$, donde x describe los resultados de la medición, A representa el número de funciones faltantes que se detecta cuando el sistema / software no alcanza la función requerida, y B se refiere al número de funciones especificadas que se pueden obtener en la especificación de requisitos, especificación de diseño o manual del usuario.

Las fórmulas en ISO/IEC 25022, ISO/IEC 25023, ISO/IEC 25024 son funciones lineales como se puede observar en la ecuación anterior, pero esto no es adecuado para todas las situaciones de medición. Consecuentemente, lo que se implementó en PQEM es estandarizar los resultados de la medición entre 0 (requerimiento no logrado) y 1 (requerimiento logrado) para luego calcular el nivel de calidad, sin tener que computar valores en base a los resultados de la medición que puedan o no estar normalizados.

6.2.5. Paso 5: Evaluación del Nivel de Calidad del Producto

Es posible realizar el análisis del nivel de calidad obtenido mediante la Ecuación (5), y la comparación con EQL_i definida por el Paso 1. Una vez que se hayan realizado todas las mediciones del Paso 4, el valor de TOC_i obtenido para la iteración actual se comparará con el nivel de calidad esperado (EQL_i). En esta línea, existen dos posibilidades, siguiendo la Figura 3:

---

[8] **ISO/IEC 25022:2016**, Systems and software engineering — Systems and software quality requirements and evaluation (SQuaRE) — Measurement of quality in use, https://www.iso.org/standard/35746.html

[9] **ISO/IEC 25023:2016**, Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — Measurement of system and software product quality https://www.iso.org/standard/35747.html

[10] **ISO/IEC 25024:2015**, Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — Measurement of data quality, https://www.iso.org/standard/35749.html

- Si el **valor de TOC_i es mayor o igual** que el valor de EQL_i, entonces es posible recopilar el valor de TOC_i.
    - Si hay otra iteración, comenzará el paso de medición.
    - Si no hay otra iteración, el proceso se detiene.
- Si el **valor de TOC_i es menor** que el valor EQL_i, entonces se necesita introducir las mejoras identificadas y realizar una nueva medición para lograr al menos el valor EQL_i.

Como tal, la Ecuación (5) permite construir la Ecuación (7) que contiene la lista de valores TOC_i obtenidos por la iteración *i = (1, 2, ..., y)*. Cabe mencionar que cuando TOC_i es igual o mayor a EQL_i (Nivel de Calidad Esperado para la iteración i, del inglés: *Expected Quality Level*) se realiza la recolección de mediciones.

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

6.3 QTT: Soporte software para PQEM

La aplicación manual del método PQEM insume tiempo al usuario, debido a que la esquematización y puesta en práctica de los primeros pasos definidos requieren un análisis pormenorizado del producto bajo análisis y la definición de los QARs presupone un cierto tiempo para lograr su completitud. En pos de abordar esta consideración, se llevó a cabo el diseño e implementación de una herramienta software denominada *Quality Tracker Tool* (QTT) que busca posicionarse como un soporte a la ejecución del método, que actualmente se está finalizando su desarrollo para comenzar con las pruebas con usuarios.

QTT es una plataforma web, cuyo landing se observa en la Figura 4, a la que puede accederse con el ingreso de un usuario. Al iniciar sesión, se podrá acceder a un barra de navegación que contendrá los productos software, los análisis efectuados y las características de calidad incluidas en los análisis (ver Figura 5).

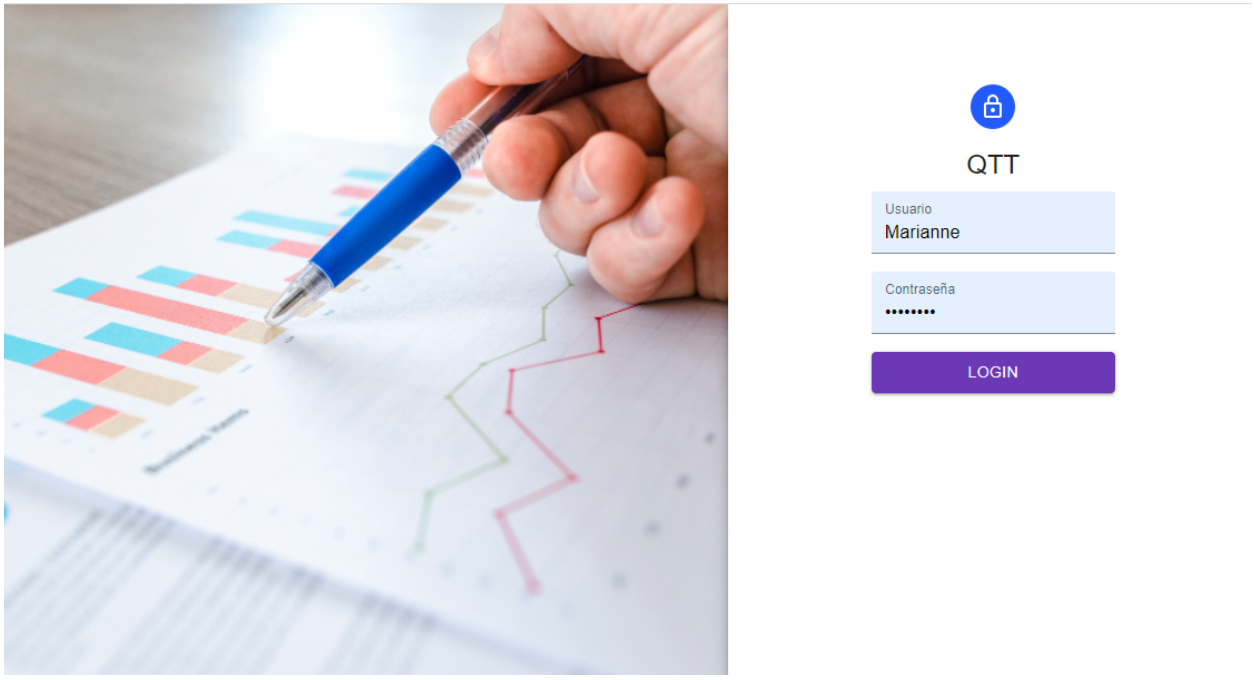

Figura 4. Landing page de QTT. Fuente: Elaboración propia.

En `productos`, será posible incorporar aquellos productos software que se buscará analizar y obtener su nivel de calidad. Una vez creado el producto, en la opción de `análisis` es posible setear las iteraciones y el nivel de calidad esperado para ese producto seleccionado.

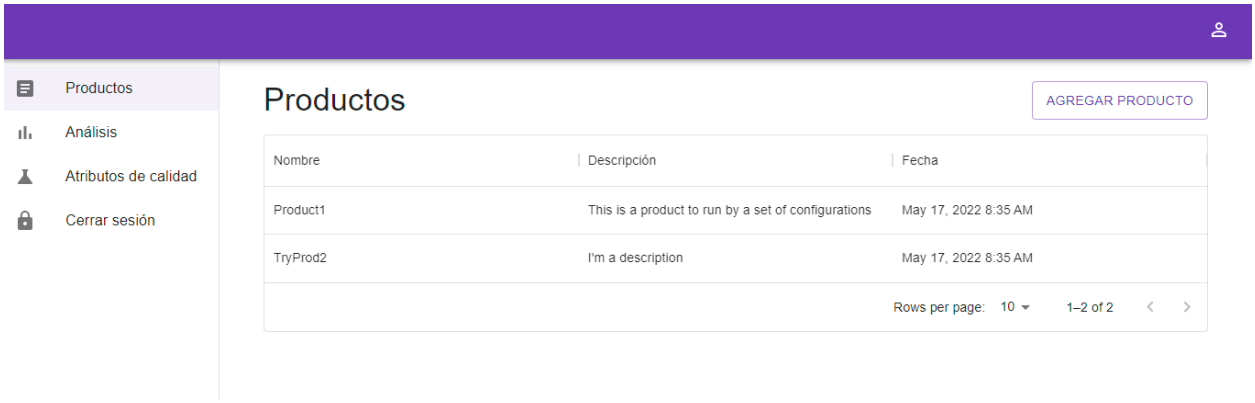

Figura 5. Vista de productos de QTT. Fuente: Elaboración propia.

En `análisis`, es viable cargar un csv con el set de preguntas y los resultados para cada QAR, y desplegará en pantalla los resultados de la iteración, denotando los QARs totales para cada característica de calidad, la cantidad de QARs que han pasado por cada característica y el nivel de calidad alcanzando en esa iteración.

7. Product Quality Evaluation Method con pesos (PQEMw)

7.1 Contexto

En la definición anterior de PQEM, todos los atributos de calidad se consideran de manera uniforme y equitativa, porque los atributos tenían la misma importancia para el stakeholder. Pero los diferentes dominios dan lugar a diferencias en la valoración, ya que algunos atributos de calidad se posicionan sobre otros.

Por ejemplo, en el campo médico, la Interoperabilidad y la Seguridad se establecen como prioritarias sobre otras, pero eso no significa que las otras no sean importantes o necesarias, sino que sólo establecen una idea de jerarquía en una valoración superior para un atributo con respecto al otro (Gritzalis, 1998; Hovenga, 2008; Benson & Grieve, 2016; Blobel, 2020).

Con base en lo anterior, se pensó en definir pesos (del inglés, *weights*) para ser incluidos en PQEM, de acuerdo a los requerimientos del dominio en el que se encuentra embebido el producto software a analizar.

La definición original del nivel de calidad TOC_i implicaba la suma de la cobertura alcanzada por cada atributo de calidad, donde el rango de valores que puede tomar está entre 0 y 1, por lo que cuanto más cerca de 1 está el de calidad TOC_i, mejor será el valor de calidad de esa iteración. El criterio de aceptación es un número que puede tomar valores entre 0 y 1, con tendencia a presentar valores superiores a 0,5; mientras que son definidos por el stakeholder.

7.2 Definición

Las Ecuaciones son las siguientes:

$$O_vCw_{qi} = O_vC_{qi} * W_{qi} \quad (8)$$

$$TOCw_i = \frac{\sum_{q=1}^{n} OvCw_{qi}}{\sum_{q=1}^{n} OvCw_{qi}\, max} \quad (9)$$

donde:

- *q* identifica cada característica de calidad,
- *i* identifica cada iteración,
- *n* es el número de características de calidad definidas,
- O_vC_{qi} es la cobertura general por característica de calidad por iteración,
- W_{qi} es el peso por característica de calidad para cada iteración [que toma valores enteros positivos],
- O_vC_{wqi} es la cobertura general obtenida con pesos por característica de calidad para cada iteración,
- TOC_{wi} es la cobertura total obtenida de QARs por iteración con pesos,
- $\sum_{q=1}^{n} OvCwqi\, max$ es el valor máximo posible para la suma de la cobertura con ponderaciones cuando todos los QARs pasaron dentro de la iteración

Tradicionalmente, la forma más frecuente de definir ponderaciones es con coeficientes, cuya suma da 1. En este caso, no se lo consideró conveniente de aplicar en PQEMw debido a que, si se toma una ponderación entre 0 y 1, la multiplicación entre la cobertura obtenida para la característica de calidad y el peso definido, puede resultar en un valor decimal menor que el obtenido originalmente al calcular dicha cobertura. Dicho evento puede ocasionar al *stakeholder* una interpretación errónea sobre el peso elegido.

