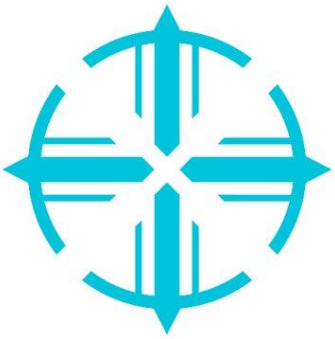
PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR

ESMERALDAS



ESCUELA DE INGENIERÍA DE SISTEMAS Y COMPUTACIÓN

TESIS DE GRADO

"EVALUACIÓN DEL SISTEMA VIÁTICOS DEL GOBIERNO AUTÓNOMO DESCENTRALIZADO DE LA PROVINCIA DE ESMERALDAS (GADPE) BASADO EN LA NORMA ISO 25000"

LÍNEA DE INVESTIGACIÓN

GOBIERNO DE TECNOLOGÍAS DE LA INFORMACIÓN

Previo a la obtención del título Ingeniero de Sistemas y Computación

AUTOR:

BAUTISTA ANGULO JENNIFFER VERÓNICA

ASESORA:

MGT. SUSANA PATIÑO

Esmeraldas, Ecuador, 2019

Tesis de grado aprobada luego de haber dado cumplimiento a los requisitos exigidos por el reglamento de Grado de la PUCE-E, previo a la obtención del título de INGENIERO DE SISTEMAS Y COMPUTACIÓN.

TRIBUNAL DE GRADUACIÓN

Título: "EVALUACIÓN DEL SISTEMA VIÁTICOS DEL GOBIERNO AUTÓNOMO DESCENTRALIZADO DE LA PROVINCIA DE ESMERALDAS (GADPE) BASADO EN LA NORMA ISO 25000"
Autor: BAUTISTA ANGULO JENNIFFER VERÓNICA

……

Mgt. Susana Patiño Rosado

Asesora

……

Mgt. Evelin Flores

Lector 1

……

Mgt. Jaime Sayago Heredia

Lector 2

……

Mgt. Xavier Quiñónez Ku

Director de Escuela de Sistemas y Computación

……....

Ing. Maritza Demera Mejía

Secretaria General PUCE-E

Esmeraldas, Ecuador, 2019

AUTORÍA

Yo, JENNIFFER VERÓNICA BAUTISTA ANGULO portadora de la cédula de identidad No. 0804471548 declaro mediante la presente que los resultados en la investigación que presento como tesis de grado, previo a la obtención del título de **"**Ingeniero de Sistemas y Computación" son absolutamente originales, personales y legítimos.

En tal virtud, declaro que el contenido ciñendo resultados, conclusiones, efectos legales y académicos que se desprenden en el trabajo de investigación propuesto son y serán de exclusiva responsabilidad académica y legal.

.....……

JENNIFFER VERÓNICA BAUTISTA ANGULO

CI: 080447154-8

CERTIFICACIÓN

Mgt. Susana Patiño, docente investigador de la PUCE-Esmeraldas, certifica que:

La investigación realizada por BAUTISTA ANGULO JENNIFFER VERÓNICA bajo el título "EVALUACIÓN DEL SISTEMA VIÁTICOS DEL GOBIERNO AUTÓNOMO DESCENTRALIZADO DE LA PROVINCIA DE ESMERALDAS (GADPE) BASADO EN LA NORMA ISO 25000" reúne los requisitos de calidad, originalidad y presentación exigibles a una investigación científica, además de haber sido incorporadas al documento final las sugerencias realizadas, en consecuencia, está en condiciones de ser sometida a la valoración del Tribunal encargada de juzgarla.

Y para que conste a los efectos oportunos, firma la presente en Esmeraldas, mayo del 2019.

Mgt. Susana Patiño
ASESORA

DEDICATORIA

El presente trabajo de investigación se lo dedico principalmente*:*

A Dios, por darme la oportunidad de mantenerme con vida y por siempre estar conmigo en cada paso que doy en el camino, por día a día iluminar mi mente y fortalecer mi corazón en toda circunstancia sea esta buena o mala y por haber puesto en mi camino a aquellas personas que han sido mi soporte y compañía a lo largo de mi periodo académico.

A mis padres por ser el estribo fundamental en todo lo que soy ahora, en todas las enseñanzas tanto espiritual, como educativa, por todo apoyo, amor y sacrificio durante todos estos años.

A mi hijo y esposo por ser un soporte más en mi camino, por ser esa fuente de motivación e inspiración en momentos de decline y así poder superarme cada día más para que la vida nos depare un mejor futuro.

Todo este trabajo ha sido posible gracias a ellos.

AGRADECIMIENTOS

En primer lugar, me gustaría agradecer ***a Dios*** por enaltecer la oportunidad de llegar hasta donde he llegado, porque este sueño y meta tan anhelada se pudo cumplir.

A mis padres y hermanas por todo el apoyo brindado en cada momento, por los valores inculcados y por haberme dado la oportunidad de tener una excelente educación a lo largo de mi vida.

A mi asesora de tesis, Mgt. Susana Patiño Rosado por su esfuerzo, tiempo y dedicación, quien, con sus conocimientos, experiencia, paciencia y motivación ha logrado en mí que pueda terminar mi trabajo de investigación con éxito.

También me gustaría agradecer ***a mis profesores y compañeros de curso*** con los que pasamos momentos de lucha constante y felicidad, aquellos que me acompañaron en mi carrera profesional porque todos han aportado durante mi formación académica.

Sin más que decir, son muchas las personas que formaron parte de mi vida profesional a las que agradezco su amistad, consejos, apoyo, ánimo y compañía en los momentos más difíciles de mi vida.

RESUMEN

La presente investigación fue elaborada con el fin de evaluar la calidad interna de un producto de software y cada uno de sus respectivos componentes y así brindar las debidas recomendaciones a servicios o productos de calidad para una empresa u organización; en este caso se analizó y evaluó la mantenibilidad del producto de software viáticos del Gobierno Autónomo Descentralizado de la Provincia de Esmeraldas, todo esto basándose en la norma ISO(International Organization for Standardization) e IEC (International Electrotechnical Commission) 25000.

La importancia de evaluar la mantenibilidad del software es porque los desarrolladores no proporcionan el valor que se requiere de obtener un software y mantenerlo. Existen varios factores que obstaculizan la mantenibilidad de un producto de software, esto conlleva que haya la dificultad de que un módulo del software no pueda adaptarse a los cambios de entorno, mejorar su funcionamiento y presente problemas al ser modificado

Por último, se evidenció qué tan importante es la calidad de un producto de software teniendo como base los resultados obtenidos de la evaluación del sistema Viáticos del Gobierno Autónomo Descentralizado de la Provincia de Esmeraldas, que permitió formular criterios de acuerdo al comportamiento del sistema en un estado de producción, permitiendo ser un apoyo para la empresa.

Palabras Clave: *Calidad, Evaluación, Estándar, Mantenibilidad, ISO/IEC 25000, Software, Tecnología*

ABSTRACT

The present investigation was elaborated with the purpose of evaluating the internal quality of a software product and each one of its respective components and thus to offer the due recommendations to services or products of quality for a company or organization; in this case the maintainability of the software product was analyzed and evaluated by the Decentralized Autonomous Government of the Province of Esmeraldas, all this basing on the norm ISO (International Organization for Standardization) and IEC (International Electrotechnical Commission) 25000.

The importance of evaluating software maintainability is because developers do not provide the value required to obtain software and maintain it. There are several factors that hinder the maintainability of a software product, this leads to the difficulty that a software module cannot adapt to changes in the environment, improve its performance and present problems when modified.

Finally, it was evident how important is the quality of a software product based on the results obtained from the evaluation of the Viáticos system of the Decentralized Autonomous Government of the Province of Esmeraldas, which allowed to formulate criteria according to the behavior of the system in a state of production, allowing to be a support for the company.

Keywords: *Quality, Testing, Standard, Maintainability, ISO/IEC 25000, Software, Technology*

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

PRESENTACIÓN DEL TEMA DE INVESTIGACIÓN

El valor del desarrollo de este documento de investigación es exponer detalladamente la importancia que tiene evaluar un producto de software y así brindar servicios o productos de calidad para una empresa u organización; lo que se busca es analizar y evaluar el producto de software viáticos del Gobierno Autónomo Descentralizado de la Provincia de Esmeraldas, basándose en la norma ISO (International Organization for Standardization) e IEC (International Electrotechnical Commission) 25000. Para mejor entendimiento del tema de investigación se detallaron temas como: la calidad de software, calidad del producto de software, métricas de calidad de software, estándares de calidad de software, de manera que está enfocado a la evaluación de la calidad de un sistema. Evaluar un producto de software mediante la implementación de la norma ISO/IEC 25000 corre con grados de complejidad, puesto que cumplir este tipo de trabajo forja realizar una buena inversión para la empresa u organización y por ello es necesario efectuarlo de una manera adecuada. Por lo tanto, la finalidad del desarrollo de esta investigación fue evaluar el sistema viáticos del Gobierno Autónomo Descentralizado de la Provincia de Esmeraldas mediante la aplicación de buenas prácticas que propone el modelo ISO 25000 para mejorar la calidad del producto.

Basándose en la metodología escogida se aplicaron las métricas de calidad interna al proceso de evaluación del sistema viáticos del Gobierno Autónomo Descentralizado de la Provincia de Esmeraldas, por consiguiente, se analizaron los resultados y se logró brindar las respectivas recomendaciones para mejorar la calidad del producto de software.

PLANTEAMIENTO DEL PROBLEMA

Cumplir con un nivel alto de calidad y validez en los servicios o productos de una empresa es un factor muy importante, tanto para el bienestar de la organización como la satisfacción de las necesidades del cliente. Si los consumidores finales se sienten satisfechos con el producto o servicio brindado, seguirán optando por el mismo proveedor y no recurrirán a otra empresa.

El ámbito de los negocios ha estado en constante desarrollo y el nivel de calidad a su vez ha ido incrementado, para que hoy una empresa sea exitosa y competitiva debe ofrecer productos o servicios de alta calidad que ofrezcan valor al cliente. Los clientes buscan calidad en los productos y que cumplan con las expectativas, puesto que, si el producto no cumple con las expectativas y no beneficia al cliente, el poseer una buena calidad tampoco podrá beneficiarlo por muy bueno que sea el producto. Las características de un producto que desea adquirir el cliente es de rapidez, confiabilidad, eficiencia, garantía, confidencialidad.

Uno de los problemas del software es la inadecuada gestión de las restricciones de los proyectos de software y la deficiencia en la evaluación de la calidad del producto software, es decir que se prioriza tiempos, alcance y costos de desarrollo del servicio en vez de la calidad del mismo [1].

Para prevenir los errores en el producto de software se debe basar en el uso correcto de metodologías ya que previo a la elección de ésta se debe realizar un análisis de los requisitos del cliente.

La evaluación de un producto de software se basa en diferentes modelos y métricas de calidad que permite la especificación de requisitos de calidad y visión profunda de la eficacia del proceso. Para ello se debe emplear un modelo de calidad de software de acuerdo a los parámetros a evaluar.

Por consiguiente, lo que se planteó del trabajo de investigación fue evidenciar qué tan importante es la calidad de un producto de software con el objetivo de evaluar el sistema, analizar los parámetros y expectativas del consumidor final y así plantear recomendaciones para la empresa.

JUSTIFICACIÓN

La evaluación de un producto de software es fundamental para asegurar la calidad en los servicios, permitiendo durante la evaluación identificar los errores y evitar que estos aparezcan una vez entregado al consumidor final.

Al desarrollar un producto se desea que sea encaminado a la calidad y eficiencia, buscando la satisfacción de las necesidades del cliente, de manera que es la persona quien califica y adquiere los beneficios de ese servicio.

Una de las actualizaciones de los estándares internacionales para evaluar la calidad de los productos de software es la familia ISO/IEC 25000; no obstante, no son muy empleados por la falta de estudio teórico actual sobre ello y conlleva a que las empresas opten por desarrollar sus propias metodologías basados en la necesidad.

Es necesario tener presente en una empresa el entorno de la evaluación de un producto de software mediante un estándar de calidad de software adecuado para lograr la satisfacción del cliente que es quien revisa detenidamente el producto. Por consiguiente, la calidad se considera como un factor importante que debe ser tratado al inicio y durante el ciclo de vida del sistema de software; además beneficia a la empresa ya que de manera interna crea su propio sistema para el perfeccionamiento del oficio. También beneficia a los trabajadores de la empresa ya que hacen uso del sistema de manera cotidiana.

La evaluación del sistema viáticos del Gobierno Autónomo Descentralizado de la Provincia de Esmeraldas beneficia al departamento de TICS de la institución, determinando un análisis de producto de calidad, en menor tiempo y con menor costo, contribuyendo no solo al GADPE sino también al cliente (personal administrativo, etc.), que es la persona que desea obtener lo que requiere aspirando a obtener un servicio o producto fiable y de calidad en un tiempo futuro.

OBJETIVOS

OBJETIVO GENERAL:

Evaluar el sistema viáticos del Gobierno Autónomo Descentralizado de la provincia de Esmeraldas (GADPE) mediante la aplicación de las métricas de la característica Mantenibilidad de la norma ISO/IEC 25000 para mejorar la calidad del producto.

OBJETIVOS ESPECÍFICOS:

- Analizar el modelo de calidad de software ISO/IEC 25000 a utilizarse.
- Aplicar las métricas de la característica Mantenibilidad al sistema informático.
- Identificar los fallos en el sistema basado en las métricas de calidad seleccionadas.

CAPITULO I: MARCO TEÓRICO

1.1. ANTECEDENTES

Una investigación remarca la importancia teórica y el aporte que realizan los especialistas, aquellas contribuciones consecuentemente reflejarán los avances y estados del conocimiento sobre la evaluación de calidad del software.

En la investigación [2] plantean que "El control de calidad no significa alcanzar la perfección. Significa conseguir una eficiente producción con la calidad que espera obtener en el mercado".

En una organización es de suma importancia ofrecer servicios y productos eficientes para llegar a satisfacer a los usuarios, por ello, efectuar un modelo de calidad brinda la mejora y reconocimiento de un producto en el mercado disminuyendo el riesgo de ser una mala organización e ir a la quiebra en el mercado.

Los estándares de calidad del software de la Organización Internacional de Normalización (ISO) han ido mejorando con el pasar del tiempo, la ISO/IEC 15598, tenía problemas en la ejecución y proceso de evaluación debido a la falta de interpretación y definición de los objetivos y relaciones entre actividades; ante esa problemática se propuso un proceso llamado "k-process", que consiste en desarrollar el estándar a través de un procedimiento mejorado y de directrices o normas adicionales [3].

La investigación puntualiza que para obtener una excelente evaluación de producto no solo se debe basar en las medidas de calidad del producto para mejorar, también debe consistir en el proceso de evaluación de calidad del producto, por ello uno de los pilares es la ISO/IEC 25000, donde el tema de calidad es más amplio y al realizarse de manera correcta conlleva a obtener un producto con mejoras en la usabilidad, seguridad y sobre todo brindará satisfacción al cliente.