Si, por ejemplo, el valor de cobertura obtenido para una característica de calidad fue 0.23 y el peso es 0.11, el resultado TOC_w es 0.025, que es un número menor que 0.23. Esto puede llevar a que el usuario no considere significativo que el valor haya disminuido significativamente y puede preguntar: "*¿Pero luego bajó el nivel de calidad?*" cuando en realidad no bajó.

Tomando esto como base, ahora proponemos una ponderación asociada a cada característica de calidad que puede tomar un valor entero en un rango definido por el interesado, siempre considerando valores enteros positivos. Este valor representa el nivel más alto que se puede considerar al realizar el análisis de calidad, y que establecerá el rango de valores posibles que puede tomar el W_{qi}.

Siguiendo la idea de que este peso solo puede tomar valores enteros positivos, y considerando la lógica propuesta por R. Likert en su método con escalas de evaluación respecto a un conjunto de elementos (que en este caso serían los pesos a elegir para cada calidad atributo), se está considerando un rango entre 0 y n para los pesos, donde n es un valor entero positivo.

Si bien, el stakeholder no especifica su nivel de acuerdo o desacuerdo con respecto a un simétrico de enunciados, la metodología de Likert funciona como base para analizar la idea de elegir un valor viable de este entero positivo que permita luego comprender el nivel de calidad alcanzado, a partir de la caracterización del dominio en el que se inserta el producto a analizar.

Es importante entender que $\sum_{q=1}^{n} OvCwqi\,max$ permite calcular el valor máximo de cobertura de la iteración, en una condición ideal; ese es aquel en el que todos los requisitos de atributos de calidad (QARs) de cada característica de calidad pasaron con éxito. Este resultado permite simplificar el valor de TOC_{wi}, permitiendo que el stakeholder tenga un valor similar

comparable al TOC_i obtenido anteriormente. El hecho de que los pesos sean un número entero ayuda a que el análisis de los resultados sea uniforme y comprensible para el tomador de decisiones. De igual forma, el denominador en la Ecuación (9) es mayor que el numerador, por lo que siempre será un número entre 0 y 1. En el caso de que el numerador sea 0, implica una situación en la que ninguno de los QARs tuvieron éxito. Si el numerador es igual al denominador, entonces es la situación ideal donde todos los QARs pasaron, por lo que estos dos valores extremos llevan a la justificación de que el posible rango de valores se encuentre entre 0 y 1.

7.3 Base Matemática

Para incluir los pesos se utilizó la base matemática presentada por S. Morasca (2010), quien describe el uso de sumas ponderadas en el proceso de definición de medidas en Ingeniería de Software. El autor considera un conjunto dado de *n* pesos con valor real (x_1, ..., x_n) y de *n* pesos con valor real (w_1, ..., w_n) y defina la suma ponderada *ws* como la suma de *i* de *1..n* de w_i * w_i. Para el caso en el que la suma de w_i es igual a 1, se dice que las sumas ponderadas están normalizadas.

A partir de las Ecuaciones (1) a (6), y una vez calculada la cobertura por característica de calidad y el valor de calidad de la iteración, se incluye el peso, que se multiplicará por cada cobertura obtenida, lo que permitirá obtenga un nuevo valor de calidad TOC_w.

Con la incorporación previa al método, una de las primeras cuestiones que surgen es el valor a asignar a cada ponderación, y para ello existen diferentes estrategias que se pueden considerar (Morasca, 2010):

- **Pesos definidos por el medidor** *(del inglés, measurer-defined weights)*: Los medidores son capaces de seleccionar pesos basados en los objetivos de medición y el entorno, por lo que se pueden elegir pesos que varían de un medidor a otro, mientras que el mismo medidor puede seleccionar diferentes pesos en diferentes circunstancias.

- **Pesos subjetivos** *(del inglés, subjective weights)*: en este caso, los pesos pueden ya haber sido definidos o propuestos previamente, y pueden basarse en consideraciones teóricas o experiencia previa.

- **Pesos predeterminados** *(del inglés, default weights)*: es posible seleccionar pesos predeterminados, cuando faltan consideraciones teóricas o experiencia previa. Todas las medidas pueden pesar por igual.

Estas tres estrategias no son incompatibles, por lo que en este contexto, se consideró una integración entre la primera y la tercera estrategia, porque los pesos se eligen en función de los objetivos de la aplicación a los que su nivel de calidad, y considerando su dominio, ya que este último impacta cuál es la ponderación y las características de calidad obligatorias para evitar una distorsión en las mediciones; mientras que la tercera estrategia es el caso cuando todos los pesos son 1, considerando efectivamente que todas las características de calidad pesan lo mismo. Los pesos se definen en la elicitación en conjunto con los usuarios del método, para cada una de las características y, una vez definido, se mantiene a lo largo de todo el ciclo de vida del producto.

8. Aplicaciones Ilustrativas

La presente sección presenta una aplicación ilustrativa del método PQEM, con el fin de ayudar a los profesionales y stakeholders a comprender los pasos y los costos relacionados para realizar la medición.

8.1 Objetivo principal y contexto

El presente ejemplo ilustrativo tiene como objetivo introducir los resultados de la aplicación de PQEM a tres iteraciones de una aplicación web y mobile, cuyo objetivo principal es garantizar que la recuperación de los pacientes cardíacos pueda tener lugar en un entorno fuera de los hospitales (Falco & Robiolo, 2019). Dicha aplicación se encuentra enmarcada en el contexto de una necesidad detectada en el Servicio de Cardiología, a partir de un cliente externo, quien fue partícipe en la selección de los atributos de calidad.

Las tres iteraciones, denominadas HeartCare (Falco et al., 2019), Life+ (Falco et al., 2020) y Corazón Valiente, tienen tres tipos de usuarios y cada uno de ellos utiliza la aplicación en diferentes entornos.

- Los **pacientes** utilizan la aplicación móvil durante sus sesiones de ejercicio, compuesta por la suma de los ejercicios asignados. Cada paciente debe utilizar la aplicación tantas veces como le indique el médico, intentando conseguir la rehabilitación. El paciente puede estar en el interior o al aire libre cuando realiza los ejercicios.

- Los **médicos** utilizan la aplicación siempre que lo consideran necesario para asignar una rutina a un paciente. El médico probablemente esté dentro de su consultorio, en una computadora de escritorio con el paciente presente. El tiempo de uso no debe ser muy prolongado, para no alargar las esperas y el tiempo perdido por el médico. Por lo tanto, el sistema de carga debe ser fácil de usar.

- El **administrador**, por su parte, no utilizará la aplicación de forma recurrente ya que se asume que los cambios de equipos y médicos no ocurren todos los días. El personal administrativo utilizará la aplicación web de su oficina en una computadora.

La primera iteración, incluyó una arquitectura en capas que contiene un sistema multiagente y un sensor de frecuencia cardíaca (Polar H10) que ayuda al paciente a monitorear su condición cardíaca mientras está en posición de reposo, o mientras realiza un ejercicio físico, a través de un teléfono móvil (dispositivo con sistema operativo Android). La literatura describe ejemplos similares a la iteración (Kakria et al., 2015; Gay et al., 2009). Además, dicha iteración se analizó con PQEM y obtuvo un nivel de calidad de 0,775 (Falco & Robiolo, 2019).

La segunda iteración, y en su versión móvil, muestra una serie de rutinas, previamente cargadas por un grupo médico, que serán realizadas por el paciente durante el transcurso de su rehabilitación. El sistema recopila datos sobre la frecuencia cardíaca a través de un sensor cardíaco administrado por el paciente, con el fin de obtener información sobre el estado físico del paciente durante su rehabilitación.

Esa información puede ser vista por un grupo médico a través de una interfaz web para obtener un informe sobre su estado de salud y asignar nuevas rutinas. La arquitectura de la segunda iteración se basa en el principio de diseño en capas y tiene en cuenta la necesidad de desarrollar módulos separados que puedan evolucionar de forma independiente, donde uno de estos módulos es responsable de gestionar el sensor de frecuencia cardíaca.

En cuanto al lenguaje back-end, se eligió un paradigma orientado a objetos, utilizando el lenguaje Java. Spring Framework fue seleccionado porque, de acuerdo con los requisitos de la aplicación de la segunda iteración, era necesario tener una velocidad de lectura y escritura lo suficientemente rápida para soportar el flujo de datos de múltiples sensores, para poder mantener datos estructurados, y es el único que tiene una integración rápida con swagger para la generación automática de documentación.

Luego, la tecnología elegida para el front-end móvil fue *React Native*, ya que permitía la integración con Android y IOs requerida, mientras que para la versión web, se seleccionó Angular con el fin de lograr la modularización, la separación del comportamiento, la separación completa de el back-end, y además, siendo desarrollado por Google, facilita la resolución de problemas que puedan surgir en el futuro. Además, se eligió *Firebase Cloud Messaging* para enviar notificaciones desde el backend a la aplicación móvil, que es desarrollada por Google y se integra fácilmente en Spring Framework.

Finalmente, en la tercera iteración la versión web se desarrolló con Angular, mientras que en la versión *mobile* se utilizó React Native para poder medir la frecuencia cardíaca a

través de Bluetooth (en este caso, un *Android Wear*). En cuanto a las bases de datos, se utilizó PostgreSQL.

8.1.1 Contexto

En el contexto de la aplicación de los ejemplos ilustrativos, y considerando la existencia de stakeholders en el ámbito médico en correlación con una asignatura de una de las carreras de grado de la Facultad de Ingeniería (en la que se logran productos luego de un desarrollo anual en sprints, con clientes reales quienes definen el scope y el alcance anual y por sprint), es necesario puntualizar los siguientes ítems referidos a la interacción de PQEM con los stakeholders y otros roles en el equipo y la comunicación existente:

A. Las aplicaciones fueron desarrolladas en conjunto con el equipo de gestión de la asignatura y los clientes reales del dominio.

B. Para la medición como tal, se consideró una iteración como un conjunto de sprints, debido a que se buscaba medir la aplicación cuando todas las funcionalidades se encontraran implementadas (y dicho evento era viable solo después de varios sprints ejecutados). Vale destacar que cada sprint es mensual, y una iteración puede incluir entre cuatro y cinco sprints.

C. En cuanto a roles, se incluyen el Product Owner, los Technicals Leaders, el Quality Analyst y los desarrolladores.

    a. **Product Owner**: En particular, médico a cargo del Servicio de Rehabilitación Cardiológica, que tiene la capacidad como stakeholder de definir los atributos de calidad a medir en cada iteración y el nivel de calidad esperado para la misma. También, participa en la validación de las iteraciones

    b. **Technical Leaders**: Guían los equipos de desarrollo a lo largo de los sprints, y en particular, son quienes validan lo hecho con lo medido, y aportan su visión técnica de las funcionalidades implementadas junto con el nivel de calidad observable de las mismas.

c. **Quality Analyst**: Es quien realiza la medición de la iteración, y brinda su feedback con respecto a lo medido

d. **Desarrolladores**: Son quienes desarrollan la iteración, pero a su vez, están involucrados en la definición del alcance (en pos de vislumbrar las funcionalidades dentro de la iteración) y la medición, junto al Quality Analyst.

D. Con respecto a las capacidades de los roles:

a. En la **definición**, es el Product Owner quien como dueño de la necesidad toma la decisión del alcance, ayudado por el Quality Analyst.

b. En la **medición**, se encuentran los desarrolladores y el Quality Analysts para cotejar los QARs alcanzados durante el desarrollo.

c. En la **validación**, el Product Owner es quien realiza la validación de lo logrado, junto con los Technical Leaders y el Quality Analyst.