Además, es importante implementar metodologías de desarrollo seguro que sean aplicadas en cada fase del ciclo de vida del software: análisis, diseño, desarrollo y pruebas [1]. Se considera importante evaluar la mantenibilidad desde etapas tempranas del ciclo de vida del software, porque ayudaría no solo a generar un software de calidad, sino que facilitaría la

capacidad de que el software permita ser modificado de manera eficiente debido a necesidades ya sean estas evolutivas y correctivas.

Además, la evaluación de la calidad de un desarrollo de software no se debe limitar exclusivamente al producto final, sino que debe incluir la evaluación de la calidad de los productos intermedios como parte de la determinación del grado de satisfacción que tiene el usuario final, y quienes quieran seguir desarrollando nuevas versiones, por consiguiente es útil evaluar los productos que se generan a lo largo del proceso de desarrollo [4].

De acuerdo a las citas de autores investigativos es de suma importancia resaltar que cada modelo basado en la evaluación de calidad de un producto de software se da de cualidades diferentes, es decir que son productos con requerimientos y necesidades diferentes, por ello no existe un solo modelo de estándar que pueda aplicar a cada paso de proceso de evaluación al momento de desarrollar un producto.

1.2. BASES TEÓRICAS CIENTÍFICAS

En la actualidad, una empresa posee áreas de tecnologías de la información donde desempañan diferentes tareas para los servicios, productos de software, gestión de procesos, que permitirá a la empresa obtener una mayor productividad y ganancia. Por ello de acuerdo con las necesidades, día a día las diferentes entidades tecnológicas proponen estándares, metodologías, normas de calidad para mejorar, evaluar y brindar un punto de apoyo al desarrollo de un producto de software para determinar su nivel de calidad e impulsar un ambiente de calidad.

En consecuencia, es importante conocer la definición precisa de lo que es un producto de software y la calidad de software. Por lo tanto, un producto es lo que diseñan y construyen los ingenieros del software, abarcando los programas, documentos que componen formularios virtuales e impresos y datos que combinan números y textos, incluyendo representaciones de información en audio, video e imágenes [5]. Se puede definir como un conjunto de procesos, datos informáticos, programas que son utilizados a los usuarios.

Si se habla de la calidad de un producto, se explica que solo es un producto de calidad, si llega a cumplir con las características o requerimientos pedidos y que lleguen a satisfacer al cliente, a quien va dirigido o persona que lo adquiera. Además, la calidad de software es un proceso eficaz que se aplica de manera útil proporcionando un valor medible a quienes lo producen y a quienes lo utilizan [6].

Por lo tanto, Pressman, recalca tres características fundamentales que se deben cumplir para llegar a obtener un software de calidad:

Un proceso eficaz de software establece la infraestructura que da apoyo a cualquier esfuerzo de elaboración de un producto de software de alta calidad [6]. Para evitar errores y pérdidas de recursos se debe analizar y comprender cada proceso del software, eligiendo la metodología adecuada de desarrollo.

Un producto útil entrega contenido, funciones y características que el usuario final desea; sin embargo, de igual importancia es que entrega estos activos en forma confiable y libre de errores. Un producto útil siempre satisface los requerimientos establecidos en forma explícita por los participantes [6]. En esta deducción es posible que aquellas características implícitas

que un software requiere son necesarias y con los requerimientos o especificaciones del cliente harán posible que el resultado final sea de buena calidad; puesto que sirve de un complemento importante para el desarrollo del mismo.

Al agregar valor para el desarrollador y para el usuario de un producto, el software de alta calidad proporciona beneficios a la organización que lo produce y a la comunidad de usuarios finales [6]. Un software bien desarrollado y con las debidas evaluaciones realizadas tiene menos probabilidades de fallar o producir errores, esto permite que a futuro haya una reducción de costos evitando repetir el producto.

Asimismo, la calidad de software es el grado con el que un sistema, componente o proceso cumple los requerimientos especificados y las necesidades o expectativas del cliente o usuario [2]. Dentro de las organizaciones la calidad de software es muy importante, debido a que afecta la calidad del producto ofrecido, en la Figura 1 se determina la administración de la calidad de un producto.

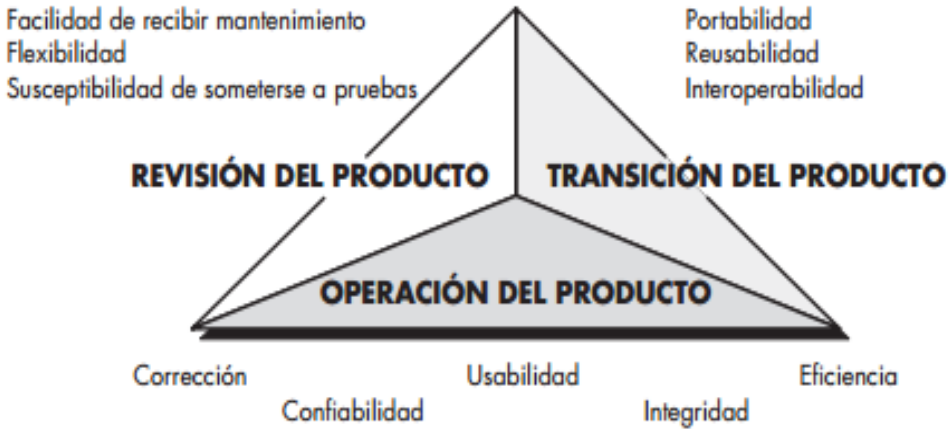


Figura 1. *Administración de la Calidad* [5].

Hay que tener en cuenta que la calidad de un producto es el grado en que el producto software satisface las necesidades expresadas o implícitas, cuando se usa bajo condiciones determinadas [7].

Existen diferentes enfoques de calidad de un producto de software como son: interna, externa y la calidad del software en el uso (Figura 2).

Figura 2. *Calidad de un Producto*

- Calidad Interna del software: Puede ser medida y evaluada desde características y cualidades propias del producto de software como por ejemplo el código. Por lo tanto, calidad interna es la totalidad de atributos de un producto que determina su capacidad de satisfacer necesidades explícitas e implícitas cuando es usadas bajo condiciones específicas [8].
- Calidad Externa del Software: Puede ser medida y evaluada desde características dinámicas, es decir desde el comportamiento del producto de software como por ejemplo durante su ejecución en tiempo real desde un computador personal. Por ello la calidad externa es el grado en que un producto satisface necesidades explícitas e implícitas cuando se utiliza bajo condiciones especificadas [8] .
- Calidad en el Uso: Esta puede ser medida y evaluada desde el punto de vista de los usuarios, es decir, es evaluada de acuerdo con los resultados obtenidos mediante la

utilización del producto de software. Además hay que tener en cuenta que la calidad en uso es la capacidad de un producto de software de facilitar a usuarios específicos alcanzar metas específicas con eficacia, productividad, seguridad y satisfacción en un contexto específico de uso [8].

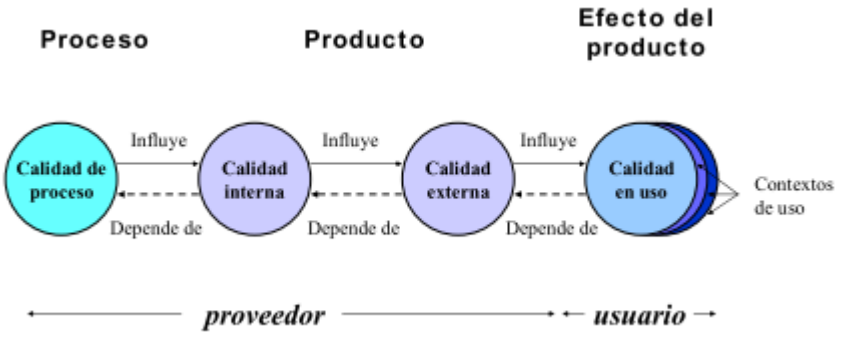


Figura 3. *Ciclo de Vida del Software Respecto a la Calidad* [9].

En la Figura 3, se observa el ciclo de vida del software que representa las dependencias de los distintos enfoques de calidad mencionados anteriormente (interna, externa y en uso); se puede apreciar que la calidad de proceso ayuda a optimizar la calidad del producto de software sea en la calidad interna/externa del producto y a su vez contribuye a optimizar la calidad en uso. Por consiguiente, mejorar las medidas de proceso de desarrollo favorece a obtener un producto de mejor calidad y evaluar la calidad del producto de software favorece a mejorar la calidad en uso. Es decir que para lograr un producto de calidad es necesario realizar un proceso de desarrollo de calidad.

Dentro del marco de la Ingeniería de Software la frase evaluación de software cumplen un papel muy importante en el desarrollo de software, porque ayuda al proceso de desarrollo cumpliendo con los requerimientos establecidos por el cliente.

Evaluación es un proceso que tiene por objeto determinar en qué medida se han logrado los objetivos previamente establecidos que supone un juicio de valor sobre la programación establecida, y que se emite al contrastar esa información con dichos objetivos [10].

En la actualidad no solo en el campo de la tecnología deben realizarse evaluaciones o pruebas de software para verificar que un producto esté bien hecho y que cumpla las indicaciones requeridas. Particularmente en la Ingeniería del Software, el porqué de las valoraciones de

software está para que un producto de software quede completamente acabado y garantizado, y necesita ser evaluado en todos los ámbitos posibles, es decir que: hace lo que tiene que hacer, funcione correctamente y que lo haga de manera eficaz.

El proceso de evaluación de un producto de software "Es un componente crítico del ciclo de vida del desarrollo de software, cuyo objetivo es ayudar a mejorar la calidad y fiabilidad del producto mediante la identificación de defectos, la prevención de errores, y la observación y presentación de reportes" [11]. Se considera que evaluar un producto de software es la única forma precisa para determinar fallos en la programación o incluso en otra parte del proceso del software, que presenta el producto que se esté desarrollando.

"Demostrar al desarrollador y al cliente que el software cumple con los requerimientos, y encontrar situaciones donde el comportamiento del software sea incorrecto, indeseable o no esté de acuerdo con su especificación" [12]. Es muy importante crear productos de software de calidad, aplicando pruebas de calidad de software para evitar graves problemas al producto y reducir errores o fallos en los sistemas para la satisfacción del cliente.

Es posible dividir los tipos de pruebas de software en tres grupos generales: ***Pruebas Funcionales***, que están basadas en la interoperabilidad de las operaciones que realiza un determinado sistema; ***Pruebas No Funcionales***, que corresponden a la medición de las características de una aplicación que puedan ser cuantificadas mediante una determinada escala, y ***Pruebas Estructurales***, que están centradas en los procedimientos del software a partir del código fuente, analizando los flujos de cumplimiento en los procedimientos mediante diversas entradas que produzcan salidas fiscalizadas [13].

Existen ciencias, técnicas, tipos, requisitos y pasos necesarios para poder evaluar un software específico de forma correcta, además de pruebas dedicadas para cada fase del proceso del software; es decir no solo se debe centrar únicamente en el código, a pesar de que, en esta etapa, los únicos errores corregibles serán los de codificación.

El cómo evaluar un producto de software va a depender principalmente del tipo de software que se esté desarrollando. Al ver las distintas maneras posibles en las que un software podría ser probado o todas las consideraciones que se deben de tener al momento de iniciar las respectivas evaluaciones, es fácil caer en la duda que conlleva a la siguiente pregunta: ¿Es

posible evaluar un software de manera que encontremos y corrijamos todos sus errores? La respuesta a esta pregunta es un poco complicada, puesto que es imposible cuantificar todas las características que deberían ser corregidas para que un software sea totalmente evaluado, sin embargo, es importante siempre definir todas las posibles entradas y salidas de un software, o todos los pasos necesarios que culminan en la resolución de un problema.

En resumen, se puede considerar que las pruebas alcanzan objetivos diferentes: en las metodologías convencionales se utilizan principalmente como base de la verificación y la validación del producto desarrollado, mientras que en las metodologías ágiles se llegan a utilizar en sustitución de las especificaciones de requisitos y como guía para el desarrollo de software [14].

Para ello existen dos métodos para realizar una evaluación que son:

a) ***Evaluación objetiva:*** En una evaluación objetiva se requiere la medición, ya que no es suficiente que la investigación se encuentre basada en opiniones, se elabora en base a un análisis y a un diagnóstico realizado sobre el hecho o fenómeno a evaluar. Cuando se efectúa la medición es preciso asignar números para cuantificar el objeto o hecho que se está investigando.
b) ***Evaluación subjetiva:*** Es una evaluación que se encuentra basada en opiniones de personas sobre un hecho o fenómeno, pero que carece de fundamento por lo tanto no puede ser certificado.

La evaluación de sistemas no solo hace referencia a los activos informáticos, sino también a la aptitud de las personas que laboran en la organización y a la colectividad que se encuentra fuera de la organización, esto en base a sus deseos, fines, requerimientos e intenciones. La evaluación de los sistemas permite dar como resultado un diagnóstico de la organización para saber el período en el que se encuentra e identificar qué falencias posee; es importante recalcar que en una organización todas las mejoras en un sistema deben ser aprobados, evaluados y monitoreados para dar a conocer los progresos alcanzados.

La realización de una evaluación persigue alcanzar los objetivos planteados y para ello se debe seguir un proceso de pasos fundamentales según en el estudio de Blázquez [15]:

a) ***Medir la consecución de los objetivos previamente establecidos:*** Se determina si se cumple la evaluación con los factores de eficacia, eficiencia o factor de impacto de un servicio o sistema de información.
b) ***Disponer de un instrumento para diagnosticar los puntos débiles en el funcionamiento:*** El instrumento puede ser el propio sistema de información o metodología específica para la medición de factores o indicadores de actividad, calidad, servicio, producción, etc.
c) ***Facilitar el proceso de la toma de decisiones:*** Utilizando información objetiva, no basadas en opiniones o suposiciones.
d) ***Permitir la comparación entre sistemas mediante la construcción de estándares de referencia:*** Evaluar la unidad de información y documentación en función de trabajos de evaluación de otras bibliotecas, centros de documentación o archivos. Por lo tanto, se fijan unos indicadores comunes por los que determinar la eficiencia y eficacia de los servicios y procesos. Éste método también ha sido denominado evaluación exógena.
e) ***Justificar la existencia de los servicios y sistemas de información:*** Evaluar para justificar el buen funcionamiento y mantenimiento de un servicio en función de su rendimiento económico, difusión o alcance, resultados operativos, calidad de servicio y respuesta al usuario. Todo ello implica como resultado la satisfacción del usuario.

Para saber sobre la eficacia de los procesos del software se requiere de la utilización de métricas de calidad. Las métricas son medidas realizadas sobre los procesos que permiten tomar decisiones al momento de mejorar un atributo mediante la comparación de indicadores o características. En la investigación [16] explican que "Una métrica es una medida cuantitativa del grado en que un sistema, componente o proceso posee un atributo dado". Por ello es muy importante establecer un estándar de medición o modelo de calidad para lograr proporcionar cantidades o valores cuantitativos a las métricas de calidad.

La clasificación de las métricas señalada por Luque Javier [17] es explicada de la siguiente manera:

a) ***Métricas de Proceso***: estas se obtienen especificando los atributos por proyecto, es decir se desarrolla un conjunto de métricas significativas sobre la información que se dispone de estos atributos, posteriormente se obtendrían los indicadores que permitirían obtener una estrategia donde mejoraría el proceso para futuras situaciones.
b) ***Métricas de Proyecto***: son las más importantes a nivel de conjunto de proyecto, sobre todo a la hora de la planificación y a la hora de proporcionar la estrategia que se tomará para llevar a cabo el proyecto, son utilizadas por parte del director de proyecto y su equipo para controlar de una manera global la situación y transcurso del proyecto.
c) ***Métricas de Software****:* su objetivo es la valoración cuantitativa de la calidad del software. Las mediciones realizadas permiten confirmar el correcto funcionamiento del software al evaluar la calidad de análisis, modelos de diseños y código fuente del mismo.

Las métricas de software abarcan un amplio contexto de actividades para suministrar información relevante a tiempo y así mejorar tanto los procesos como los productos, así como lo explica el estudio [18]:

a) Aseguramiento y Control de calidad
b) Modelos de fiabilidad
c) Modelos y evaluación de ejecución
d) Modelos y medidas de productividad

Para la recopilación de métricas de software se debe realizar el siguiente proceso (Figura 4).

Figura 4. *Proceso de Recopilación de Métricas de Software.*

Las métricas se clasifican de la siguiente manera según cada criterio de complejidad, calidad, competencia, desempeño y estilizadas [18]. (Tabla 1):

Complejidad	Métrica que definen la medición de la complejidad: volumen, tamaño, anidaciones y configuración.
Calidad	Métricas que definen la calidad del software: exactitud, estructuración o modularidad, pruebas, mantenimiento.
Competencia	Métricas que intentan valorar o medir las actividades de productividad de los programadores con respecto a su certeza, rapidez, eficiencia y competencia.
Desempeño	Métricas que miden la conducta de módulos y sistemas de un software, bajo la supervisión del SO o hardware.
Estilizadas	Métricas de experimentación y de preferencia: estilo de código, convenciones, limitaciones, etc.

Tabla 1. *Clasificación de las Métricas de Software* [18].

Además, Chisaguano Balseca recalca que las métricas de software deben ser detalladas con el objetivo, identificando las características de calidad del producto software más relevantes que se analizarán y se ejecutarán en la evaluación [19]. Para ello se debe utilizar una tabla de métricas descrito por la ISO/IEC 25020, puntualizados en la Tabla 2.

TABLA DE MÉTRICAS	
Ítem	**Descripción**
Subcaracterística	Subcaracterística de calidad.
Nombre de la métrica	Nombre asignado a la métrica de calidad.
Fase del ciclo de vida de calidad del producto	Fase del ciclo de vida: calidad interna, calidad externa y calidad en uso
Método de aplicación	Manera de cómo se va a aplicar la métrica.
Formula y cálculo de datos	Establece la fórmula de medición y especifica los significados de los datos que se van a utilizar.
Valor deseado	Proporciona el rango y los valores preferibles y recomendados.
Tipo de medida	Especifica el tipo de medida que se va seleccionar, como: tamaño (tamaño de la función, tamaño de la fuente), tiempo (lapso de tiempo, tiempo de usuario), contar (número de cambios, números de fallas).
Recursos utilizados	Específica los recursos que se utilizarán para poder medir cada métrica, entre los recursos utilizados pueden estar: entrevistas a usuarios, código fuente, documentación, entre otras.

Tabla 2. *Puntualización de la Tabla de Métricas* [19].

La ISO/IEC 25020 también puntualiza las métricas para medir la calidad interna/externa y calidad en uso, puntualizados en la Tabla 3 y Tabla 4.

CALIDAD INTERNA/EXTERNA – MANTENIBILIDAD MÉTRICAS		
Característica	**Subcaracteristicas**	**Métricas**
Mantenibilidad	Modularidad	✓ Capacidad de condensación ✓ Acoplamiento de clases.
	Reusabilidad	✓ Ejecución de reusabilidad.
	Capacidad de ser Analizado	✓ Capacidad de pistas de auditoría. ✓ Diagnóstico de funciones suficientes.
	Capacidad de ser modificado	✓ Complejidad ciclomática ✓ Grado de localización de corrección de impacto
	Capacidad de ser probado	✓ Capacidad de reinicio de pruebas ✓ Capacidad de prueba autónoma

Tabla 3. *Métricas de la Calidad Interna/Externa-Mantenibilidad* [19].

CALIDAD EN USO - MÉTRICAS		
Características	**Subcaracteristicas**	**Métricas**
Efectividad	Efectividad	✓ Completitud de la tarea. ✓ Efectividad de la tarea. ✓ Frecuencia de error.
Eficiencia	Eficiencia	✓ Tiempo de la tarea. ✓ Tiempo relativo de la tarea. ✓ Eficiencia de la tarea. ✓ Eficiencia relativa de la tarea. ✓ Porcentaje productivo. ✓ Numero relativo de las acciones del usuario.
Satisfacción	Utilidad	✓ Nivel de satisfacción. ✓ Uso discrecional de las funciones. ✓ Porcentaje de quejas de los clientes
Libertad de riesgo	Libertad del riesgo económico	✓ Retorno de la Inversión (ROI). ✓ Tiempo para lograr el retorno de la inversión. ✓ Rendimiento relativo de negocios. ✓ Tiempo de entrega. ✓ Ganancias para cada cliente. ✓ Errores con consecuencias económicas. ✓ Corrupción del software.
	Libertad del riesgo de salud y seguridad	✓ Frecuencia de problemas en la salud y seguridad del usuario. ✓ Impacto en la salud y seguridad del usuario. ✓ Seguridad de las personas afectadas por el uso del sistema.
	Libertad del riesgo ambiental	✓ Impacto Ambiental.
Cobertura de Contexto	Completitud de Contexto y Flexibilidad	✓ Completitud de contexto ✓ Función flexible del diseño

Tabla 4. *Calidad en Uso – Métricas* [19].

Para evaluar la característica mantenibilidad de la calidad interna/externa del producto de software se necesita aplicar las siguientes métricas de la ISO/IEC 25022 puntualizadas en la Tabla 5.

MÉTRICAS PARA LA CARACTERÍSTICA DE LA CALIDAD INTERNA/EXTERNA MANTENIBILIDAD								
Subcaracterística	**Métrica**	**Fase del ciclo de vida de calidad del producto**	**Propósito de la métrica de calidad**	**Método de aplicación**	**Fórmula**	**Valor deseado**	**Tipo de medida**	**Recursos utilizados**
Modularidad	Capacidad de condensación	Interna	¿Qué tan fuerte es la relación entre los componentes del sistema?	Contar el número de componentes que no son afectados por cambios de otros componentes y el número total de componentes específicos	x = B/A A = Número de componentes que no son afectados por cambios de otros componentes B = Número total de componentes específicos Dónde: B > 0	0<=X<=1 El más cercano a 0, es el mejor	X=Contable/ Contable A=Contable B=Contable	Código fuente
	Acoplamiento de clases	Interna	¿Qué tan fuerte es la relación entre una función del sistema con otras clases implementadas?	Contar el número de relaciones que tiene una función con respecto a otras clases.	X = A A = Número de relaciones que tiene una función con respecto a otras clases	1<=X<=5 El más cercano a 1, es el mejor	X=Contable A=Contable	Código fuente
Reusabilidad	Deuda Técnica	Interna	¿Cuántos elementos pueden ser reutilizados?	Contar el número de elementos reutilizados y el número total de elementos de la biblioteca reutilizable	X = A / B A = Número de elementos reutilizados B = Número total de elementos de la biblioteca reutilizable	0<=X<=1 El más cercano a 1, es el mejor	X=Contable/ Contable A=Contable B=Contable	Código fuente

					Dónde: B > 0			
Capacidad de ser analizado	Capacidad de pistas de auditoría	Interna/Externa	¿Los usuarios pueden identificar fácilmente la operación específica que causó el fallo?	Contar el número de datos realmente grabadas durante la operación y el número de datos previstos a grabarse para controlar el estado del sistema durante la operación	X = A / B A = Número de datos realmente grabadas durante la operación B = Número de datos previstos a grabarse para controlar el estado del sistema durante la operación Dónde: B > 0	0<=X<=1 El más cercano a 1, es el mejor	X=Contable/ Contable A=Contable B=Contable	Especificación de requerimientos, Código fuente, Desarrollador, Tester
	Diagnóstico de funciones suficientes	Interna/Externa	¿Hasta qué punto las funciones de diagnóstico están preparadas o hasta qué punto funcionan para el análisis causal?	Contar el número de funciones de diagnóstico implementadas y contar el número de funciones de diagnóstico requeridas en la especificación de requerimientos	X = A/B A = Número de funciones de diagnóstico implementadas B = Número de funciones de diagnóstico requeridas en la especificación de requerimientos Dónde: B > 0	0<=X<=1 El más cercano a 1, es el mejor	X=Contable/ Contable A=Contable B=Contable	Especificación de requerimientos, Código fuente, Desarrollador, Tester
Capacidad de ser modificado	Complejidad ciclomática	Interna	¿Cuál es la complejidad estructural de un código fuente?	Contar las instrucciones condicionales, bucles, salidas de métodos y clausulas AND	X = A+1 A = Número de instrucciones condicionales que tiene una función	1<=X<50 El más cercano a 1, es el mejor	X=Contable A=Contable	Código fuente

				y OR dentro de los condicionales.				
	Grado de localización de corrección de impacto	Interna/Externa	¿Hasta qué punto los problemas causados pueden tener como consecuencia un mantenimiento ?	Contar el número de fallas aparecidas después que se ha resuelto un fallo y contar el número de fallas resueltas	X = A/B A = Número de fallas aparecidas después que se ha resuelto un fallo B = Número de fallas resueltas Dónde: B > 0	0<=X<=1 El más cercano a 0, es el mejor	X=Contable/Contable A=Contable B = Contable	Especificación de requerimientos, Código fuente, Desarrollador, Tester
Capacidad de ser probado	Capacidad de prueba autónoma	Interna	¿Qué tan independiente es el software al ser probado?	Contar el número de pruebas que están dependiendo de otros sistemas y contar el número total de pruebas dependientes con otros sistemas	X = A/B A = Número de pruebas que están dependiendo de otros sistemas B = Número total de pruebas dependientes con otros sistemas Dónde: B > 0	0<=X<=1 El más cercano a 0, es el mejor	X=Contable/Contable A=Contable B=Contable	Código fuente, Tester

Tabla 5. *Métricas de la Característica Mantenibilidad - ISO/IEC 25022* [19].

Existen diferentes modelos y estándares de calidad de software que sirven tanto para evaluar la calidad de los productos de software, además se encargan de describir las características de los productos a través de mediciones que posteriormente serán interpretadas. La implementación de estos estándares en una empresa ayuda a la planificación de buenos proyectos, a la toma de decisiones, reducción de riesgos, y mejora continua de la tecnología.

Existen varias normativas a nivel del producto de software (Figura 5) como McCall, Boehm, ISO, SQAE o Software Quality Assessment Exercise, pero a medida que se han mejorado los métodos tecnológicos se han dejado de implementar en las empresas. Las normativas ISO más relevantes para la calidad del producto de Software se pueden observar en la Figura 6.

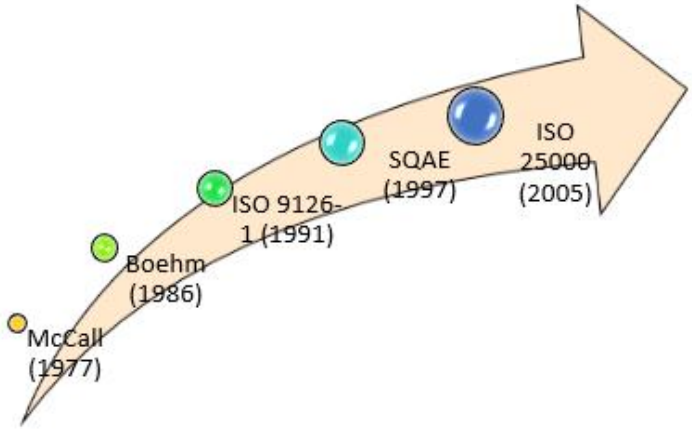


Figura 5. *Modelos de Calidad a Nivel de Producto.*

Figura 6. *Normativas de Calidad en el Producto.*

ISO/IEC 9126 es un conjunto de normas/estándares desarrollados por la ISO (International Organization for Standardization) e IEC (International Electrotechnical Commission), dedicado a la evaluación de la calidad del software. Se definen seis principales características y cada característica consta de subcaracteristicas las cuales debe cumplir todo producto de software para ser considerado de calidad, ver Figura 7. En el estudio [20], se define a un modelo de calidad como "El conjunto de características y las relaciones entre las mismas, que proveen la base para especificar requerimientos de calidad y evaluar calidad"; es decir que la calidad no solamente está ligado al producto, sino que también va de la mano con el cliente reuniendo las cualidades de todos los bienes y servicios relacionados entre sí.

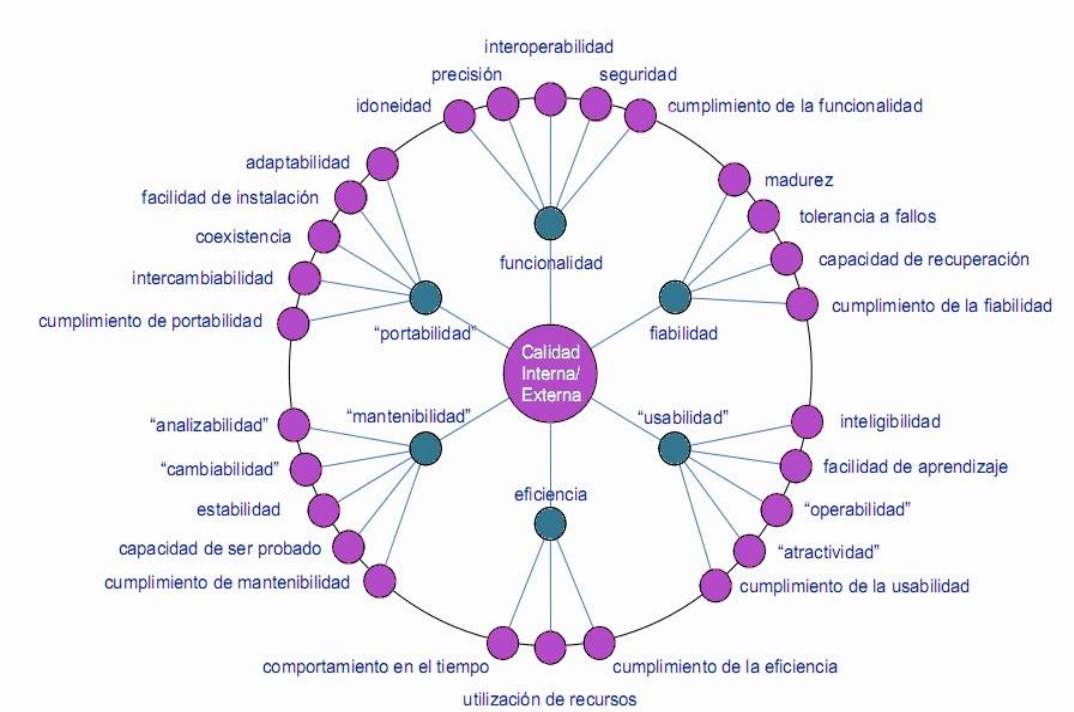


Figura 7*. Principales Características de la Calidad Interna/Externa del Software ISO 9126* [21].