E. En base a los canales de comunicación, el diálogo con los stakeholders ocurre al inicio de la iteración donde se define el alcance y en la medición, pero vale destacar que los stakeholders están involucrados en el progreso de la iteración a través de las demos mensuales de avances de funcionalidades. Para el hito de medición, la comunicación es constante entre todos los roles durante el período de tiempo (entre dos y cinco días) que dure la medición, para establecer una base completa de diálogo sobre el estado y los resultados obtenidos.

F. Referido a la retroalimentación, los resultados de la medición son el punto que posibilita el diálogo y entendimiento del estado actual del producto y la toma de decisión relativa. En base al nivel de calidad obtenido, y al feedback del stakeholder, los Technical Leaders y el Quality Analysts, es viable decidir retrabajar QARs para lograr el nivel de completitud esperado, y luego volver a medir. En base al feedback del stakeholder con respecto a los QARs que no pasaron, pueden también volver a considerarse para la próxima iteración.

8.2 Aplicación de PQEM

La presente sección describe con detalle la aplicación de PQEM a la primera iteración de la app web y mobile, y luego, esboza los resultados de las segunda y tercera iteraciones, junto con un análisis de tendencias y patrones en las tres iteraciones.

8.2.1. PQEM en la Iteración 1

8.2.1.1. **Paso 1: Setup del Producto**

El Paso 1 aborda la definición de la cantidad de iteraciones para un producto de software, así como los criterios de aceptación con respecto a la calidad esperada de las iteraciones, que son definidos como el Nivel de Calidad Esperada (del inglés, *Expected Quality Level* - EQL_i). En el contexto de la aplicación web y móvil, las partes interesadas han definido que tres iteraciones componen el ciclo de vida.

En la misma línea, se definieron los niveles de calidad esperada para cada una de ellas: para la primera iteración se estableció un valor de 0.70, la segunda con un valor de 0.80 y la tercera con un valor de 0.85 (ver Tabla 2). Conceptualmente, se puede entender que, al aumentar el valor correspondiente a los criterios de aceptación, el actor espera aumentar gradualmente el nivel de calidad del producto.

Tabla 2. Iteraciones, nombres y niveles de calidad esperados (EQLi). Fuente: Elaboración propia.

Iteración	Nombre	EQL_i
1	HeartCare	0.70
2	Life+	0.80
3	Corazón Valiente	0.85

8.2.1.2. **Paso 2: Elicitación de los Quality Attributes Requirements (QARs)**

8.2.1.2.1. Paso 2.1: Seleccionar características y subcaracterísticas de calidad

Una vez definida la cantidad de iteraciones y el criterio de aceptación del nivel de calidad esperado para cada una de ellas, en base a la descripción y a los requerimientos del producto software que se desarrollará, se deben seleccionar aquellas características de calidad en base al esquema organizado por la ISO/IEC 25010:2011. En particular, para la primera iteración, se han seleccionado las siguientes características: Availability, Functional Suitability, Interoperability, Modifiability, Performance, Security, y Usability.

8.2.1.2.2. Paso 2.2: Especificar Quality Attributes Requirements (QARs)

A partir de las características de calidad elegidas, es posible definir los QARs para cada una de ellas, lo que conduce a obtener la lista de aspectos que deberán estudiarse en el producto software. Dichos aspectos se redactan en forma de preguntas, siguiendo la estructura de GQM (Basili et al., 1994). Por ejemplo, para Performance Efficiency, un QAR definido dentro de la primera iteración es el siguiente: *What is the average request time?*, como se observa en la columna <Question> de la Tabla 3.

Tabla 3. Artefacto para almacenar datos. Ejemplo de un subset de Performance Efficiency en la primera iteración. Fuente: Elaboración propia.

Quality Characteristic	Question	Metric	Acceptance Criteria	Result
Performance Efficiency	What is the average request time?	Average request time	Less than 200ms	Passed
Performance Efficiency	What is the responsiveness of a web server?	Duration from the user or client making an HTTP request to the first byte of the page being received by the client's browser.	Time to first byte (TTFB) should be less than 250ms	Passed
Performance Efficiency	What is the mobile application size?	Mobile app size	Less than 40mb	Passed
Performance Efficiency	What is the mobile application power consumption?	Mobile power consumption	Less than 3 watts	Failed
Performance	Is the user wait time for the	Number of seconds it takes to	Less than 10	Failed

Efficiency	connection of the sensor with the mobile adequate?	make the connection	seconds	

De esta manera, fue posible definir QARs para cada una de las características de calidad: Availability (12), Interoperability (31), Performance Efficiency (18), Security (19), Usability (27), Modifiability (16), and Functional Suitability (15). En total, se definieron 138 QARs para la primera iteración de la aplicación.

8.2.1.2.3. Paso 2.3: Definir Métricas y Criterios de Aceptación para cada Quality Attribute Requirement (QAR)

Ahora, se hace necesario continuar aplicando GQM y especificar las métricas. Entonces, para cada QAR se definieron métricas en pos de responder cada pregunta de forma cuantitativa. Continuando con el mismo ejemplo de Performance Efficiency, la métrica para ese QAR es: *Request time*, como se observa en la columna <Metric> de la Tabla 3.

8.2.1.2.4. Paso 2.4: Definir Criterios de Aceptación para cada Quality Attribute Requirement (QAR)

Ya teniendo las características de calidad, el conjunto de QARs y las métricas, para concluir el paso 2 es necesario definir los criterios de aceptación para cada uno de los QARs. Es de suma importancia su definición debido a que es el que permite especificar si es parte o no del producto software bajo análisis. Finalmente, la Tabla 3 denota que el criterio de aceptación en el ejemplo empleado es el siguiente: *Less than 200ms*, como se observa en la columna <Acceptance Criteria>.

8.2.1.3. **Paso 3: Medición y Testeo de cada Quality Attribute Requirement (QAR)**

El proceso de medición y testeo de cada QAR consiste en determinar si se cumple o no el criterio de aceptación definido. Entonces, para el ejemplo se efectivizaron pruebas con la aplicación y el average request time obtenido varió entre 30 y 70 ms lo que permitió especificar como correcto ese QAR y por ende, nomenclarlo como <passed> (ver Tabla 3). Los QARs que denotan la etiqueta <failed> implican que no cumplieron con los criterios de aceptación definidos. Cabe mencionar que este proceso debe repetirse para cada QAR definido y, en general, debe ejecutarse para cada iteración.

8.2.1.4. **Paso 4: Recolectar y Sintetizar los Resultados**

Hasta este punto, hemos llenado un artefacto con la colección completa de características de calidad, QARs, métricas, criterios de aceptación y resultados, donde un ejemplo particular se muestra en la Tabla 3. Vale la pena señalar que las etiquetas que se muestran en la Tabla 3, deben reemplazarse con 0 (<failed>) y 1 (<passed>), para poder calcular la cobertura.

Con el fin de obtener el nivel de calidad para la iteración, se hace necesario continuar con la ejecución del Paso 4, empleando las fórmulas que se describen en la Sección 6, las cuales permitirán calcular el valor de cobertura para cada una de las características de calidad definidas.

Partiendo del artefacto completo obtenido como output del paso 3, se deben seguir las siguientes actividades:

1. **Actividad 1**: Contabilizar para cada característica de calidad, la cantidad de QARs definidos. Esto dará un número entero positivo para cada característica.
2. **Actividad 2**: Contabilizar para cada característica de calidad, la cantidad de QARs que han pasado (es decir, aquellos QARs que tienen el tag de <passed>). Esto dará un número entero positivo para cada característica.
3. **Actividad 3**: Calcular la cobertura alcanzada por cada característica de calidad individualmente.
    a. Implica calcular la Ecuación (1) para cada característica de calidad. Por ejemplo, para Performance Efficiency se haría de la siguiente forma: OC_{q1} = $NpQAR_{q1}/NQAR_{q1}$ = 16/18 = 0.89, siendo i = 1, por ser la primera iteración de la aplicación; $NpQAR_{q1}$ es igual a 16, que es la cantidad de QARs que pasaron, y $NQAR_{q1}$ es igual a 18, que es el número de QARs para Performance Efficiency en esta iteración.
4. **Actividad 4**: Calcular la cobertura esperada por cada característica de calidad, en el contexto del número total de QARs para la iteración

a. Es el máximo valor posible de cobertura que cada característica de calidad puede alcanzar dentro de la iteración, y se computa a partir de la Ecuación (2). Para este caso, se obtiene de la siguiente forma: EC_{q1} = $NQAR_{q1}/TNQAR_1$ = 18/138 = 0.130, donde $NQAR_{q1}$ es el número definido de QARs para, en este caso, Performance Efficiency, y $TNQAR_1$ es el número total de QARs para la iteración. A partir de la Ecuación (4) se obtiene la sumatoria de las coberturas esperadas para la iteración, la cual representa la cobertura total esperada que puede tomar como máximo el valor 1.

5. **Actividad 5**: Calcular la cobertura alcanzada por cada característica de calidad, en el contexto del número total de QARs para la iteración.

    a. Para cada característica de calidad, se deberá calcular la Ecuación (3) O_vC_{q1} lo que brindará la cobertura alcanzada dentro de la iteración. En este caso: O_vC_{q1} = $NpQAR_{q1}/TNQAR_1$ = 16/138 = 0.116, donde $NpQAR_{q1}$ es el número de QARs que pasaron por cada característica de calidad, siendo 16 para Performance Efficiency.

6. **Actividad 6**: Obtener el valor de calidad de la iteración

    a. La sumatoria de O_vC_{q1} dará el valor de calidad obtenido por el producto software en la iteración, representado en la Ecuación (5), y es en TOC_1 en donde se sintetizan las coberturas obtenidas de todas las características de calidad, logrando ese número multidimensional de calidad.

El output final del paso 4 será una tabla que contenga la síntesis de los cálculos, donde las columnas representarán los componentes de las fórmulas calculadas. Para la iteración 1, puede observarse dicha síntesis en la Tabla 4.

Tabla 4. Resumen de resultados de la aplicación de PQEM a la primera iteración. Fuente: Elaboración propia.

Quality Characteristics	$NQARq_1$	$NpQARq_1$	OCq_1	ECq_1	$OvCq_1$
Availability	12	12	1.00	0.087	0.087
Interoperability	31	27	0.87	0.225	0.196
Performance Efficiency	18	16	0.89	0.130	0.116
Security	19	9	0.47	0.138	0.065
Usability	27	27	1.00	0.196	0.196
Modifiability	16	1	0.06	0.116	0.007
Functional Suitability	15	15	1.00	0.109	0.109
Total	**$TNQAR_1$ = 138**	**107**		**TEC_1 = 1.00**	**TOC_1 = 0.776**

8.2.1.5. **Paso 5: Evaluación del Nivel de Calidad del Producto**

Este paso tiene dos objetivos: el primero, radica en comparar el nivel de calidad obtenido con el criterio de aceptación definido en el paso 1 para la iteración, es decir: se compara TOC_1 con EQL_1; y el segundo, implica tomar una decisión a partir del resultado de la comparación y recolectar o no el nivel de calidad obtenido considerando la Ecuación (7), que solo es posible de aplicarla cuando TOC_i es igual o mayor a EQL_i.

Para la iteración 1 se definió un nivel de calidad esperado $EQL_1 = 0.70$ y se obtuvo un nivel de calidad $TOC_1 = 0.775$. Puede decirse, por tanto, que la iteración logró alcanzar el nivel de calidad esperado para la iteración, siendo viable de aplicar la Ecuación (7); y por ende, se puede continuar con el paso 2 del método PQEM para la iteración siguiente. Cuando no haya iteraciones restantes, el proceso de medición culmina.

Si hubiera ocurrido el caso de que el resultado de TOC_1 no lograba superar el valor $EQL_1 = 0.70$, será necesario analizar en detalle cuáles fueron los QARs que fallaron, cuáles son

aquellos puntos en que debe trabajarse a nivel código o modelo, y determinar si es necesario modificar o incorporar nuevos QARs en base a esos cambios (se vuelve al Paso 2) o una vez optimizada la aplicación, se realiza una nueva medición para la misma iteración (se vuelve al Paso 3).

8.2.2. PQEM en las Iteraciones 2 y 3

Con respecto a la Elicitación de los Quality Attributes Requirements (QARs), según las necesidades de los stakeholders, la primera iteración incluyó 138 QARs, la segunda iteración incrementó a 258 QARs y, finalmente, en la tercera iteración el número de QARs aumentó a 293; lo que también se vio reflejado en el aumento en la cantidad de características de calidad (CQs) elegidas entre iteración e iteración: 7 CQs en la primera iteración, 10 CQs en la segunda, y 10 CQs en la tercera iteración.

La Tabla 5 muestra el resumen de resultados de la aplicación del método PQEM a la iteración 2, y la Tabla 6, a la iteración 3.

Tabla 5. Resumen de resultados de la aplicación de PQEM a la segunda iteración. Fuente: Elaboración propia.

Quality Characteristics	$NQARq_2$	$NpQARq_2$	OCq_2	ECq_2	$OvCq_2$
Availability	12	10	0.83	0.047	0.039
Fault-Tolerance	5	5	1.00	0.019	0.019
Recoverability	7	5	0.71	0.027	0.019
Functional Suitability	59	56	0.95	0.229	0.217
Interoperability	6	4	0.67	0.023	0.016
Modifiability	59	59	1.00	0.229	0.229
Performance Efficiency	17	15	0.88	0.066	0.058
Security	19	13	0.68	0.074	0.050
Usability	64	58	0.91	0.248	0.225
Portabillity	10	8	0.80	0.039	0.031
Total	**$TNQAR_1$ = 258**	**233**		**TEC_1 = 1.00**	**TOC_1 = 0.903**

Tabla 6. Resumen de resultados de la aplicación de PQEM a la tercera iteración. Fuente: Elaboración propia.

Quality Characteristic	$NQARq_3$	$NpQARq_3$	OCq_3	ECq_3	$OvCq_3$
Availability	14	14	1.00	0.048	0.048
Fault-Tolerance	4	4	1.00	0.014	0.014
Recoverability	7	6	0.86	0.024	0.020
Functional Suitability	57	50	0.88	0.195	0.171
Interoperability	22	12	0.55	0.075	0.041
Modifiability	67	60	0.90	0.229	0.205
Performance Efficiency	17	16	0.94	0.058	0.055
Security	30	27	0.90	0.102	0.092
Usability	64	64	1.00	0.218	0.218
Portabillity	11	11	1.00	0.038	0.038
Total	**$TNQAR_1$ = 293**	**264**		**TEC_1 = 1.00**	**TOC_1 = 0.902**

8.2.2.3. Evaluación del Nivel de Calidad del Producto

La evaluación aborda el análisis del valor obtenido por la Ecuación (5) en base al cálculo previo de la cobertura de todas las características de calidad. Para la segunda iteración, se definió un criterio de aceptación en 0.70; y siguiendo la Tabla 5, el nivel de calidad TOC_2 para la segunda iteración fue 0.90. Como puede verse, el nivel de calidad TOC_2 no solo alcanzó sino que también superó los criterios de aceptación definidos (0.80).

Con respecto a la tercera iteración, había una característica de calidad más nueva (eficiencia de rendimiento), así como 35 QARs más para analizar (donde el 50% pertenecía a la característica de calidad agregada). Considerando en la Tabla 6, el nivel de calidad obtenido para esta tercera iteración fue TOC_3 = 0.90, el cual fue mayor al criterio de aceptación definido para la iteración (0.85), por lo que fue posible aceptar la iteración.

Al analizar cada una de las características de calidad que se muestran en las Tablas 4, 5 y 6, es posible comprender que ninguna de ellas logra una gran diferencia entre la cobertura esperada y la obtenida.

A partir de la Ecuación (5), se puede obtener el siguiente resultado para la Ecuación (7): $TOC_{product}$ *= {0.775; 0.90; 0.90}*, con los valores de TOC_i *(i = 1,2,3)* para cada una de las tres iteraciones.

En este contexto, y debido a que el valor de TOCi fue mayor que el definido por EQL_i, en cada iteración planificada, entonces es posible terminar el ciclo iterativo, debido a que las tres iteraciones alcanzaron o superaron los criterios de aceptación previamente definidos.

8.3 Análisis de tendencias

La Figura 6 muestra la tendencia en los valores de calidad alcanzados en cada una de las tres iteraciones. También permite realizar una comparación visual entre el Valor de Calidad Esperado (EQL) y el valor de calidad obtenido (Cobertura Total Obtenida, *Total Obtained Coverage* o abreviado como TOC_i):

- Para la primera iteración (i = 1), es posible ver $EQL_1 = 0.70$ y $TOC_1 = 0.775$;
- Para i = 2, $EQL_2 = 0.80$ y $TOC_2 = 0.90$;
- Para i = 3, $EQL_3 = 0.85$ y $TOC_3 = 0.90$.

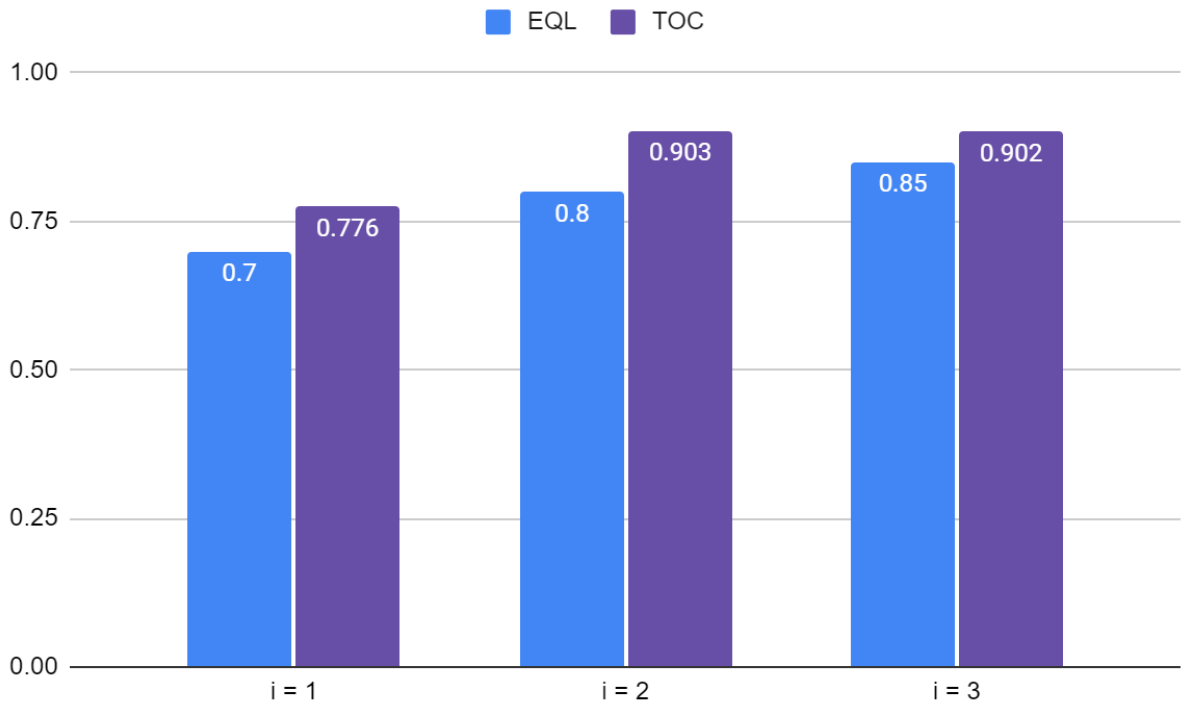

Figura 6. Análisis de tendencias de las tres iteraciones. Fuente: Elaboración propia.

De la Figura 6, es posible entender que a medida que avanzaba la iteración, el criterio de aceptación (o ese valor de calidad esperado) fue aumentando gradualmente, lo que se traduce en un aumento en el nivel de calidad esperado para cada iteración. En otras palabras, tanto los stakeholders como los líderes de calidad coincidieron en que, dada la tasa de variabilidad de las funcionalidades y los requisitos del dominio de cada iteración, los criterios de aceptación debían crecer para adaptarse a esas necesidades.

Afortunadamente, los valores de calidad de TOC_i obtenidos en la primera medición de cada iteración no requirieron una nueva medición, pero fue posible continuar con la siguiente iteración. Esto no implica que no haya puntos a mejorar, al contrario, esas diferencias obtenidas entre EQL_i y TOC_i, que se identifican como deuda técnica para la próxima iteración. La deuda técnica le da la capacidad al gerente de producto o al líder de calidad de identificar y cuantificar las necesidades de calidad no resueltas.

Si se considera la cantidad de QARs por iteración (138, 258, 293) y el nivel de calidad esperado (0.70, 0.80, 0.85), es posible entender que, como hubo ajustes y/o requisitos funcionales que impactaron en la arquitectura o en el modelo de la base de datos, junto con la necesidad de los stakeholders de lograr un mayor nivel de calidad debido a las demandas del dominio, se incrementó la cantidad de QARs por iteración.

Se extrajeron los siguientes corolarios, entre iteración e iteración:

- Se ajustaron los requisitos y se agregaron nuevas funcionalidades.
- El nivel de calidad esperado (EQL_i) se incrementó a solicitud de las partes interesadas.
- Aumentó la cantidad de requisitos de atributos de calidad (QARs) en función del aumento de calidad requerido.
- Se agregaron características de calidad a medida.

La Figura 6 también refuerza la idea de que la aplicación de PQEM permite lograr este análisis comparativo de forma sencilla e intuitiva, con un único valor multidimensional que representa la calidad; y a partir del cual, es posible tomar la decisión de avanzar o no a la iteración, en función de los valores de cobertura obtenidos.

8.4 Ejemplo ilustrativo de PQEMw

El objetivo principal de esta sección es ejemplificar la introducción de pesos en el PQEM. Se utilizarán como base los datos de la tercera iteración.

En este contexto, a la Tabla 6, se le adicionaron dos columnas más, una que contiene los pesos W_{i3} y otra con los resultados de la cobertura obtenida por características de calidad OvC_{wq3}. En este caso, para realizar los cálculos de cobertura, el valor de los pesos variará entre 1 y 5, en pos de obtener luego el valor TOC_{w3}, que proporcionará el valor de calidad de la iteración con los pesos.

En este punto, es importante recordar que el valor de las ponderaciones siempre debe ser un entero positivo. El hecho de que sea un número entero ayuda a que el análisis de los resultados sea uniforme y comprensible para aquel que toma decisiones. En este caso, el valor

w se seleccionó entre 1 y 5, a partir de consideraciones basadas en la experiencia en el uso del método y en los resultados previamente obtenidos.