- **Funcionalidad**

La funcionalidad de un software debe estar orientada a cumplir los objetivos para los cuales fue programado, es decir que cada una de sus funciones o características estén correctamente definidas y cumplan con su propósito específico.

La funcionalidad está expresada con las siguientes características:

a) ***Idoneidad:*** Sus funcionalidades son exactamente las necesarias y óptimas.

b) ***Precisión:*** Datos exactos, sin ambigüedades ni errores.
c) ***Interoperabilidad:*** Manejable internamente, puede estar sujeto a cambios durante su desarrollo.
d) ***Seguridad:*** el nivel de seguridad debe ser el adecuado según los requisitos.

- **Fiabilidad**

Se refiere al nivel de confianza que el software tenga a partir de los resultados obtenidos mediante las pruebas y bajo las condiciones propuestas, es decir que funcione correctamente bajo cualquier circunstancia que se plantee. Está expresada con las siguientes características:

a) ***Madurez:*** debe ser lo suficientemente óptimo.
b) ***Tolerancia a fallos:*** debe ser capaz de lidiar con ciertos fallos predeterminados internos o de usuarios.
c) ***Capacidad de recuperación:*** además de tolerar fallos debe responder a estos.

- **Usabilidad**

Significa que el software debe ser lo suficientemente entendible y fácil de usar para que los usuarios a los cuáles va dirigido se sientan cómodos y no tengan dificultades relacionadas con el uso correcto del producto. Está basada en las siguientes características:

a) ***Inteligibilidad:*** debe ser lo suficientemente sencillo y completo para ser entendido por los usuarios.
b) ***Facilidad de aprendizaje:*** el software debe ser lo suficientemente sencillo para los usuarios a los que va dedicado.
c) ***Operabilidad:*** que el manejo de las funciones sea intuitivo y fácil.
d) ***Atractividad:*** su forma de uso debe llamar la atención a quienes va dirigido.

- **Eficiencia**

El software debe presentar niveles estables de rendimiento y consumo de memoria tanto en su uso mínimo, como prolongado, sin errores que impidan que el usuario deje de utilizarlo, como son:

a) ***Utilización de recursos:*** no debe sobrepasar los recursos especificados y funcionar sin problemas en equipos no tan actuales.
b) ***Comportamiento en el tiempo:*** por ningún motivo el software debería bajar su rendimiento o comportarse de manera distinta con el uso continuo.

- **Mantenibilidad**

El software junto con sus características y funcionalidades deben brindar la suficiente documentación y facilidad de código para que sea posible actualizar sus funciones de forma sencilla sin necesidad de mucho esfuerzo. Está expresada en las siguientes características:

a) ***Capacidad de ser probado:*** en todas sus versiones debe cumplir con las características suficientes para que se puedan seguir realizando las pruebas sobre este.
b) ***Estabilidad:*** sus características principales y para las cuales fue creado deben mantenerse con las mejoras.
c) ***Cambiabilidad:*** sus funcionalidades deben poder ser cambiables de ser necesario.
d) ***Analizabilidad:*** debe ser lo suficientemente entendible como para ser analizado a fondo.

- **Portabilidad**

Se refiere a la capacidad del software de funcionar en diferentes plataformas con todas sus características deseadas sin error alguno. Está expresado en las siguientes características*:*

a) ***Intercambiabilidad:*** sus datos de entrada, salida y guardado deben funcionar adecuadamente en cualquier sistema soportado.
b) ***Coexistencia:*** su instalación y uso no debe interferir con ningún otro software o hardware del equipo en el cuál es instalado.
c) ***Facilidad de instalación:*** su instalación debe ser intuitiva y fácil de completar en un tiempo adecuado.
d) ***Adaptabilidad:*** debe poder adaptarse al entorno del sistema en el cuál es instalado.

En la Figura 8 se puede observar de manera detallada las características principales de la calidad del software, especificando sus subcaracteristicas y propiedades.

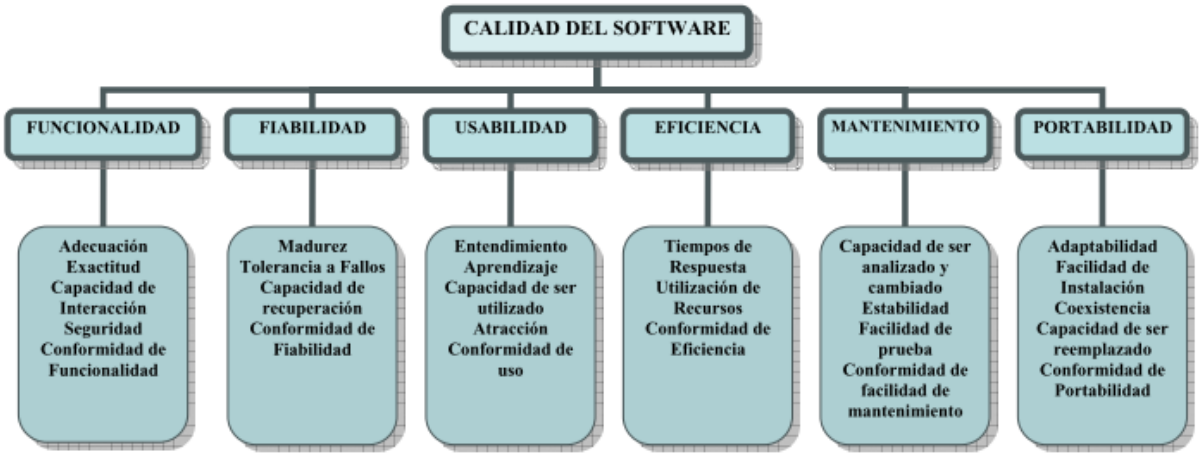


Figura 8. *Propiedades de la Calidad del Software según la ISO/IEC 9126* [22].

Estas propiedades son definidas por la ISO/IEC 9126, la investigación [22] explica cómo "El aseguramiento de la calidad del software conformará el conjunto de actividades planificadas y sistemáticas necesarias para aportar la confianza en que el producto cumple con los requisitos de calidad establecidos. El proceso de aseguramiento de calidad debe garantizar no sólo los aspectos que se refieren a cuestiones funcionales, sino también los que se refieren al resto de propiedades definidas (eficiencia, usabilidad, fiabilidad, portabilidad, etc.)".

En relación al modelo de calidad del producto software, el estándar ISO/IEC 9126 [8] afirma que está dividido en cuatro partes:

1. ***ISO/IEC 9126-1:*** Presenta un modelo de calidad del software, estructurado en características y subcaracteristicas.
2. ***ISO/IEC TR 9126-2:*** Proporciona métricas externas para medir los atributos de seis características de calidad externa y una explicación de cómo aplicar las métricas de calidad de software.
3. ***ISO/IEC TR 9126-3:*** Proporciona métricas internas para medir atributos de seis características de calidad interna.
4. ***ISO/IEC TR 9126-4:*** Define métricas de calidad en uso para medir los atributos.

El estándar ISO/IEC 14598 [23] es un conjunto de normas basado específicamente para evaluar un producto de software. Para evaluar la calidad de los productos de software esta

norma establece un equipo de trabajo y proporciona métodos, métricas para el proceso de evaluación de los mismos.

La norma define las principales características del proceso de evaluación (Figura 9):

Figura 9. *Características del Proceso de Evaluación - ISO/IEC 14598*

Para que las características establecidas por la ISO/14598 posean un buen funcionamiento, es necesario aplicar las medidas concretas definidas en la Figura 10:

Figura 10. *Medidas Concretas - ISO/IEC 14598*

La Norma ISO/IEC 14598 [23] afirma que para evaluar un producto de software realiza un proceso que consta de seis segmentos:

- **ISO/IEC 14598-1** Enfoque General: Brinda una perspectiva general de las otras cinco partes y manifiesta la concordancia entre la valoración del producto software y el estándar de calidad definido en la ISO/IEC 9126.

- **ISO/IEC 14598-2** Proyección y Gestión: Posee los requerimientos y pautas para las funciones de soporte tales como la organización y gestión de la evaluación del producto del software.
- **ISO/IEC 14598-3** Proceso para el desenvolvedor: Suministra los requerimientos y pautas para la evaluación del producto software en el momento de que la evaluación es llevada a cabo en relación con el desarrollo.
- **ISO/IEC 14598-4** Proceso para compradores: Suministra los requerimientos y pautas para que la evaluación del producto software sea llevada a cabo en función a los clientes que desean adquirir un producto de software efectivo o pre-desarrollado.
- **ISO/IEC 14598-5** Proceso para fiadores: La evaluación es llevada a cabo por evaluadores independientes y se suministran los requerimientos y pautas para la evaluación del producto software.
- **ISO/IEC 14598-6** Documentación de Módulos: provee las guías para la documentación del módulo de evaluación.

El proceso de evaluación radica en el seguimiento de un conjunto de pasos cuyo objetivo principal es evaluar la calidad del producto de software. En la Figura 11 se muestra el proceso de evaluación de un producto de software basada en la ISO/IEC 14598.

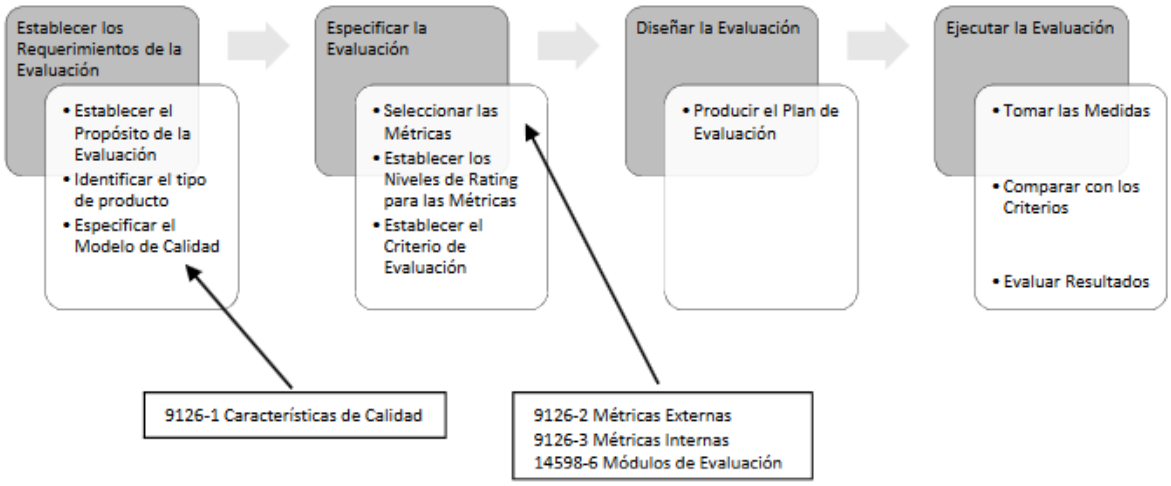


Figura 11. *Proceso de Evaluación de la Calidad del Producto de Software - ISO/IEC 14598* [24].

Hay que tener en cuenta que en la investigación [24] se explicó que " El proceso de evaluación se ve influenciado por su contexto (ambiente) y requiere recursos (personal, herramientas, tiempo, etc.), estos dos aspectos son gestionados por actividades de soporte a la evaluación". Todo el proceso de evaluación del producto de software es tratado por la norma ISO/IEC 14598. La Figura 12, planteada por la norma ISO/IEC 9126 permite analizar la concordancia entre las normas ISO/IEC 9126 que se encarga específicamente de la calidad del producto de software e ISO/IEC 14598 que se encarga del proceso de la evaluación del producto de software.

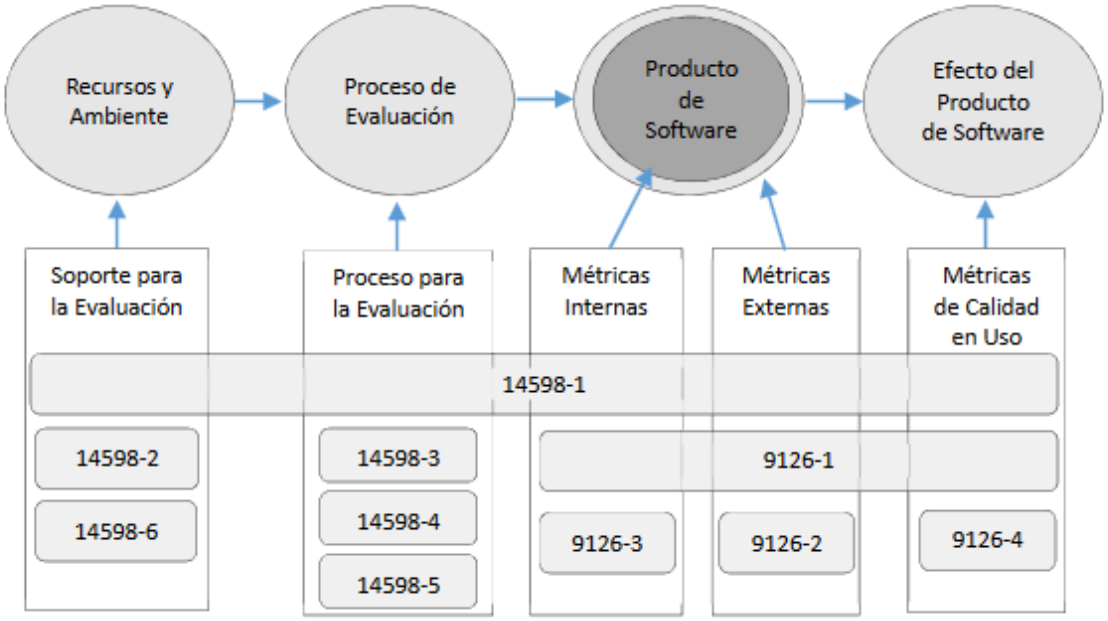


Figura 12. *Analogía entre las Normas ISO/IEC 9126 e ISO/IEC 14598* [24].

El estándar ISO/IEC 25000 [25] es una familia de normas que tiene por objetivo la creación de un marco de trabajo común para evaluar la calidad del producto software. La serie ISO/IEC 25000 es el resultado de la evolución de otras normas anteriores, especialmente de la serie de normas ISO/IEC 9126, que describe las particularidades de un modelo de calidad del producto software, y la serie ISO/IEC 14598, que abordaba el proceso de evaluación de productos software. Esta familia de normas ISO/IEC 25000 se encuentra compuesta por cinco divisiones (Figura 13).

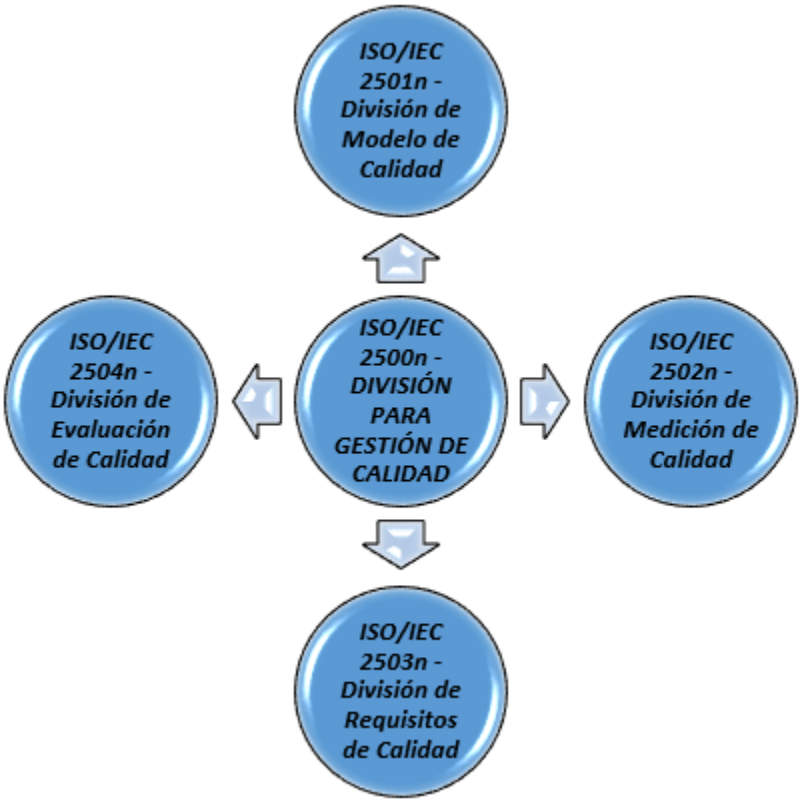


Figura 13. *Familia ISO/IEC 25000.*

ISO/IEC 25000 - División de Gestión de Calidad. - Las normas que forman este apartado definen todos los modelos, términos y definiciones comunes referenciados por todas las otras normas de la familia 25000. Según [25] afirma que división de la familia 25000 se encuentra formada por:

- ***ISO/IEC 25000 (2005)*** – ***Guía para SQuaRE***: Este modelo está sujeto a la arquitectura de SQuaRE, donde adquiere modelos de referencia y un resumen de las partes de la terminología de la familia.

ISO/IEC 2501n - División de Modelo de Calidad: Las pautas de este apartado están regidos en modelos de calidad conteniendo las características externas, internas y en uso de un producto de software. Según [25] afirma que presenta las siguientes subdivisiones como:

a) ISO/IEC 25010 (2010) –Modelos de Calidad de Software y Sistemas: Detalla un modelo de calidad en uso para el producto de software, presentando características y subcaracteristicas de calidad para su posterior evaluación del mismo.
b) ISO/IEC 25012 – Modelo de Calidad de Datos: Precisa un modelo para la calidad de los datos, adaptable para aquellos que son parte un sistema de información y se encuentren almacenados de manera organizada.

ISO/IEC 2502n - División de Medición de Calidad: Sostienen un modelo de la medición de la calidad del producto de software, tales como la medida de la calidad interna, externa y en uso para su respectiva aplicación. Según [25] afirma que la división se encuentra establecida por:

a) ISO/IEC 25020 - Guía y Modelo de Referencia para la medición: Provee un manual que sirve como guía para que los clientes puedan desarrollar y aplicar las medidas de calidad propuesta por las normas ISO.
b) ISO/IEC 25021 - Elementos de Medida de Calidad: Puntualiza un conjunto de métricas base que se pueden utilizar en el proceso de medida de la calidad interna, externa y en uso del sistema a lo largo del ciclo de vida del desarrollo de software. También especifica la forma de crear nuevas métricas de calidad en el modelo.
c) ISO/IEC 25022 - Medición de Calidad Interna: Precisa concretamente las métricas para efectuar la medición de la calidad interna de los productos de software.
d) ISO/IEC 25023 - Medición de la Calidad Externa: Precisa concretamente las métricas para efectuar la medición de la calidad externa de los productos de software.
e) ISO/IEC 25024 - Medición de la Calidad de los Datos y en Uso: Precisa las métricas para desarrollar la medición de la calidad de los datos y en uso del producto en función de características y subcaracteristicas, además un manual de usuario donde especifica la utilización de dichas medidas.

ISO/IEC 2503n - División de Requisitos de Calidad: Ayuda a definir requisitos de calidad en la que son utilizados como entrada para el proceso de evaluación del producto de software. Para ello según [25] afirma que se establece de la siguiente manera:

ISO/IEC 25030 (2007) – Requisitos de Calidad: Suministra un conjunto de recomendaciones para efectuar las especificaciones de los requisitos de calidad del producto software.

ISO/IEC 2504n - División de Evaluación de Calidad: Aquellas normas brindan recomendaciones, requisitos y una guía para llevar a cabo el proceso de evaluación del producto software. Según [25] afirma que está constituida por:

a) ISO/IEC 25040 - Guía y Modelo de Referencia de la Evaluación: Propone un modelo general para la evaluación, en donde se consideran las entradas al proceso de evaluación del producto de software, las restricciones y los recursos precisos para conseguir las convenientes salidas.
b) ISO/IEC 25041 - Guía de Evaluación para Desarrolladores, Adquirentes y Evaluadores Independientes: Detalla las respectivas recomendaciones y requerimientos para la implementación práctica de la evaluación del producto software desde la perspectiva de los evaluadores independiente, adquirentes y desarrolladores.
c) ISO/IEC 25042 - Módulos de Evaluación: Se considera un modelo de evaluación, documentación, estructura y contenido que se debe implementar al momento de definir estos módulos.
d) ISO/IEC 25045 - Módulo de Evaluación para Recuperabilidad: Precisa un módulo para la evaluación de la Subcaracterística Recuperabilidad.

En la Figura 14 se observa un modelo de las ocho principales características para la calidad interna y externa del producto de software dentro de la familia ISO/IEC 25000 que son: Adecuación Funcional, Eficiencia de Desempeño, Compatibilidad, Usabilidad, Fiabilidad, Seguridad, Mantenibilidad, Portabilidad y cada característica consta de subcaracteristicas.

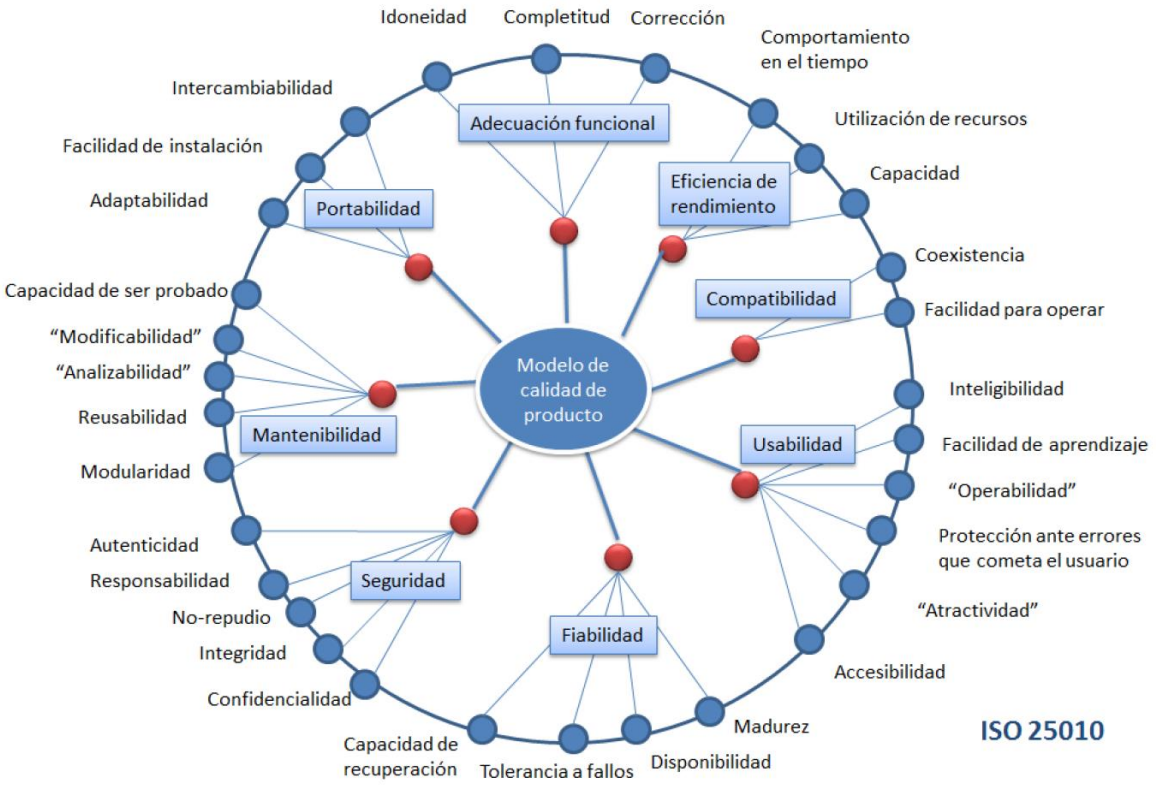


Figura 14. *ISO/IEC 25000 - Calidad del Producto Software* [26].

La norma ISO/IEC 25024 explica cómo proporciona un conjunto de medidas con las funciones de medición asociadas, por cada una de las características de calidad en el uso, ver Figura 15 [27].

Figura 15. *Peculiaridades del Modelo de Calidad en Uso – ISO/IEC 25000* [27].

La norma ISO/IEC 25000 define el proceso para realizar la evaluación de un producto software en cinco actividades [28]:

Figura 16: *Proceso de Evaluación de un Producto Software - ISO/IEC 25000.*

1. **Establecer los requisitos de la evaluación.**

La actividad consiste en obtener cada uno de los requisitos de la evaluación.

Según [28] afirma que se compone de:

1.1 Establecer el propósito de la evaluación: En esta tarea se establece el propósito por el cual la organización quiere evaluar la calidad de un producto de software.

1.2: Obtener los requisitos de calidad del producto: Se identifican las partes interesadas en el producto de software (desarrolladores, posibles adquisidores, usuarios, proveedores, entre otros) y se detallan los requisitos de calidad del producto manipulando un modelo de calidad.

1.3: Identificar las partes del producto que se deben evaluar: Se deben identificar y documentar las partes del producto de software incluidas en la evaluación. Tener en cuenta que el tipo de producto a evaluar depende de la fase en el ciclo de vida en que se realiza la evaluación.

1.4: Definir el rigor de la evaluación: Se centra en definir el rigor de la evaluación basándose en el propósito y uso del producto de software. Al hablar de riesgos se hace referencia a diferentes tipos, tales como riesgos para la seguridad, riesgos económicos o riesgos ambientales.

2. Especificar la evaluación.

En esta actividad se especifican los paradigmas de evaluación (métricas, herramientas y técnicas), junto con los criterios de decisión a aplicar, según [28] afirma los siguientes procesos dentro de esta actividad como:

2.1: Seleccionar los módulos de evaluación: El personal encargado de realizar la evaluación es quien selecciona las métricas de calidad, técnicas y herramientas que cubran todos los requisitos de la evaluación. Las métricas seleccionadas deben poder cotejar con los criterios definidos para poder tomar decisiones. Para esta tarea, se puede tener en cuenta la norma ISO/IEC 25020.

2.2: Definir los criterios de decisión para las métricas: Se definen los criterios de decisión, los cuales son umbrales numéricos que se pueden relacionar con los requisitos de calidad y con los criterios de evaluación para decidir la calidad del producto.

2.3: Definir los criterios de decisión de la evaluación: Se deben definir los criterios para las diferentes características evaluadas. Estos resultados, en un mayor nivel de abstracción, permiten realizar la valoración de la calidad del producto en forma general.

3. Diseñar la evaluación.

En esta actividad se define el plan con las tareas que se deben realizar en la evaluación [28].

3.1: Planificar las actividades de la evaluación: Requiere planificar las actividades de la evaluación, teniendo en cuenta la disponibilidad de los recursos humanos y materiales necesarios, el presupuesto, los métodos de evaluación y estándares adaptados, las herramientas de evaluación, entre otros.

4. Ejecutar la evaluación.

Actividad destinada a la ejecución de las actividades de la evaluación, obteniendo las métricas de calidad y aplicando los criterios de evaluación, según [28] afirma los siguientes procesos dentro de esta actividad como:

4.1: Realizar las mediciones: Realizar las mediciones sobre el producto de software para obtener los valores de las métricas seleccionadas e indicadas en el plan de evaluación. Todos los resultados deben ser registrados.

4.2: Aplicar los criterios de decisión para las métricas: Aplicar los criterios para las métricas sobre valores obtenidos en la medición de un producto.

4.3: Aplicar los criterios de decisión de la evaluación: Se deben aplicar los criterios de decisión de la evaluación, generando como resultado el grado en el que el producto cumple con los requisitos de calidad establecidos.

5. Concluir la evaluación.

En esta última actividad se finaliza la evaluación de la calidad del producto de software, realizando un informe de resultados que será entregado al cliente y a su vez se revisan los resultados obtenidos. Según [28] afirma los siguientes procesos dentro de esta actividad:

5.1: Revisar los resultados de la evaluación: Los encargados de revisar los resultados obtenidos en la evaluación son el cliente y el personal evaluador, con la finalidad de realizar una mejor elucidación y tener una mejor detección de errores.

5.2: Crear el informe de evaluación: Luego de haber analizado los resultados, se elabora un informe de evaluación, en donde se indican los requisitos, los resultados, las limitaciones y restricciones, el personal de evaluación, etc.

5.3: Revisar la calidad de la evaluación: El evaluador se encarga de examinar los resultados de la evaluación junto con la validez del proceso de los indicadores y de las métricas aplicadas. En base a esto se obtiene una realimentación, el cual debe servir para mejorar el proceso de evaluación.

5.4: Tratar los datos de la evaluación: Una vez terminada la evaluación, se debe realizar un proceso de toma de datos acordados con el cliente, hecho por el evaluador, ya sea devolviéndolos, modificándolos, guardándolos, etc.

La norma ISO/ IEC 25000 busca establecer un estándar más completo y evolucionado para evaluar dicha calidad, tratando de llenar los vacíos presentados por sus antecesoras (ISO/IEC 9126 y ISO/ IEC 14598); estableciendo un modelo SQUARE (requisitos para la calidad de productos de software y su evaluación), que a diferencia de sus antecesores, no se centra de manera tan marcada y exclusiva en el proceso de producción (punto especialmente criticado de las normas inicialmente mencionadas), sino que trata de buscar un enfoque de carácter más amplio y completo que, sin descuidar en ningún momento dicho proceso, preste mucha mayor atención a las características internas y externas del producto terminado. De igual

forma, el protocolo permite establecer de antemano los requerimientos de calidad esperados del producto [29].

A continuación, se presenta un cuadro comparativo de cada una de las normas ISO para la realización de proyectos de TI o para el Desarrollo de Software, puesto que es de vital importancia determinar un estándar y norma adecuado para el perfeccionamiento de un producto de calidad que satisfaga al cliente, llevando un orden en el lapso de su desarrollo (Tabla 6).