8.4.1 Resultados de una iteración ideal con PQEMw

Se define un escenario que llamamos “ideal” con la aplicación de PQEMw de la tercera iteración, considerando que la cobertura de cada característica de calidad se alcanzó en un 100%, lo que significa que todos los QARs para cada característica lograron pasar los criterios de aceptación.

En este escenario, se calcula el valor máximo posible de TOC_{w3} para esos pesos seleccionados. La Tabla 7 muestra los valores obtenidos en esa ejecución.

Tabla 7. Resumen de resultados de la aplicación de PQEMw a la tercera iteración cuando todos los QARs pasaron (condición ideal). Fuente: Elaboración propia.

Quality Characteristic	NQARq3	NpQARq3	OCq3	ECq3	OvCq3	Wq3	OvCwq3
Availability	14	14	1.00	0.048	0.048	5	0.239
Fault-tolerance	4	4	1.00	0.014	0.014	4	0.055
Recoverability	7	7	1.00	0.024	0.024	3	0.072
Functional Suitability	57	57	1.00	0.195	0.195	2	0.389
Interoperability	22	22	1.00	0.075	0.075	5	0.375
Modifiability	67	67	1.00	0.229	0.229	3	0.686
Performance Efficiency	17	17	1.00	0.058	0.058	4	0.232
Security	30	30	1.00	0.102	0.102	5	0.512
Usability	64	64	1.00	0.218	0.218	3	0.655
Portability	11	11	1.00	0.038	0.038	2	0.075
Total	**293**	**293**		**1.00**	**1.00**		**3.29**

La Tabla 7 muestra la situación ideal donde todos los QARs están presentes dentro de la aplicación bajo estudio, y también, se incluyó una selección de pesos para analizar este caso, en base a las ideas de los stakeholders y las características del dominio.

Como se observa en la Tabla 7, $\sum_{q=1}^{n} OvCwq3\ max$ = 3.29, y que representa el valor más alto posible que se puede calcular cuando todos los QARs pasaron y para ese conjunto de pesos definidos. En consecuencia, al calcular la Ecuación (9) y realizar 3.29/3.29, el valor resultante es TOC_{w3} = 1.

8.4.2 Resultados de una iteración regular con diferentes pesos

A partir de este valor máximo alcanzado, ahora es necesario analizar los resultados de la aplicación de PQEMw al conjunto de características de calidad con los valores de cobertura reales obtenidos (ver Tabla 6) con la inclusión de los pesos previamente utilizados. La Tabla 8 describe esta ejecución regular con el conjunto de pesos.

Tabla 8. Resumen de resultados de la aplicación de PQEMw a la tercera iteración. Fuente: Elaboración propia.

Quality Characteristic	NQARq3	NpQARq3	OCq3	ECq3	OvCq3	Wq3	OvCwq3
Availability	14	14	1.00	0.048	0.048	**5**	**0.239**
Fault-Tolerance	4	4	1.00	0.014	0.014	**4**	**0.055**
Recoverability	7	6	0.86	0.024	0.020	**3**	**0.061**
Functional Suitability	57	50	0.88	0.195	0.171	**2**	**0.341**
Interoperability	22	12	0.55	0.075	0.041	**5**	**0.205**
Modifiability	67	60	0.90	0.229	0.205	**3**	**0.614**
Performance Efficiency	17	16	0.94	0.058	0.055	**4**	**0.218**
Security	30	27	0.90	0.102	0.092	**5**	**0.461**
Usability	64	64	1.00	0.218	0.218	**3**	**0.655**
Portability	11	11	1.00	0.038	0.038	**2**	**0.075**
Total	**293**	**264**		1.00	**0.901**		**2.92**
					TOCw3 = 2.92/3.29 = 0.88		

A partir de los datos de la Tabla 8, el valor de TOC_{w3} da como resultado 0.88, que se calcula como la división entre la cobertura total obtenida (2.92) y el valor máximo para la iteración ($\sum_{q=1}^{n} OvCwq3\ max$ = 3.29). En este caso, el valor obtenido (0.88) es un valor cercano al valor máximo posible en la escala y comparado con el valor anterior de TOC_i = 0.90, se puede decir que existe similitud entre ellos y, por lo tanto, los pesos previamente seleccionados que multiplican dichas coberturas individuales no afectan el valor resultante y es posible entender dichos valores en términos de calidad.

8.4.3 Análisis de los resultados de PQEM y PQEMw

Las sumas ponderadas (del inglés, *weighted sums*) continúan utilizándose en Ingeniería de Software como un medio para cuantificar un conjunto de atributos de software. La idea es emplear una suma ponderada para combinar varias medidas de nivel inferior para construir una única medida de nivel superior, que cuantifica diferentes aspectos de un atributo al mismo tiempo, y que puede emplear para atributos de software tanto internos como externos (Fenton y Bieman, 2019; Morasca, 2010).

La literatura reporta estudios en los que se han utilizado sumas ponderadas para especificar números únicos que permiten evaluar una funcionalidad, calidad en general u otros atributos, con el fin de entender cuál de ellos tiene mayor grado de funcionalidad (o nivel de completitud), mayor calidad, entre otros.

En particular, se han utilizado estas sumas ponderadas para trabajar con características de calidad y subcaracterísticas basadas en ISO/IEC 25010:2011, con el fin de obtener el nivel de calidad de un conjunto de características en cada iteración del ciclo de vida de un producto software.

Con PQEM es posible obtener el nivel de calidad mediante TOC_i que representa un valor multidimensional, obtenido a partir de la cobertura de cada característica de calidad analizada; mientras que TOC_{wi} resume la cobertura multiplicada por un peso definido por la parte interesada incrustada en el dominio.

Una de las ventajas de utilizar sumas ponderadas en la reducción dimensional, es que son capaces de transformar un problema n-dimensional en uno unidimensional, donde siempre se puede encontrar un orden total. En particular, se ha reducido a un número TOC_i las coberturas de cada característica de calidad, el cual permite comprender el desempeño de calidad de cada iteración dentro del ciclo de vida del producto. La simplicidad es otra ventaja, porque las Ecuaciones (1) a (9) son sencillas de entender, ya que son solo cocientes entre la cantidad de QAR aprobados y el total.

Una de las principales desventajas de utilizar pesos es la confiabilidad de los mismos. Aunque los valores relativos de las ponderaciones pueden cambiar sin afectar el orden de las evaluaciones entre las diferentes opciones, cuando no se tiene una idea sólida de si las

ponderaciones son sensibles a los rangos de variación, podría afectar las evaluaciones finales, dando lugar a sumatorias poco fiables.

Además, es cierto que con un solo número agregado, es posible perder información sobre las medidas originales que pueden ser de utilidad a la hora de tomar una decisión. TOC_i y TOC_{wi} resumen el desempeño de calidad de la aplicación dentro de la iteración, y eso suma las coberturas de cada característica de calidad, no elimina esos valores originales, más bien, los presenta de una manera diferente, por lo que, al realizar ingeniería inversa, es posible obtener rápidamente esos valores de cobertura.

Asimismo, generalmente el resultado final de una suma ponderada es una medida que no tiene una unidad de medida como TOC_i o TOC_{wi}, aunque tienen un rango de valores posibles, no tienen un unidad de medida asociada, sino que esos mismos valores se representan en un gráfico de tendencia del nivel de calidad por iteración.

Otro tema es la supuesta objetividad en cada peso seleccionado, que en nuestro caso es totalmente dependiente del punto de vista de las partes interesadas y la configuración del dominio, que este nivel de subjetividad puede afectar la toma de decisiones del gerente de proyecto cuando tiene que decidir. para pasar a la siguiente iteración o no. Esta subjetividad puede afectar al no poder obtener una medida única para todos que pueda ser una medida totalmente sensible para un atributo interno o una medida útil para un atributo externo en todos los dominios. Es posible determinarlas una vez, en la elicitación de las características y mantenerlas fijas durante el ciclo de vida del producto para no distorsionar los resultados.

9. Amenazas a la Validez

El presente capítulo aborda el análisis de la validez del método definido y los resultados de haber aplicado el método a un conjunto de iteraciones de una aplicación web y mobile embebida en el ámbito médico. En lo que se refiere a análisis de validez y a la clasificación de los distintos tipos de amenazas a la validez de un experimento / ejemplos ilustrativos, Campbell y Stanley (1963) definieron dos tipos, amenazas a la validez interna y externa.

Cook y Campbell (1979) amplían la lista a cuatro tipos de amenazas a la validez de los resultados experimentales. Las cuatro amenazas son validez de conclusión (del inglés, *conclusion validity*), interna (del inglés, *internal validity*), de constructo (del inglés, *construct validity*) y externa (del inglés, *external validity*). De esta manera, cada una de las cuatro categorías presentadas por Cook y Campbell (1979) está relacionada con una cuestión metodológica en la experimentación.

Siguiendo el esquema anterior en base al planteo de Wohlin et al. (2012), las amenazas a la validez se describen de la siguiente manera:

- **Validez de conclusión**: Busca el aseguramiento de la existencia de relaciones con un significado dado.
- **Validez interna**. Busca que el origen de las relaciones provoque el resultado o efecto, no que sea consecuencia de un factor no medido o no considerado.
- **Validez del constructo**. Se refiere a la relación entre teoría y observación, donde es necesario que se refleje la construcción de las causas y que el resultado refleje apropiadamente dicha construcción.
- **Validez externa**. Relacionado con la generalización. Si existe una relación causal entre el constructo de la causa y el efecto, se debe analizar si es posible generalizar el resultado del estudio fuera del alcance definido.

9.1 Amenazas relativas a PQEM

9.1.1 Validez de la conclusión

El principal output tangible de la aplicación del método PQEM es el nivel de calidad, que se expresa como un valor entero positivo en el rango comprendido entre 0 y 1. Esto puede ser considerado como amenaza a la validez, en cuanto a cómo determinar si ese valor es válido o no para determinar un nivel de calidad. Si bien, puede resultar no común en cuanto a resultado final de calidad, en reglas generales, el proceso de medición no siempre se ejecuta correctamente y depende de varios factores que hacen a los distintos atributos propios de cada producto software. Es viable, entonces considerar un número como nivel de calidad debido a que no es un número al azar, sino que es obtenido a partir del análisis de cada característica de calidad en forma individual que luego por sumatorias, se obtiene el valor TOC_i; por ende, representa la multidimensionalidad de la calidad.

Uno de los grandes desafíos de los modelos de calidad de software definidos en las últimas décadas es que no han sido ampliamente aceptados en todos los dominios de aplicaciones, por la subjetividad que subyace a la noción de calidad debido a que el dominio juega un papel importante en la definición de la calidad del software. Los diferentes stakeholders definen la calidad desde diferentes perspectivas (Siavvas et al., 2017). En este contexto, PQEM puede ser aplicado independientemente del dominio porque solo es necesario seleccionar y/o definir los *quality attributes requirements* para cada característica de calidad que apliquen, por lo que el análisis de calidad se puede efectuar para el producto software y en cada iteración sin que el análisis sea incompleto (Heidrich et al., 2014).

9.1.2 Validez interna

El método PQEM conlleva la utilización de un modelo que está basado en el conjunto de características y subcaracterísticas establecido por el estándar ISO/IEC 25010:2011, y que además, define el conjunto de QARs para cada característica y subcaracterística a partir de la utilización del enfoque GQM (Basili et al., 1994). Una amenaza a la validez en esta línea de

pensamiento conlleva a preguntarse si ese conjunto de características es apropiado para soportar el modelo propuesto por PQEM.