Organismo	Estándares y Normas	¿Quién lo regula?	Aplicable:	Utilidad:
ISO	ISO 12207	Organización Internacional para la Estandarización (ANSI)	En la aplicación de estándares en proyectos TI.	Se basa en la Modularidad y responsabilidad. Se adapta a las necesidades de cualquiera que lo use.
	ISO 15504		Estructura organización, filosofía administrativa, modelo o ciclo de vida, tecnologías de software o método de desarrollo	Determina la capacidad de tales procesos dentro de las metas de calidad, costo y programación.
	ISO 20000		Gestión de Servicios de TI	Provee una guía para la realización de auditoria y para la remedición de los hallazgos. Garantiza la efectividad de los servicios TI.
	ISO 25000		Guía para el desarrollo de los productos de Software. (ISO/IEC 9126 y en ISO/IEC 14598)	Especificación de requisitos y evaluación de características de calidad interna, externa y calidad en uso; busca establecer un estándar más completo y evolucionado para evaluar

				dicha calidad, tratando para ello de llenar los vacíos presentados por sus antecesoras: ISO/IEC 9126 y ISO/IEC 14598
	ISO 27000		En la Administración de Seguridad de la Información. (ISO/IEC)	Proporciona un modelo para establecer, implementar, monitorear, revisar y mejorar los sistemas.
	ISO 9001, 9000-3. 9004-2		9001.- Describe el sistema de calidad aplicado. 9003.- para el desarrollo de software. 9004-2.- directrices para el servicio de facilidades del software	9001.- utilizado para mantener el desarrollo de un producto que implique diseño. 9003-1.- documento específico que interpreta el ISO 9001. 9004-2.- proporcionado como soporte de usuarios
	ISO 9126		Para el desarrollo de Software	Contribuye en la funcionalidad, confiabilidad, recuperación, usabilidad, eficiencia, facilidad de mantenimiento y portabilidad del software.
	ISO 9000		Establece sistemas de gestión de calidad y seguimiento.	Posibilidad de dar mayor calidad al producto o servicio.

Tabla 6. *Cuadro Comparativo de Normas ISO para el Desarrollo de Software* [30].

Para llevar a cabo el análisis correcto de la información de un software es necesario la utilización de herramientas especializadas, y así cumplir con el proceso de evaluación. Por ello es necesario el análisis de diversas herramientas para este fin.

Herramientas de Calidad Software

Herramientas	***Descripción***
PMD	Analizador estático de código que utiliza unos conjuntos de reglas para identificar problemas dentro del software. Detecta cosas como código duplicado, código muerto (variables, parámetros o métodos sin usar), complejidad de métodos (if innecesarios, etc., te recomiendo aquí este otro post sobre código complejo). Trabaja principalmente con lenguaje Java, aunque, con menos soporte, también posee conjuntos de reglas para JavaScript, xsl y ecmascript. Licencia BSD (distribución de software Berkeley).
Check Style.	Herramienta de análisis estático de código que se utiliza para comprobar que el código analizado cumple con una serie de reglas de estilo. Ejemplo, analiza el código según el estándar "Sun Code Conventions" (mira las cabeceras, importaciones de paquetes, Javadoc, etc.). Trabaja para Java. La licencia es: GNU Lesser General Public License Version 2.1.
SonarQube.	Una herramienta de software libre y gratuita que permite gestionar la calidad del código fuente. Al instalarla podremos recopilar, analizar, y visualizar métricas del código fuente. Sonar es básicamente la fusión de las siguientes herramientas Checkstyle y PMD, más otras como: Findbugs, Clover y Cobertura. También realiza un histórico de todas las métricas del proyecto. Permite visualizar informes con resúmenes de las métricas. Trabaja, principalmente, para Java. Aunque da soporte, gracias a la amplia librería de plugins (algunos de

	pago), a los siguientes lenguajes: ABAP, C, Cobol, C#, Delphi/Pascal, Flex/ActionScript, Groovy, JavaScript, Natural, PHP, PL/SQL, Visual Basic 6, Web y XML. La licencia es: LGPL (Licencia General Pública).
Google CodePro Analytix.	Ofrece un entorno para evaluación de código, métricas, análisis de dependencias, cobertura de código, generación de Test unitarios, etc. Mira las excepciones, refactorizaciones potenciales, convenios de JavaDoc, métricas, etc. Disponible como plugin de Eclipse. Trabaja para Java, concretamente en Eclipse. La herramienta es gratis.
Simian	Herramienta para detectar código duplicado (que es el mayor enemigo de la mantenibilidad) en desarrollos realizados con los lenguajes: Java, C#, C, C++, COBOL, Ruby, JSP, ASP, HTML, XML y Visual Basic. La licencia es libre si su uso está destinado a proyectos OpenSource.

Tabla 7: *Herramientas de Calidad de Software [31].*

Una vez analizadas las herramientas para detectar la calidad de un software y conociendo la complejidad de los datos, se considera el uso de la herramienta SonarQube, puesto que es una plataforma para evaluar código fuente de manera estática y es uno de los más utilizados, ya que es básicamente la fusión de las siguientes herramientas Checkstyle y PMD, más otras como: Findbugs, Clover y Cobertura. Es un software libre que usa diversas herramientas de análisis estático de código fuente como Checkstyle, para obtener métricas que pueden ayudar a mejorar la calidad del código de nuestros programas. Además, tiene soporte para más de 20 lenguajes de programación entre los que se encuentran Java, C#, C / C++, PL / SQL, Cobol, ABAP, Python, JavaScript, etc. [32].

Otra de las razones por la que es importante la herramienta de SonarQube, es porque cubre los 7 ejes principales de la calidad del software (Figura 17) y una vez analizado un proyecto permite observar la información detallada sobre la arquitectura y el diseño, comentarios del programa, código duplicado, reglas de programación acordes con el lenguaje que esté utilizando en el programa, bugs potenciales y su posible solución, complejidad del proyecto [33].

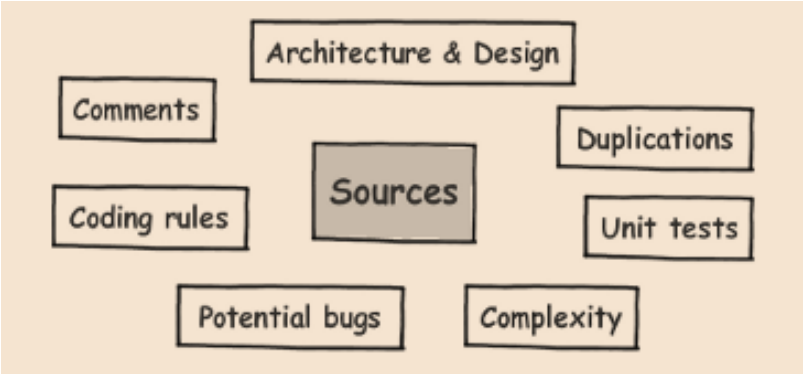


Figura 17: *Los 7 Ejes Principales de la Calidad del Software* [33].

1.3. BASES LEGALES

El uso de sistemas de información y de redes electrónicas, incluida la internet, ha adquirido importancia para el desarrollo del comercio y la producción, permitiendo la realización y concreción de múltiples negocios de trascendental importancia, tanto para el sector público como para el sector privado. A través de este servicio se realizan actos de carácter civil y financiero que es de gran necesidad controlarlos mediante una ley especializada en la materia. Es imprescindible que el Estado Ecuatoriano cuente con herramientas jurídicas que le permitan el uso de los servicios electrónicos, incluido el comercio electrónico y acceder con mayor facilidad a la red de los negocios internacionales cada vez más compleja.

Dentro de la República del Ecuador existe un marco legal titulado *"LEY DE COMERCIO ELECTRONICO, FIRMAS Y MENSAJES DE DATOS",* el cual está vigente desde el 7 de abril de 2002. En el documento se exponen diferentes normas o artículos para la correcto régimen de uso de Tecnologías de la Información, tales como: Mensajes de datos, La Firma Electrónica, los Servicios de Certificación, la Contratación Electrónica y Telemática, la Prestación de Servicios Electrónicos, a través de Redes de Información, incluido el Comercio Electrónico y la Protección a los usuarios de estos Sistemas, Servicios de Internet e Intranet, Organización Informática, etc. Estas se encuentran distribuidas a través de artículos que van desde el Articulo 1 hasta el Artículo 64 [34].

Además, dentro de la República del Ecuador también existe un documento vigente legal titulado *"CÓDIGO ORGÁNICO DE LA ECONOMÍA SOCIAL DE LOS CONOCIMIENTOS, CREATIVIDAD E INNOVACIÓN (INGENIOS)"* [35], el cual fue aprobado por el ex mandatario Sr. Ec. Rafael Correa Delgado el 29 de diciembre del 2016. Dicho documento consta de 628 artículos organizados en cuatro libros, cuya propuesta esencial es generar un cambio en la matriz productiva del conocimiento, especialmente sobre aspectos tales como: La Promoción De La Investigación Responsable, Innovación, Ciencia y Tecnología, Saberes Ancestrales y Tradicionales, Talento Humano y otros, en el contexto de la protección de los derechos de propiedad intelectual. El Ecuador busca con este Código de Ingenios que los ciudadanos innovadores ecuatorianos tengan el apoyo en sus procesos de investigación y así dejar que el Ecuador dependa de recursos finitos como el petróleo, o de importar tecnologías y productos de otros países para saltar a una producción donde el motor sea el talento humano ecuatoriano a través de la generación de ciencia y tecnología nacional [36].

CAPÍTULO II: METODOLOGÍA

2.1. DESCRIPCIÓN DEL LUGAR

La investigación se llevó a cabo en el departamento de Tecnologías de la Información y Comunicación del Gobierno Autónomo Descentralizado de la Provincia de Esmeraldas (GADPE). El departamento se encuentra ubicado en el segundo piso del edificio principal, donde se desarrollan actividades como: dar soporte técnico, facilitar los medios tecnológicos a los demás departamentos para desarrollar y cumplir las actividades tanto administrativas como técnicas**.**

2.2. TIPO DE INVESTIGACIÓN

De acuerdo con el objetivo que se deseaba alcanzar se determinó que es una investigación descriptiva explicativa porque permitió describir, interpretar y comprender paso a paso las características del sistema viáticos del Gobierno Autónomo Descentralizado de la Provincia de Esmeraldas (GADPE), conforme con las características fundamentales del modelo de estándares de calidad de software ISO/IEC 25000.

2.3. DEFINICIÓN CONCEPTUAL OPERACIONALIZACIÓN DE VARIABLES

Para el perfeccionamiento de esta investigación se definió una investigación descriptiva explicativa, puesto que, para realizar una evaluación de calidad de producto de software es necesario aplicar las métricas de calidad interna de la norma 25022 para determinar qué tan exitoso es el producto de software.

Variable	Definición	Definición Operativa	Dimensiones	Indicadores
Calidad Interna	Determina la totalidad de características del producto de software desde una perspectiva interna.	Mantenibilidad	• Modularidad • Reusabilidad • Capacidad de ser analizado • Capacidad de ser modificado • Capacidad de ser probado	Se basará en las fórmulas de las métricas de calidad interna establecidas de la norma 25022.

2.4. MÉTODOS Y TÉCNICAS

2.4.1. MÉTODO CUALITATIVO Y DEDUCTIVO

El presente proyecto de investigación se basó en el método cualitativo y deductivo, porque se empezó detallando los diferentes tipos de modelos de métricas de calidad de software y así seleccionar el adecuado. Además, se evaluó el sistema basándose en la norma internacional ISO 25000 detallando una perspectiva más amplia de lo que se debe mejorar en el software actual para que sea un producto de calidad.

2.4.2. TÉCNICAS E INSTRUMENTOS

Para determinar el nivel de calidad del producto de software analizado se utilizó la técnica del check list en cada fase de realización de la evaluación, que mediante los indicadores se evaluaron los aspectos principales de la característica Mantenibilidad del producto de software como son: modularidad, reusabilidad, capacidad de ser analizado, capacidad de ser modificado y capacidad de ser probado. También permitió ser un apoyo para el respectivo análisis, determinando las conclusiones y recomendaciones óptimas.

2.4.2.1. Representación del Instrumento

Para realizar el análisis de la calidad interna del software viáticos se aplicó una matriz de calidad, la cual permitió realizar la evaluación de la calidad del producto de software de manera eficaz y concisa. La matriz de calidad consiste en apartados principales claramente definidos: tal como se presentan en la Tabla 5, donde se detallan las métricas descritas por la norma ISO 25022, en la Tabla 3 (métricas de Calidad Interna y Externa). Para aplicar el instrumento fue necesario llenar los espacios en blanco respondiendo a cada apartado marcando un visto debajo del criterio que mejor se adecue.

En cuanto el diseño de la evaluación, se tuvieron reuniones periódicamente durante el tiempo del proceso de la auditoría para definir los requisitos de mantenibilidad y analizar los resultados.

2.5. POBLACIÓN Y MUESTRA DE ESTUDIO

Se eligió como población y muestra los datos estadísticos obtenidos sobre un universo de estudio explícito, por lo tanto, debido al ambiente de la investigación elaborada, no aplican.

2.6. DESCRIPCIÓN Y VALIDACIÓN DEL INSTRUMENTO

La obtención de los datos a través de la aplicación del check list basada en una matriz de calidad, fueron analizados de manera rigurosa elaborando un bosquejo con diferentes apartados claves para la ejecución de la actual investigación; los resultados de los apartados obtenidos mediante la aplicación de la herramienta SonarQube abarcaron ítems específicos acorde al modelo seleccionado, el cual facilitó la obtención de la información y requerimientos permitiendo establecer las debidas recomendaciones para mejorar la calidad del producto de software.

2.7. TÉCNICAS DE PROCESAMIENTO Y ANÁLISIS DE DATOS

Para llevar a cabo la auditoría del software viáticos se definió en primer aspecto el equipo de trabajo que está formado por el asesor de la investigación que gestionó el programa de evaluación y el alumno investigador que será quien lleve el liderazgo de la auditoria. Luego se realizó el cronograma de actividades a realizar calculando un tiempo de finalización de evaluación en el mes de diciembre. El software Viáticos del GADPE está compuesto por dos softwares internos que son el software Solicitud de Licencia y el software Informe de Solicitud de Licencia; es decir que cada funcionario de la organización debe primero realizar una solicitud de licencia para poder adquirir sus viáticos, luego una vez realizada la solicitud debe realizar un informe del mismo detallando todas las actividades realizadas de ese viaje para que sean cubiertos sus gastos por la empresa, caso contrario no serán cubiertos los gastos.

Dentro del proceso y análisis de los resultados obtenidos durante la investigación fue necesario analizar herramientas especializadas en pruebas de software para obtener las estadísticas acerca de la mantenibilidad de la aplicación viáticos y así llevar a cabo un buen manejo de análisis apropiado de la misma. Una vez analizadas las herramientas, para llevar a cabo el proceso de pruebas y así examinar los sitios críticos de la aplicación se consideró el instrumento SonarQube ya que es una herramienta open sources que permite gestionar, analizar y medir la calidad del producto de software mediante una serie de métricas que ofrece.

El análisis de la evaluación se realizó en función de la información obtenida al realizar un test del código fuente ejecutado a través de la herramienta SonarQube, a partir de esos resultados se evaluó cada una de las métricas obteniendo las medidas que establecían cada una. Además, se analizó la documentación disponible por ser un requisito importante de mantenibilidad para saber el funcionamiento del sistema.

2.8. NORMAS ÉTICAS

Los datos obtenidos durante el proceso de la investigación sobre la evaluación del producto de software Viáticos del GADPE no serán divulgadas bajo ningún consentimiento y guardan total reserva. Toda la información recolectada fue utilizada con responsabilidad únicamente para la prueba de evaluación del software y el desarrollo de la investigación; además el documento está desarrollado bajo el régimen académico de la universidad.

CAPÍTULO III: RESULTADOS

En este apartado se mostrará de manera detallada el resultado del proceso de la evaluación de la mantenibilidad del software viáticos. Serán presentadas de acuerdo a las subcaracteristicas de la mantenibilidad.

- **Modularidad**
    - **Capacidad de condensación**

Capacidad de contar el número de componentes que no son afectados por cambios de otros componentes y el número total de componentes específicos, donde el valor que se debe obtener es 0<=X<=1, el más cercano a 0 es el mejor (Tabla 9).

Función de Medición:

Los resultados obtenidos se clasifican en función de la siguiente tabla de medición según lo explícito por SonarQube, con valor de 0 a 1:

Defectos de Modularidad	RESULTADO
0-0,4	Muy Buena
0,5-0,9	Buena
+1	Muy Mala

Tabla 8: *Función de Medición de Capacidad de condensación.*

Formula de Medición:

X= B/A

- A = Número de componentes que no son afectados por cambios de otros componentes
- B = Número total de componentes específicos

X=955/4308=0,2

Defectos de Modularidad	RESULTADO
0-0,4	Muy Buena
0,5-0,9	Buena
+1	Muy Mala

Tabla 9: *Resumen de Análisis de la Capacidad de Condensación.*

De acuerdo a la función de medición y valor obtenido en ambos subsistemas, indica que el sistema está en una condición muy buena en su capacidad de condensación dando como resultado un valor de 0,2 es decir que está apto para trabajar sin problemas.

- **Acoplamiento de clases**

Contar el número de relaciones que tiene una función con respecto a otras clases, donde el valor que se debe obtener es 1<=X<=5, el más cercano a 1, es el mejor (Tabla 11).

Función de Medición:

Los resultados obtenidos se clasifican en función de la siguiente tabla de medición según lo explícito por SonarQube, con valor de 1 a 5:

Defectos de Modularidad	RESULTADO
1-2	Muy Buena
3-4	Buena
+5	Muy Mala

Tabla 10: *Función de Medición de Acoplamiento de Clases.*

Formula de Medición:

X=A

- A = Número de relaciones que tiene una función con respecto a otras clases

X=3

Defectos de Modularidad	RESULTADO
1-2	Muy Buena
3-4	Buena
+5	Muy Mala

Tabla 11: *Resumen Análisis de Acoplamiento de clases.*

De acuerdo a la función de medición y valor obtenido en ambos subsistemas, indica que el sistema está en una buena condición dando como resultado un valor de 3, lo que indica que el sistema tiene un buen acoplamiento en sus clases.

- **Reusabilidad**
    - **Deuda Técnica**

La deuda técnica, es el coste y las utilidades a pagar por realizar un trabajo mal, entendiéndolo como la mala práctica a la hora de diseñar e implementar la aplicación. El sobre esfuerzo a pagar para mantener un producto software mal hecho.

Una de las características más importantes de SonarQube es que calcula la deuda técnica del proyecto, en las figuras 18, 19, 20 y 21 se ve que la ratio de deuda técnica es solamente de un 4.8% y 5.3% que son resultados muy buenos. Además, el rating que le da SonarQube al software de Licencias-Viáticos es el mejor (A), como se ve en la Figura 19; el rating para el software de Informes- Viáticos que le da SonarQube no es el mejor, pero es el segundo lugar que es bueno (B).

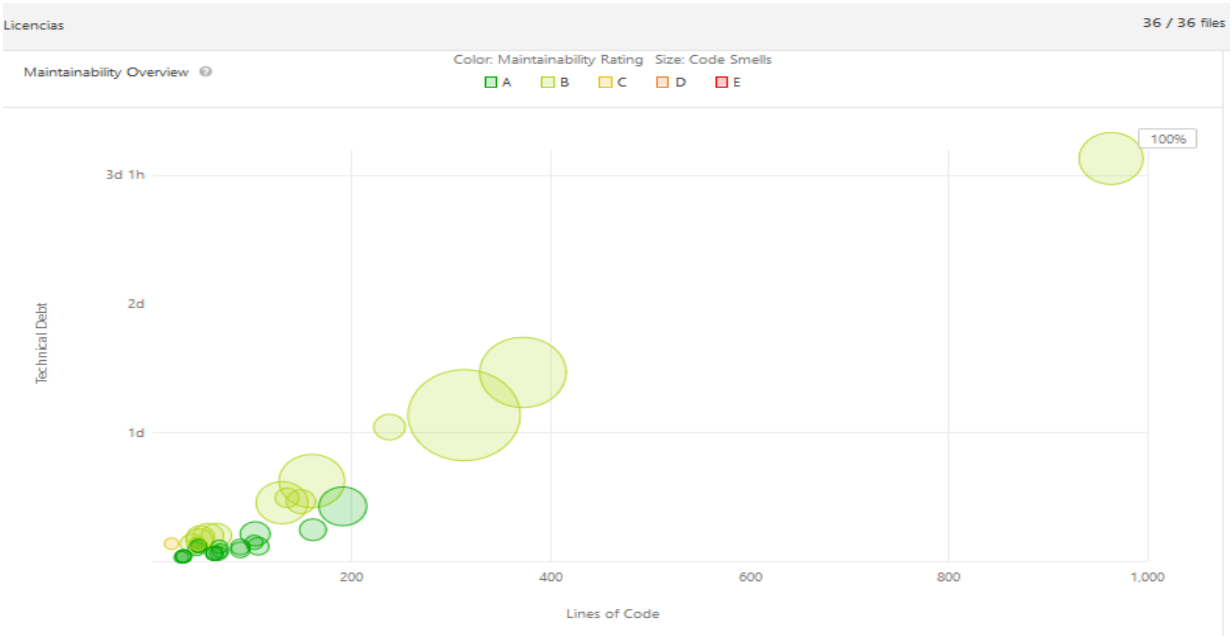


Figura 18: *Deuda Técnica Viáticos Licencias.*

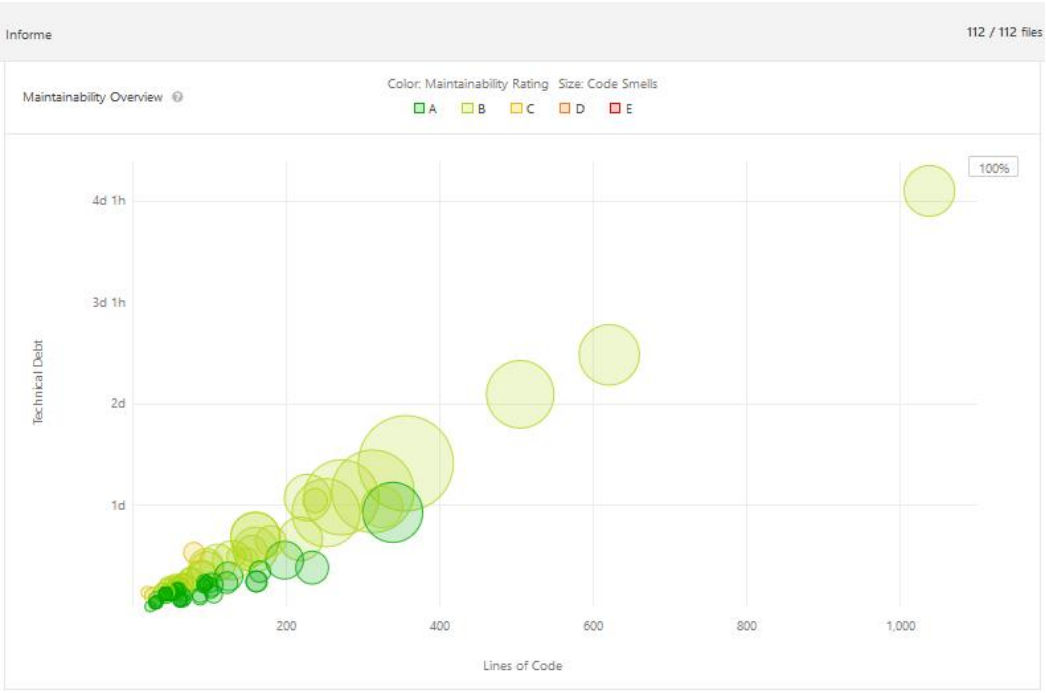


Figura 19: *Deuda Técnica Viáticos – Informe*

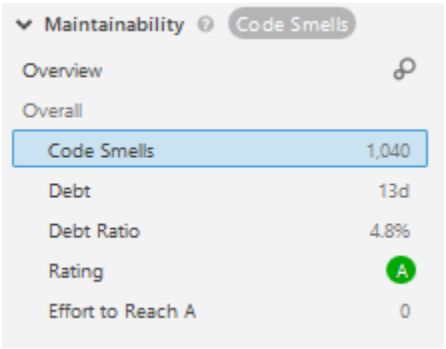


Figura 20: *Rating de Mantenibilidad Viáticos-Licencias.*

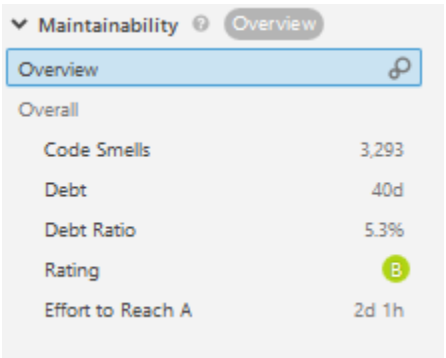


Figura 21:*Rating de Mantenibilidad Viáticos-Informes.*

- **Capacidad de ser Analizado**
    - **S. Capacidad de pistas de auditoría**

Contar el número de datos realmente grabadas durante la operación y el número de datos previstos a grabarse para controlar el estado del sistema durante la operación, en donde el valor que se debe obtener es 0<=X<=1, el más cercano a 1, es el mejor (Tabla 13).

Función de Medición:

Los resultados obtenidos se clasifican en función de la siguiente tabla de medición según lo explícito por SonarQube, con valor de 0 a 1:

Defectos de ser Analizado	RESULTADO
0-0,4	Muy Mala
0,5-0,9	Buena
1	Muy Bueno

Tabla 12: *Función de Medición Capacidad de Pistas de Auditoría.*

Formula de Medición:

X = A/B

- A = Número de datos realmente grabadas durante la operación
- B = Número de datos previstos a grabarse para controlar el estado del sistema durante la operación

X=4308/4308=1

Defectos de ser Analizado	RESULTADO
0-0,4	Muy Mala
0,5-0,9	Buena
1	Muy Bueno

Tabla 13: *Resumen Análisis de Capacidad de Pistas de Auditoría.*

De acuerdo a la función de medición y valor obtenido en ambos subsistemas, indica que el sistema está en una muy buena condición dando como resultado un valor de 1, lo que indica que en el sistema los usuarios pueden identificar fácilmente la operación específica que causa un fallo.

- **Código repetido**

El código repetido es una métrica básica de calidad, y junto con la complejidad ciclomática, es el mayor enemigo de la mantenibilidad.

Función de Medición:

Los resultados obtenidos se clasifican en función de la siguiente tabla de medición según lo explícito por SonarQube, con valor de 1 a 100:

Defectos de la capacidad para ser analizado	RESULTADO
1-25	Muy Buena
26-50	Buena
51-75	Mala
76-100	Muy Mala

Tabla 14: *Función de Medición de Código Repetido*

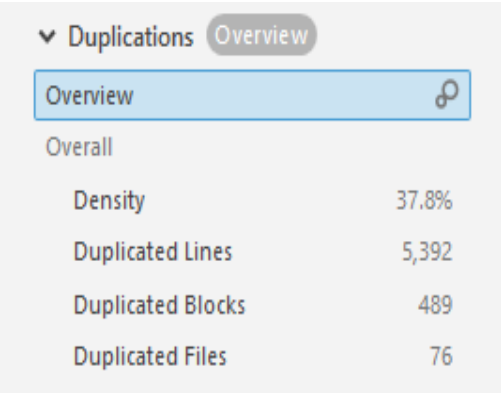


Figura 22: *Código Duplicado Informe-Viáticos.*

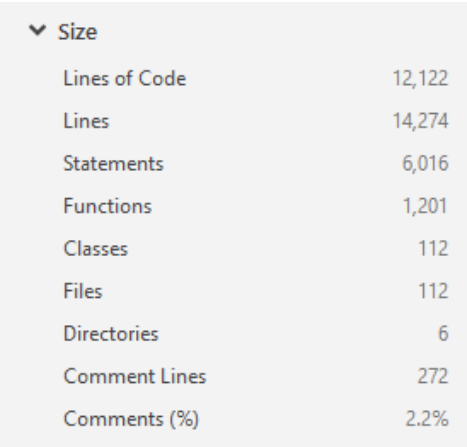


Figura 23: *Datos del Tamaño del Sistema Informe-Viáticos*

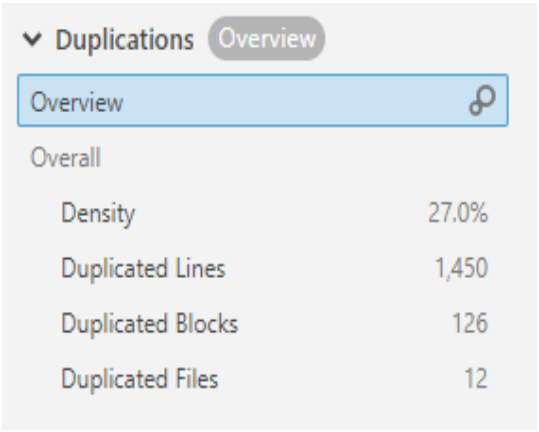


***Figura 24**: Código Duplicado Licencia-Viáticos.*

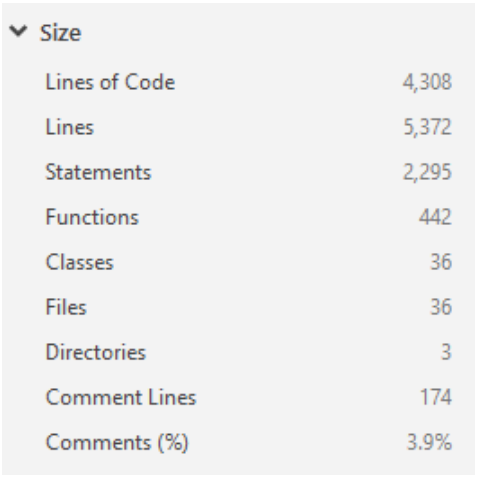


***Figura 25:** Datos del Tamaño del Sistema Licencia-Viáticos*

Para calcular la densidad de código repetido SonarQube aplica lo siguiente:

Fórmula de medición:

DCR=LD/LT *100

- LD=Líneas de código duplicadas
- LT=Líneas de totales del código
- DCR=Densidad de código repetido

$$DCRInforme=\frac{5392}{14274} * 100=37,8$$

$$DCRLicencia=\frac{1450}{5372} * 100=27,0$$

Como se puede ver en las figuras 22, 23, 24 y 25 los dos sistemas que conforman el software de Viáticos del GADPE poseen líneas de códigos duplicadas con un porcentaje no tan preocupante, pero si es importante revisar este aspecto ya que si se llegaría a obtener un valor de porcentaje alto de código repetido no favorecería la mantenibilidad del producto de software y conlleva a poseer varios riesgos como:

- ✓ El proyecto tiende a tener mayor código innecesario (a más líneas de código más complejo es el mantenimiento, y más costoso).
- ✓ Arroja los costes elevados (Mayor sitio de modificación).
- ✓ Aumenta los riesgos (Dificultad en encontrar las repeticiones)

Defectos de la capacidad para ser analizado	RESULTADO
1-25	Muy Buena
26-50	Buena
51-75	Mala
76-100	Muy Mala

Tabla 15:*Resumen de Análisis de Código Repetido*

De acuerdo a la función de medición y valor obtenido en ambos subsistemas, indica que al unir sus resultados el sistema está en una condición mala para ser analizando dando como resultado un valor de 64,8 % (Figura 26 y Figura 27) de complejidad que es mala.