Luego, ciertos aspectos del método PQEM pueden parecer extremadamente dependientes de los *stakeholders*. En su forma manual, PQEM es solo un proceso muy abstracto (aunque muy sistemático) que solo se vuelve concreto cuando llega a la agregación métrica al final. Estas condiciones son mejoradas con la creación de un catálogo que incluye métricas y preguntas, las cuales permiten disminuir la dependencia de las partes interesadas y aumentar la aplicabilidad práctica de PQEM; como así también con la incorporación de la herramienta software desarrollada.

Considerando a la calidad como sinónimo de eficiencia, flexibilidad, corrección, confiabilidad, mantenibilidad, portabilidad, usabilidad, seguridad e integridad; el hecho de incorporar en el método un estándar establece una base homogénea a partir de una serie de estándares reconocidos tanto en la academia como en la industria (Gordieiev et al., 2014; Basson et al., 2016; Haoues et al., 2017), lo que también presupone que el modelo que se genera a partir de la aplicación del método permite determinar lo que se debe especificar durante la planificación, la performance y la evaluación de la medición.

Como parte de la validez interna, se puede decir que todos aquellos QARs pertenecientes a Usabilidad tienen una subjetividad reducida debido al número de personas involucradas en los tests de usabilidad realizados. Además de la subjetividad incluida en la evaluación de los QARs, cuando se decidieron aceptarlos o rechazarlos. Pero cabe mencionar que todos ellos fueron definidos para que fueran fácilmente verificables, comprobables o medibles.

De la misma manera, un enfoque reconocido para definir medidas orientadas a objetivos en una organización es GQM (Basili et al., 1994), y si bien, existen estudios donde se han hecho extensiones del mismo o complementos a su esquema de acción, sigue siendo empleado aún en la industria (Beer & Felderer, 2018).

9.1.3 Validez del constructo

Dentro de lo que es la validez del constructo (Wohlin et al., 2012), es conveniente preguntarse si realmente el nivel de calidad obtenido a partir de los cálculos de cobertura

representa la calidad del producto. En respuesta, el nivel de calidad es un valor agregado en base al conjunto completo de QARs, donde la selección de cada QAR fue validada con los stakeholders, por lo que se puede considerar que no es necesario validar el valor obtenido per se. PQEM presenta la evaluación de un sistema en un número. Al hacerlo, asume que todos los requisitos de calidad son igualmente importantes. Vale destacar que, al haber especificado un criterio de aceptación por cada métrica definida, esto permitió efectuar un análisis individual de cada métrica que resulta en una reducción del riesgo anterior, y potenciando la robustez del método.

Ahora bien, PQEMw incorporó un set de pesos para servir de base en la ponderación de las características de calidad que están siendo medidas en una aplicación. Estos pesos no distorsionan el valor de referencia del TOCi, solo modifican los valores de cobertura obtenidos para cada característica de calidad. De esta forma, se podrían definir como mandatorias aquellas características cuando ocurra el caso de que una característica sea tan importante que invalide a otra, y pueda entonces lograrse la validación de la iteración.

Sin embargo, puede ser el caso (y en realidad es así a menudo) que el no cumplimiento de un solo requisito pueda resultar en un producto inutilizable. Este inconveniente está contenido en la generalización de PQEM donde se tiene la definición de un conjunto de pesos que permiten ponderar cada característica de calidad. Aunque pueda parecer pequeña la cantidad de características de calidad seleccionadas, se considera que la comunidad es consciente de la bondad y el alcance científico de la norma ISO/IEC 25010:2011.

Otro punto de interrogación es si el número como tal es representativo de los stakeholders; sobre lo cual se puede decir que el valor de este número surge de todo el desglose previo para todas las características de calidad, al sintetizar el desglose. Por lo tanto, si necesita comprender ese número, es posible pasar por los diferentes niveles de agregación para comprender ese número en profundidad.

9.1.4 Validez externa

Con respecto a la validez externa, una aplicación ilustrativa no es suficiente para poder generalizar el método; tampoco para el caso ilustrativo del uso de los pesos es posible asegurar que el artefacto generado con las nuevas ecuaciones sean útiles en un caso real posicionando la propuesta de sumas ponderadas en un plano teórico. Pero es un punto de

partida. Nos podemos preguntar si la selección de la aplicación para ilustrar el método se realiza en un caso relativamente pequeño donde el sistema puede parecer no significativo y sin aplicabilidad a la práctica industrial. Pero, en realidad, la aplicación posee varias características reales como concurrencia, versión web y móvil, uso de sensores, dominio sanitario, necesidad de alta disponibilidad, entre otras. Vale destacar que se planea a futuro ejecutar la validación con un caso industrial.

Las aplicaciones móviles se están convirtiendo en sistemas de software complejos que deben desarrollarse rápidamente y evolucionar continuamente para adaptarse a los nuevos requisitos de los usuarios y contextos de ejecución (Hecht et al., 2015). Todas estas características dan cuenta de la necesidad de aplicar el método para analizar el nivel de calidad de las tres iteraciones de la aplicación analizadas, debido a la necesidad de entender qué tan bien se diseñó e implementó la aplicación.

En base a las aplicaciones efectuadas, si se posiciona el foco en la replicabilidad sobre lo realizado, es viable lograr valores de cobertura similares, debido a que si bien el análisis y medición lo efectúa el Quality Analysts, son los Technical Leaders los que dan fe de que los valores obtenidos son verídicos con la realidad existente en la estructura interna de la aplicación analizada.

9.2 Amenazas generales y de aplicación

9.2.1 Validez de la conclusión

El método tiene como output un número entre 0 y 1 que explicita el nivel de calidad alcanzado en la iteración. En base a ese valor, es viable detectar como una amenaza la representatividad de ese número obtenido para con la iteración en la cual se lo aplicó. Para emplear un ejemplo regular, puede pensarse en la asignación de una nota a una entrega, en la que la aceptación y correctitud de un conjunto de ítems conllevan a la nota final resultante. En esta lógica, y considerando que el resultado de una iteración es presentado a un stakeholder en forma de demo para que evalúe el avance, PQEM permitió comprender la calidad de entrega y el trabajo de la iteración, a partir de ese número resultante, que conceptualmente puede compararse con esa nota final resultante que reúne un conjunto finito de ítems, y que en particular, también aborda el feedback del stakeholder.

Con base en la literatura reciente, algunos autores han presentado diferentes formas de estudiar un producto software y sus atributos de calidad, llegando también a la medición de métricas. Pero el problema principal, hoy en día, es si los profesionales de gestión y calidad son capaces de identificar correctamente el nivel de calidad de cada iteración dentro de un producto de software. A partir de los valores de calidad obtenidos en cada una de las iteraciones, es factible comprender la evolución del producto, donde el aumento del valor de TOC se debe a un aumento de la calidad, agregando QARs y desagregando aún más las características y subcaracterísticas de calidad elegidas. Considerando la Figura 6 que muestra los niveles de calidad esperados y alcanzados por iteración, una amenaza de validez existente radica en que es necesario revisar cada QAR medido para comprender cuáles fueron los puntos que deben corregirse para lograr una mejora de la calidad. Pero en sí, este punto es mejorado en la idea de que, siendo TOCi un número multidimensional, esa ingeniería inversa es tangible de realizar, y ese número termina resumiendo toda la medición efectuada. De la misma manera, es posible observar rápidamente de la misma figura la deuda técnica, concepto que refleja el costo implícito del retrabajo adicional causado por elegir una solución fácil (limitada) ahora en lugar de utilizar un mejor enfoque que tomaría más tiempo).

9.2.2 Validez interna

El uso de estándares de calidad proveen una base estructural de conceptos y jerarquías en lo que respecta al entendimiento de la calidad del producto, proveyendo características y subcaracterísticas para abordar el análisis y la medición del mismo. Una amenaza de validez en este punto concierne a la identificación precisa de cada QAR como parte o asociado a una característica en particular. En la aplicación del método se buscó agrupar los QARs considerando la lógica conceptual de cada característica, pero sí es cierto, que existen QARs que pueden ser considerados para más de una característica como Interoperabilidad y Seguridad, y que, en particular, fueron organizados en base a la mayor correlación existente por característica.

En la misma línea, considerando que es el stakeholder el que selecciona las características de calidad, para responder a una necesidad de calidad, una amenaza puede aparecer en que el stakeholder tal vez cometa errores o no logre la completitud del alcance de la necesidad al definir las características, y eso conlleve a cometer errores en la definición de preguntas. El hecho de que el stakeholder pueda no definir su necesidad en totalidad, puede

ocurrir cuando no se tiene el completo entendimiento del alcance total del producto en base a la necesidad. Pero esto solo ocurre en un contexto de aprendizaje continuo y mejora progresivo, tanto para el stakeholder como para el analista de calidad, en el apoyo al stakeholder para disminuir los errores en el entendimiento completo del producto y sus requerimientos. También, este proceso evolutivo ayuda a generar una capacidad intuitiva en la medición, útil para las distintas necesidades y dominios.

9.2.3 Validez del constructo

Ahora bien, en cuanto a cuán adecuado es el cálculo definido a partir de coberturas para obtener el nivel de calidad (TOCi), es viable mencionar que el cálculo de cobertura para la calidad se adecuó a partir de lo que se entiende como cobertura de testing, por lo que el cálculo es simple de aplicar y replicar en cada iteración. Es de la misma manera, una forma de emplear porcentajes para entender esos valores obtenidos de calidad ya que TOCi al tomar valores entre 0 y 1 puede ser comparado contra 0 y 100% lo que a la lógica es simple de analizar “cuán bien se logró ese nivel de calidad”; y por ende, es viable guardar la intuición del medidor para validar el producto que se está desarrollando y midiendo.

Actualmente, el método no puede ser aplicado en un contexto ágil, debido a que los métodos ágiles, por definición, no sólo rechazan el modelado sino que cualquier otra actividad que no produzca código de manera rápida. De modo que adoptar este procedimientos de varios pasos podría ser muy bien recibido en un equipo ágil. En este punto, es cierto que puede existir un punto de controversia, pero en particular, los ejemplos ilustrativos descriptos fueron ejecutados considerando un conjunto de sprints como una iteración. De la misma manera, a través de la herramienta y de un catálogo de métricas, se buscará reducir el tiempo en ejecutar el método en pos de que pueda ser aplicado con los menores inconvenientes en un entorno ágil.

10. Conclusiones

Los principios de la Ingeniería de Software y los objetivos de calidad son necesarios pero no suficientes para las necesidades del mercado actual; porque existe la necesidad de tiempos de ciclo más cortos e iterativos, y completos con menos recursos.

Establecer las métricas adecuadas para monitorear el proyecto de software es esencial, así como el requisito de que los gerentes y líderes de proyectos vean el panorama completo y general del proceso de desarrollo (Futrell et al., 2001). Por lo tanto, los líderes de proyectos y propietarios de productos deben comprender el nivel y la calidad de un producto de software de manera intuitiva; lo que facilita la decisión de aceptar o rechazar un producto.

En este contexto, se introdujo un método denominado PQM que evalúa la calidad de un producto mediante un único valor numérico entre 0 y 1. Para calcular este valor, se utilizó un modelo de calidad motivado por GQM que refina los objetivos de calidad a los requisitos de atributos de calidad a partir del estándar ISO/IEC 25010 (pregunta junto con una métrica y criterios de aceptación). La evaluación de la calidad fue derivada del valor de cobertura de los requisitos de atributos de calidad aprobados, como una extensión del concepto de cobertura de testing; y dicha derivación es viable a partir de la agregación de mediciones.

Además, presentamos un estudio de caso del sector de la salud para demostrar su aplicabilidad. Saber qué medir es un problema recurrente en los enfoques basados en datos, el uso de GQM para identificar los atributos de calidad asegura que la evaluación del producto se adapte a la organización que aplica el método PQEM, por haber sido definido como un modelo de evaluación pero también como una guía útil e intuitiva para aumentar la calidad (Mordal et al., 2013).

Es posible visualizar la contribución de PQEM, ya que permitió ayudar a una organización a refinar y concretar sus requisitos de calidad (a menudo abstractos) en criterios estrictos y medibles. En consecuencia, y con PQEM, el gerente puede saber si el proyecto tiene problemas de calidad o si el nivel de calidad está por debajo del esperado; lo mismo que el desarrollador que puede saber cuáles son los puntos de falla.

En función de lo anterior, es viable concluir que, mediante el uso de sistemas de medición, los gerentes pueden concentrarse en unos pocos indicadores clave (métricas con

interpretaciones asociadas) en lugar de monitorear una gran cantidad de métricas. Esto, en consecuencia, conduce a un mayor desempeño de los gerentes y la eficiencia de sus proyectos y organizaciones. Además, el uso de sistemas de medición comparables en múltiples proyectos puede conducir a una evaluación comparativa más fácil de los proyectos o incluso de las organizaciones; naturalmente, si las métricas tienen definiciones apropiadas (Staron et al., 2021).

Como trabajo futuro se prevé culminar con la implementación de la herramienta que representa el método y su validación en un contexto industrial. De la misma manera, sobre el método en particular se buscará generalizar el mismo en pos de que sea posible su utilización en un contexto diferente al de software, modificando el estándar empleado.

Bibliografía

Abran, A., & Robillard, P. N. (1996). Function points analysis: an empirical study of its measurement processes. IEEE Transactions on Software Engineering, 22(12), 895-910.

Ahmed, M., and Ahamad, M. Protecting health information on mobile devices. In Proceedings of the second ACM conference on Data and Application Security and Privacy, pp. 229-240, ACM, 2012.

Alenezi, M. Software architecture quality measurement stability and understandability. International Journal of Advanced Computer Science and Applications (IJACSA), 7(7), 550-559, 2016.

Arpaia, P., De Matteis, E., and Inglese, V. (2015). Software for measurement automation: A review of the state of the art. Measurement, 66, 10-25. Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.

Atoum, I., and Bong, C. H. (2015). Measuring software quality in use: state-of-the-art and research challenges. arXiv preprint arXiv:1503.06934.

Atoum, I. (2020). A novel framework for measuring software quality-in-use based on semantic similarity and sentiment analysis of software reviews. Journal of King Saud University-Computer and Information Sciences, 32(1), 113-125.

Bachmann, F., Bass, L., Klein, M., and Shelton, C. Designing software architectures to achieve quality attribute requirements. IEE Proceedings-Software, 152(4), 153-165, 2005.

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

Basili, V. R., Caldiera, G., and H. Dieter Rombach. "The goal question metric approach." Encyclopedia of software engineering, pp. 528-532, 1994.

Basili, V., Trendowicz, A., Kowalczyk, M., et al., 2014. Aligning Organizations Through Measurement The GQM+ Strategies Approach. Springer-Verlag Berlin Heidelberg.

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

Bass, L., Clements, P., and Kazman, R. Software architecture in practice. Addison-Wesley Professional. 2012.

Basson, H., Bouneffa, M., Matsuda, M., Ahmad, A., Chung, D., & Arai, E. (2016). Qualitative evaluation of manufacturing software units interoperability using ISO 25000 quality model. In Enterprise interoperability VII (pp. 199-209). Springer, Cham.

Benedicenti, L., Wang, V. W., Lee, P., & Paranjape, R. (2001). Establishing quality control in software agents. ACM SIGAPP Applied Computing Review, 9(3), 31-33.

Benson, T., and Grieve, G. (2016). Principles of Health Interoperability. Springer.

Bengtsson, P., Lassing, N., Bosch, J., and van Vliet, H. (2004). Architecture-level modifiability analysis (ALMA). Journal of Systems and Software, 69(1-2), 129-147.

Beer, A., and Felderer, M. (2018, June). Measuring and improving testability of system requirements in an industrial context by applying the goal question metric approach. In Proceedings of the 5th International Workshop on Requirements Engineering and Testing (pp. 25-32).

Biscoglio, I., and Marchetti, E. (2014, September). An experiment of software quality evaluation in the audio-visual media preservation context. In 2014 9th International Conference on the Quality of Information and Communications Technology (pp. 118-123). IEEE.

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.

Boehm, B.W., Brown, J.R., Kaspar, J.R., Lipow, M., and MacCleod, G., Characteristics of Software Quality, Elsevier Science Ltd., Amsterdam, North Holland, 1978.

Campbell, D.T., and Stanley, J.C.: Experimental and Quasi-experimental Designs for Research. Houghton Mifflin Company, Boston (1963)

Cohn, M. (2005). Agile estimating and planning. Pearson Education.

Cook, T.D., and Campbell, D.T.: Quasi-Experimentation – Design and Analysis Issues for Field Settings. Houghton Mifflin Company, Boston (1979)

Consortium for Information and Software Quality (CISQ), Measuring code quality, https://www.it-cisq.org/standards/code-quality-standards/

Corsby, P. B., Quality Is Free: The Art of Making Quality Certain, New York: McGraw-Hill, 1979.

Cyra, Ł., & Górski, J. (2007, October). Extending GQM by argument structures. In IFIP Central and East European Conference on Software Engineering Techniques (pp. 26-39). Springer, Berlin, Heidelberg.

Denning, P. J. "A New Social Contract for Research," Communications of the ACM (40:2), February 1997, pp. 132-134.

Debnath, N., Salgado, C., Peralta, M., Riesco, D., Roqué, L., Montejano, G., and Mazzi, M. (2019, November). A Software Testing Strategy Based on a Software Product Quality Model. In International Conference Europe Middle East & North Africa Information Systems and Technologies to Support Learning (pp. 248-259). Springer, Cham.

Dick, J. , Hull, E., and Jackson, K.. Requirements engineering. Springer, 2017.

Estdale, J., & Georgiadou, E.. "Applying the ISO/IEC 25010 quality models to software product." In European Conference on Software Process Improvement, pp. 492-503. Springer, Cham, 2018.

Falco, M., and Robiolo, G. A Unique Value that Synthesizes the Quality Level of a Product Architecture: outcome of a Quality Attributes Requirements Evaluation Method. In: 20th International Conference on Product-Focused Software Process Improvement (PROFES 2019) - 3rd International Workshop on Managing Quality in Agile and Rapid Software Development Processes (QuASD 2019), (pp. 649-660), Springer.

Falco, M., Scott, E., and Robiolo, G. 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). December 1-4, 2020.

Falco, M., and Robiolo, G. (2021, February). Building a Catalogue of ISO/IEC 25010 Quality Measures Applied in an Industrial Context. In: Journal of Physics: Conference Series (Vol. 1828, No. 1, p. 012077). 2020 International Symposium on Automation, Information and Computing (ISAIC 2020), 2-4 December 2020, Beijing, China.

Falco, M., and Robiolo, G. “PQEM: Product Quality Evaluation Method”, 7th International Conference on Software Engineering (SOFT 2021), July 24-25, 2021, London, United Kingdom.

Fenton, N., and Bieman, J. (2014). Software metrics: a rigorous and practical approach. CRC press.

Falco, M., Bauret, L., and Robiolo, G., “HeartCare: an Agent Oriented Architecture Implemented with Actors”, In: XXV Congreso Argentino de Ciencias de la Computación (CACIC 2019). Universidad Nacional de Río Cuarto, 14 al 18 de Octubre de 2019. Río Cuarto, Córdoba, Argentina.

Falco, M., Robiolo, G., and Salaberri, M., “Plataforma Life+: Un Entorno Flexible para la Rehabilitación de Pacientes Cardíacos”, In: Congreso Argentino de Informática y Salud (CAIS 2020) en las 49 Jornadas Argentinas de Informática (49 JAIIO), Facultad de Ingeniería - UBA, Ciudad Autónoma de Buenos Aires, AR. Octubre 2020.

Fleming, I. (2016). Defining software quality characteristics to facilitate software quality control and software process improvement. In Software Quality Assurance (pp. 47-61). Morgan Kaufmann.

Futrell, R. T., Shafer, L. I., and Shafer, D. F. (2001). Quality software project management. Prentice Hall PTR.

Garai, A. and Adamkó, A. Comprehensive healthcare interoperability framework integrating telemedicine consumer electronics with cloud architecture. In: 2017 IEEE 15th International Symposium on Applied Machine Intelligence and Informatics (SAMI) (pp. 000411-000416). IEEE, 2017.

Gay, V., Leijdekkers, P., and Barin, E. 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.

Gencel, Ç., Petersen, K., Mughal, A. A., and Iqbal, M. I. (2013). A decision support framework for metrics selection in goal-based measurement programs: GQM-DSFMS. Journal of Systems and Software, 86(12), 3091-3108.

Gordieiev, O., Kharchenko, V., Fominykh, N., and Sklyar, V. (2014). Evolution of software quality models in context of the standard ISO 25010. In Proceedings of the Ninth International Conference on Dependability and Complex Systems DepCoS-RELCOMEX. June 30–July 4, 2014, Brunów, Poland (pp. 223-232). Springer, Cham.

Gritzalis, D. A. (1998). Enhancing security and improving interoperability in healthcare information systems. Medical Informatics, 23(4), 309-323.

Habra, N., Abran, A.; Lopez, M., and Sellami, A. A framework for the design and verification of software measurement methods. Journal of Systems and Software, 81(5), 633-648, 2008.

Haoues, M., Sellami, A., Ben-Abdallah, H., and Cheikhi, L. (2017). A guideline for software architecture selection based on ISO 25010 quality related characteristics. International Journal of System Assurance Engineering and Management, 8(2), 886-909.

Hatton, L. (1970). The automation of software processes and product quality. WIT Transactions on Information and Communication Technologies, 4.

Hecht, G., Rouvoy, R., Moha, N., and Duchien, L. (2015, May). Detecting antipatterns in android apps. In Proceedings of the Second ACM International Conference on Mobile Software Engineering and Systems (pp. 148-149). IEEE Press.

Heidrich, J., Rombach, D., and Kläs, M. (2014). Model-based quality management of software development projects. In Software Project Management in a Changing World (pp. 125-156). Springer, Berlin, Heidelberg.

Hevner, A. R., March, S. T., Park, J., and Ram, S. Design science in information systems research. MIS quarterly, pp. 75-105, 2004.

Hevner, Alan, and Samir Chatterjee. "Design science research in information systems." In Design research in information systems, pp. 9-22. Springer, Boston, MA, 2010.

Hoda, R., Salleh, N., and Grundy, J. (2018). The rise and evolution of agile software development. IEEE software, 35(5), 58-63.

Horgan, J. R., London, S., and Lyu, M. R. (1994). Achieving software quality with testing coverage measures. Computer, 27(9), 60-69.

Hovenga, E. J. S. (2008). Importance of achieving semantic interoperability for national health information systems. Texto & Contexto-Enfermagem, 17, 158-167.

Hovorushchenko, T. (2017). Method of Evaluating the Weights of Software Quality Measures and Indicators. Application and Theory of Computer Technology, 2(2), 16-25.

ISO/IEC 25010 (n.d), System and software quality models, Available at: https://iso25000.com/index.php/en/iso-25000-standards/iso-25010, Last access: 1/7/2021.

Islam, S., and Falcarin, P. (2011, September). Measuring security requirements for software security. In 2011 IEEE 10th International Conference on Cybernetic Intelligent Systems (CIS) (pp. 70-75). IEEE.

Ivan, I., Zamfiroiu, A., Doinea, M., and Despa, M. L. (2015). Assigning weights for quality software metrics aggregation. Procedia Computer Science, 55, 586-592.

Jain, P., Sharma, A., and Ahuja, L. "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.

Jiménez-Fernández, S., De Toledo, P., and Del Pozo, F. 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.

Jørgensen, M. Software quality measurement. Advances in engineering software, 30(12), 907-912, 1999.

Juran, J. M., and F. M. Gryna, Jr., Quality Planning and Analysis: From Product Development Through Use, New York: McGraw-Hill, 1970.

Kan, S. H. (2003). Metrics and models in software quality engineering. Addison-Wesley Professional.

Kim, G., Behr, K., & Spafford, K. (2014). The phoenix project: A novel about IT, DevOps, and helping your business win. IT Revolution.

Kitchenham B, Brereton OP, Budgen D, Turner M, Bailey J, Linkman S. Systematic literature reviews in software engineering–a systematic literature review. Information and software technology. 2009 Jan 1;51(1):7-15.

Komi-Sirvio, S., Parviainen, P., and Ronkainen, J. (2001, April). Measurement automation: methodological background and practical solutions a multiple case study. In Proceedings Seventh International Software Metrics Symposium (pp. 306-316). IEEE.

Koziolek, H., Domis, D., Goldschmidt, T., Vorst, P., and Weiss, R. J. (2012, August). 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.

Kakria, P., Tripathi, N. K., and Kitipawang, P. A real-time health monitoring system for remote cardiac patients using smartphone and wearable sensors. International journal of telemedicine and applications, 2015.

Kazman, R., Klein, M., and Clements, P. ATAM: Method for architecture evaluation (No. CMU/SEI-2000-TR-004). Carnegie-Mellon University Pittsburgh PA Software Engineering Institute, 2000.

Kazman, R., Nord, R. L., and Klein, M. (2003). A life-cycle view of architecture analysis and design methods (No. CMU/SEI-2003-TN-026). Carnegie-Mellon University Pittsburgh PA Software Engineering Institute.

Krasner, H. (2021, January). The cost of poor software quality in the US: A 2020 report. In Proc. Consortium Inf. Softw. QualityTM (CISQTM).

Lavazza, L. (2000). Providing automated support for the GQM measurement process. IEEE Software, 17(3), 56-62.

Lavazza, L., Morasca, S., & Robiolo, G. (2013). Towards a simplified definition of Function Points. Information and Software Technology, 55(10), 1796-1809.

Li, Z., Avgeriou, P., and Liang, P. A systematic mapping study on technical debt and its management. Journal of Systems and Software, 101, 193-220, Elsevier, 2015.

Lopes, P. and Oliveira, J. L. A semantic web application framework for health systems interoperability. In Proceedings of the first international workshop on Managing interoperability and complexity in health systems (pp. 87-90). ACM, 2011

March, S. T., and Smith, G. "Design and Natural Science Research on Information Technology," Decision Support Systems (15:4), December 1995, pp. 251-266.

Mäntylä, M. V., and Lassenius, C. (2006). Subjective evaluation of software evolvability using code smells: An empirical study. Empirical Software Engineering, 11(3), 395-431.

Mayr, A., Plöesch, R., Kläs, M., Lampasona, C., and Saft, M. (2012, November). A comprehensive code-based quality model for embedded systems: systematic development and validation by industrial projects. In 2012 IEEE 23rd International Symposium on Software Reliability Engineering (pp. 281-290). IEEE.

Michael, J. B., Shing, M. T., Cruickshank, K. J., and Redmond, P. J. (2010). Hazard analysis and validation metrics framework for system of systems software safety. IEEE Systems Journal, 4(2), 186-197.

Mishra, A., and Otaiwi, Z. (2020). DevOps and software quality: A systematic mapping. Computer Science Review, 38, 100308.

Mordal, K., Anquetil, N., Laval, J., Serebrenik, A., Vasilescu, B., and Ducasse, S. (2013). Software quality metrics aggregation in industry. Journal of Software: Evolution and Process, 25(10), pp. 1117-1135, 2013.

Morasca, S., and Briand, L. C. (1997, November). Towards a theoretical framework for measuring software attributes. In Proceedings Fourth International Software Metrics Symposium (pp. 119-126). IEEE.

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).

McCall J.A., Richards P.K., and Walters, G.F., Factors in software quality, RADC TR-77-369, 1977. Vols I, II, III, US Rome Air Development center Reports NTIS AD/A-049 014, 015, 055, 1977

Neri, H. R. and Horta Travassos, G.. "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.

Nuñez-Varela, A. S., Pérez-Gonzalez, H. G., Martínez-Perez, F. E., and Soubervielle-Montalvo, C. (2017). Source code metrics: A systematic mapping study. Journal of Systems and Software, 128, pp. 164-197, 2017.

Oinas, A. (2000, March). Defining goal-driven fault management metrics in a real world environment: a case-study from Nokia. In Proceedings of the Fourth European Conference on Software Maintenance and Reengineering (pp. 101-107). IEEE.

Ortega, M., Pérez, M. and Rojas, T. Construction of a systemic quality model for evaluating a software product. Software Quality Journal, 11(3), 219-242, 2003.

Pradhan, S., & Nanniyur, V. (2021). Large scale quality transformation in hybrid development organizations–A case study. Journal of Systems and Software, 171, 110836.

Plöesch, R., Schürz, S., and Körner, C. (2015, July). On the validity of the it-cisq quality model for automatic measurement of maintainability. In 2015 IEEE 39th Annual Computer Software and Applications Conference (Vol. 2, pp. 326-334). IEEE.

Poels, G., & Dedene, G. (2000). Distance-based software measurement: necessary and sufficient properties for software measures. Information and Software Technology, 42(1), 35-46.

Rostami, K., Stammel, J., Heinrich, R., and Reussner, R. Architecture-based assessment and planning of change requests. In Proceedings of the 11th International ACM SIGSOFT Conference on Quality of Software Architectures, pp. 21-30. ACM, (2015, May).

Runeson, P., Engström, E., and Storey, M. A. The design science paradigm as a frame for empirical software engineering. In Contemporary empirical methods in software engineering, pp. 127-147. Springer, Cham, 2020.

Samoladas I., Gousios G., Spinellis D., and Stamelos I. (2008) The SQO-OSS Quality Model: Measurement Based Open Source Software Evaluation. In: Russo B., Damiani E., Hissam S., Lundell B., Succi G. (eds) Open Source Development, Communities and Quality. OSS 2008. IFIP – The International Federation for Information Processing, vol 275. Springer, Boston, MA. https://doi.org/10.1007/978-0-387-09684-1_19

Sarkar, S., Rama, G. M., and Kak, A. C. (2006). API-based and information-theoretic metrics for measuring the quality of software modularization. IEEE Transactions on Software Engineering, 33(1), 14-32.

Schramme, M., and Macías, J. A. (2019). Analysis and measurement of internal usability metrics through code annotations. Software Quality Journal, 27(4), 1505-1530.

Schrettner, L., Fülöp, L. J., Beszédes, Á., Kiss, Á., and Gyimóthy, T. (2012, March). Software quality model and framework with applications in industrial context. In 2012 16th European Conference on Software Maintenance and Reengineering (pp. 453-456). IEEE.

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.

Shore, J., and Warden, S. (2021). The art of agile development. " O'Reilly Media, Inc.".

Siavvas, M. G., Chatzidimitriou, K. C., and Symeonidis, A. L. (2017). QATCH - An adaptive framework for software product quality assessment. Expert Systems with Applications, 86, 350–366. doi:10.1016/j.eswa.2017.05.060

Siavvas, M., Kehagias, D., Tzovaras, D., & Gelenbe, E. (2021). A hierarchical model for quantifying software security based on static analysis alerts and software metrics. Software Quality Journal, 29(2), 431-507.

Software Testing Help (April 16, 2020), "What Is User Story And Acceptance Criteria (Examples)", available at: https://www.softwaretestinghelp.com/user-story-acceptance-criteria/, last accessed: October 27, 2021

Staron, M., Meding, W., and Nilsson, C. (2009). A framework for developing measurement systems and its industrial evaluation. Information and Software Technology, 51(4), 721-737.

Tan, J., Rönkkö, K., and Gencel, C. (2013, October). A framework for software usability and user experience measurement in mobile industry. In 2013 joint conference of the 23rd international workshop on software measurement and the 8th international conference on software process and product measurement (pp. 156-164). IEEE.

Tsichritzis, D. "The Dynamics of Innovation," in Beyond Calculation: The Next Fifty Years of Computing, P. J. Denning and R. M. Metcalfe (eds.), Copernicus Books, NewYork, 1998, pp. 259-265.

Tsuda, N., Washizaki, H., Fukazawa, Y., Yasuda, Y., and Sugimura, S. (2018, July). Machine Learning to Evaluate Evolvability Defects: Code Metrics Thresholds for a Given Context. In 2018 IEEE International Conference on Software Quality, Reliability and Security (QRS) (pp. 83-94). IEEE.

Urovi, V., Olivieri, A. C., Bromuri, S., Fornara, N., and Schumacher, M. I. A peer to peer agent coordination framework for IHE based cross-community health record exchange. In Proceedings of the 28th Annual ACM Symposium on Applied Computing, pp. 1355-1362, ACM, 2013.

Van Solingen, R., and Berghout, E. W. The Goal/Question/Metric Method: a practical guide for quality improvement of software development. McGraw-Hill. 1999.

Wingkvist, A., Ericsson, M., Lincke, R., and Löwe, W. (2010, September). A metrics-based approach to technical documentation quality. In 2010 Seventh International Conference on the Quality of Information and Communications Technology (pp. 476-481). IEEE.

Walls, J. G., Widmeyer, G. R., and El Sawy, O. A. "Building an Information System Design Theory for Vigilant EIS," Information Systems Research (3:1), March 1992, pp. 36-59.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., and Wesslén, A. (2012). Experimentation in software engineering. Springer Science & Business Media.

Woods, E. Industrial architectural assessment using TARA. Journal of Systems and Software, 85(9), pp. 2034-2047, (2012).

Yang, Q., Li, J. J., and Weiss, D. M. (2009). A survey of coverage-based testing tools. The Computer Journal, 52(5), 589-597.

Zhao, F., Nian, G., Jin, H., Yang, L. T., and Zhu, Y. (2015). A hybrid e-business software metrics framework for decision making in cloud computing environment. IEEE Systems Journal, 11(2), 1049-1059.

Zhao, Y., Gong, J., Hu, Y., Liu, Z., and Cai, L. (2017, May). Analysis of quality evaluation based on ISO/IEC SQuaRE series standards and its considerations. In 2017 IEEE/ACIS 16th International Conference on Computer and Information Science (ICIS) (pp. 245-250). IEEE.

Anexo

Se incluye el hipervínculo al artefacto que contiene el análisis de las tres iteraciones (Artifact - iterations).