En las siguientes figuras se puede observar el porcentaje de código repetido en el proyecto Informe-Viáticos y Licencia Viáticos:

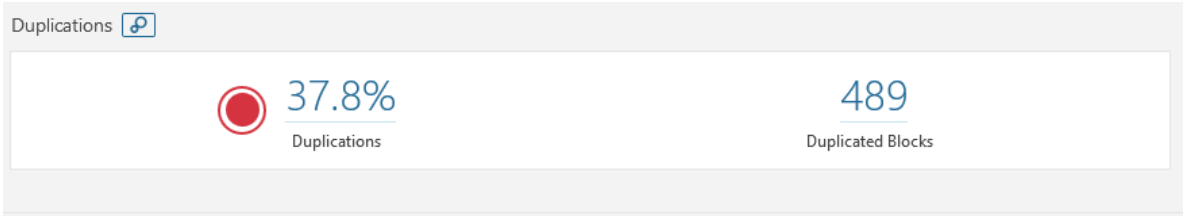


Figura 26: *Código Duplicado Informe-Viáticos*

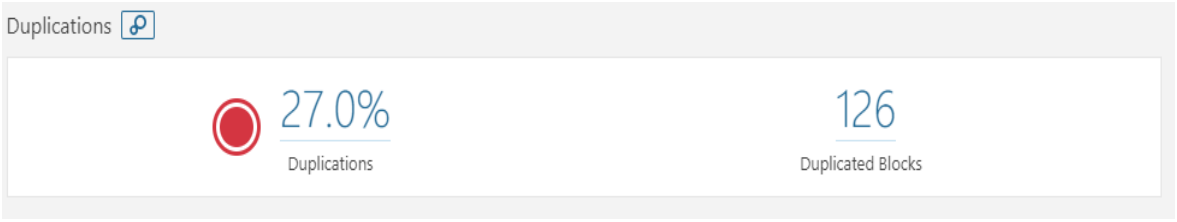


Figura 27: *Código Duplicado Licencias-Viáticos*

- **Capacidad de ser modificado**
    - **Complejidad Ciclomática**

SonarQube al analizar el código para calcular la complejidad Ciclomática mantiene un contador. Cuando el flujo de una función se altera, es decir, se produce un salto, este contador de la complejidad aumenta en 1. Cada función tiene como mínimo una complejidad de 1, se suma 1 por la llamada al método, en donde el valor deseado es `1<=X<=50`, el más cercano a 1, es el mejor.

En JAVA, por ejemplo, estas son las reglas para medir la complejidad:

- ✓ Estas palabras incrementan la complejidad: "if, for, while, case, catch, throw, return (that is not the last statement of a method), &&, ||,"
- ✓ "Else, default" no incrementan la complejidad.
- ✓ Un método simple con una sentencia switch y un enorme bloque de sentencias de casos puede tener un valor sorprendentemente alto de complejidad.

Función de Medición:

Los resultados obtenidos se clasifican en función de la siguiente tabla propuesta por Thomas McCabe, con valor de 1 a 50:

Defectos de la capacidad para ser modificado	Valor de Riesgo
1-10	Programa Simple, sin mucho riesgo
11-20	Más complejo, riesgo moderado
21-50	Complejo, programa de alto riesgo
+50	Muy alto riesgo

***Tabla 16**: Función de Medición Complejidad Ciclomática*

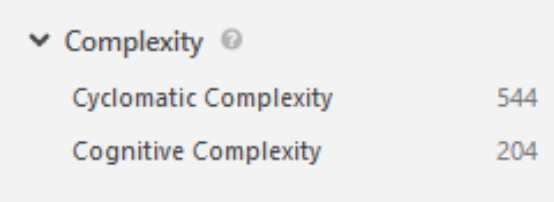


***Figura 28:** Resumen Complejidad Ciclomática Licencias-Viáticos*

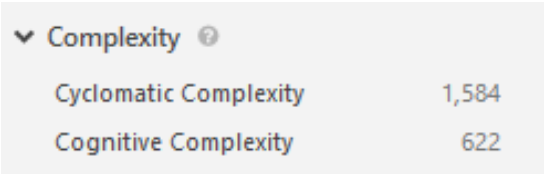


Figura 29: *Resumen Complejidad Ciclomática Informe-Viáticos*

Para analizar un poco la complejidad ciclomática más detallada, en SonarQube se puede analizar uno a uno la complejidad de cada fichero JAVA del software, en la figura 30 se observa un ejemplo de la complejidad de lic_licencia.java; se puede notar que la complejidad en este archivo es 78, menos elevada que la complejidad de Clic_inf_liquidacion.java del proyecto informe-viáticos, en la figura 31 con una complejidad de 100; siendo el archivo con más complejidad ciclomática del sistema viáticos (Figura 32).

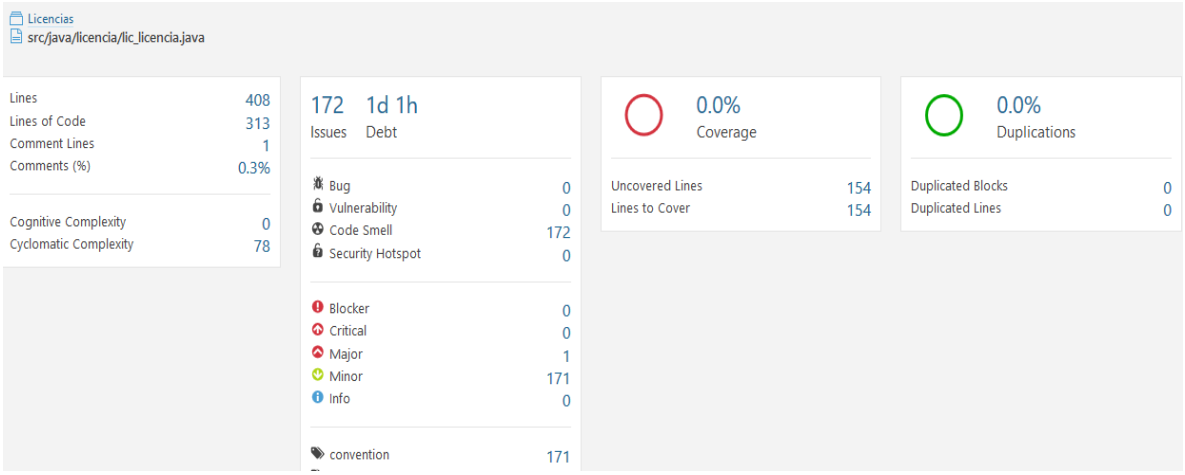


Figura 30: *Complejidad Ciclomática lic_licencia.java (Licencia-Viáticos)*

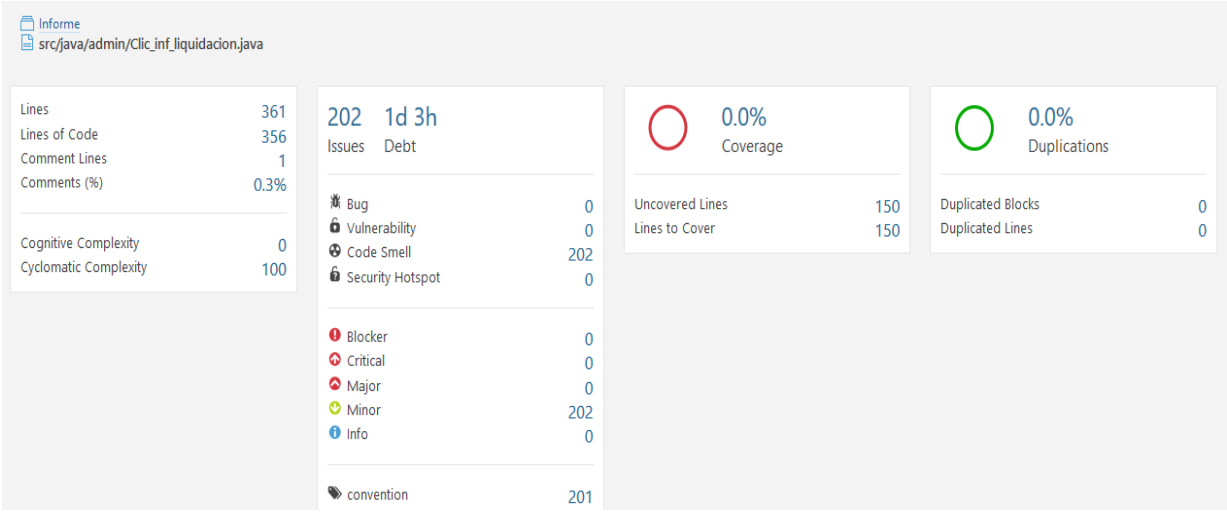


Figura 31: *Complejidad Ciclomática Clic_inf_liquidacion.java (Informe-Viáticos)*

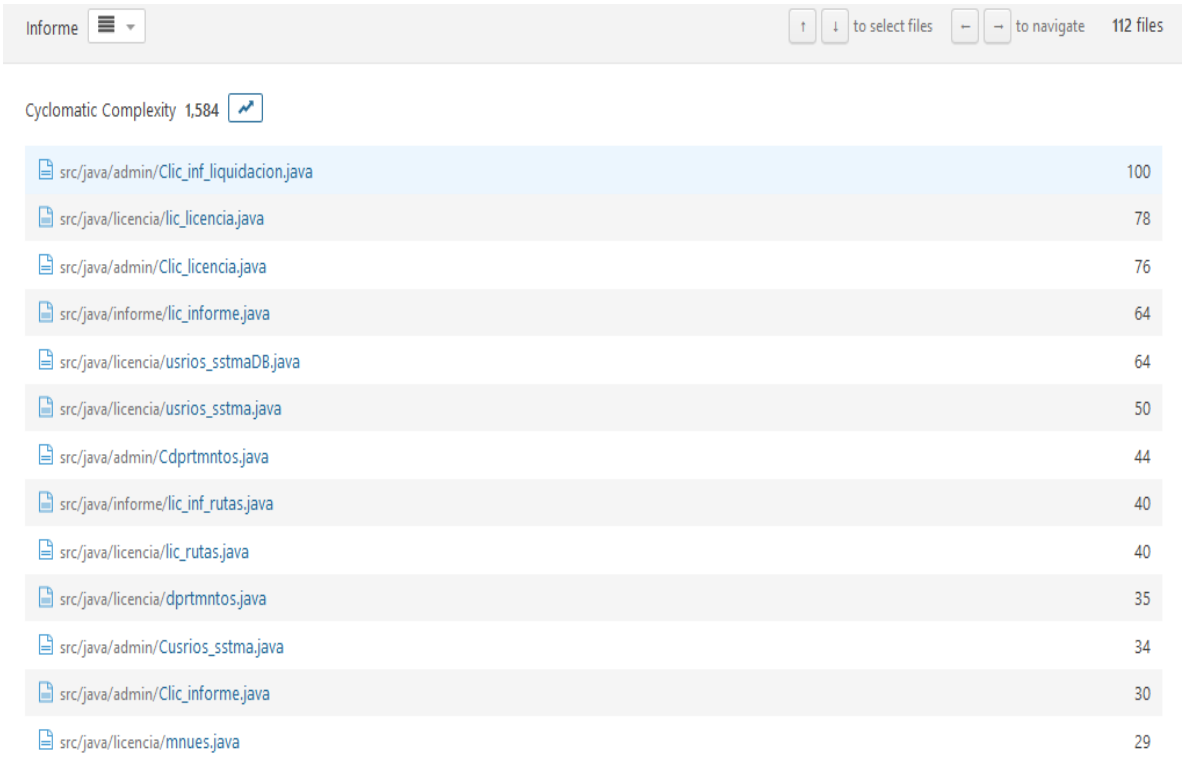


Figura 32: *Complejidad Ciclomática de Cada Archivo del Sistema.*

Fórmula de medición:

X = A+1

- A = Número de instrucciones condicionales que tiene una función

XLicencias= 544

XInformes= 1584

Defectos de la capacidad para ser modificado	Valor de Riesgo
1-10	Programa Simple, sin mucho riesgo
11-20	Más complejo, riesgo moderado
21-50	Complejo, programa de alto riesgo
+50	Muy alto riesgo

Tabla 17: *Resumen Análisis de la Complejidad Ciclomática*

De acuerdo a la función de medición y valor obtenido en ambos subsistemas, como se puede ver en las Figuras 28 y 29, la complejidad ciclomática del proyecto total es superada de 50. Además, se puede observar el desglose de la complejidad por funciones, clases y ficheros (Figura 32). Por consiguiente, en la tabla de función de medición de la complejidad ciclomática, se observa que la complejidad del software viáticos indica que es muy alto en riesgo, es decir supera dicho valor y se hace muy difícil probarlo, entenderlo y modificarlo.

- **Grado de localización de corrección de impacto**

Contar el número de fallas aparecidas después que se ha resuelto un fallo y contar el número de fallas resueltas, donde el valor deseado es 0<=X<=1, el más cercano a 0, es el mejor (Tabla 19).

Función de Medición:

Los resultados obtenidos se clasifican en función de la siguiente tabla según SonarQube, con valor de 0 a 1:

Capacidad de ser Modificado	RESULTADO
0-0,4	Muy Buena
0,5-0,9	Buena
+1	Muy Malo

Tabla 18: *Función de Medición de Grado de Localización de Corrección de Impacto.*

Fórmula de medición:

X = A/B

A = Número de fallas aparecidas después que se ha resuelto un fallo

B = Número de fallas resueltas

X =955/1311=0,72

Capacidad de ser Modificado	RESULTADO
0-0,4	Muy Buena
0,5-0,9	Buena
+1	Muy Malo

Tabla 19: *Grado de localización de Corrección de Impacto.*

- **Capacidad de ser probado.**
  - **Capacidad de prueba autónoma**

Para llegar a obtener un menor coste en los proyectos de software es necesario que el sistema sea entendible de manera fácil inclusive para los usuarios que no poseen conocimiento del mismo. Por ende, es de suma importancia que los nombres de variables tengan relación con la función de la misma.

Dentro del análisis del software Viáticos se llegó a la conclusión de que en términos generales los nombres de las variables no son tan ajustadas a su función, en la siguiente figura 33 se puede observar cómo los nombres de las variables no se puede identificar fácilmente el contenido de ellas.

```
public class srvlt_lic_gastos_buscar extends HttpServlet {

    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        request.setCharacterEncoding("UTF-8");
        PrintWriter out = response.getWriter();
```

Figura 33: *Ejemplo Nombre de Variables Viáticos.*

Otro aspecto importante que se analizó es el tamaño del programa, puesto que a la hora de evaluar un proyecto mientras mayor tamaño tenga un programa mayor será el coste de las pruebas.

En las siguientes imágenes se puede observar el tamaño total de cada software que compone el sistema de Viáticos, Licencias e Informes:

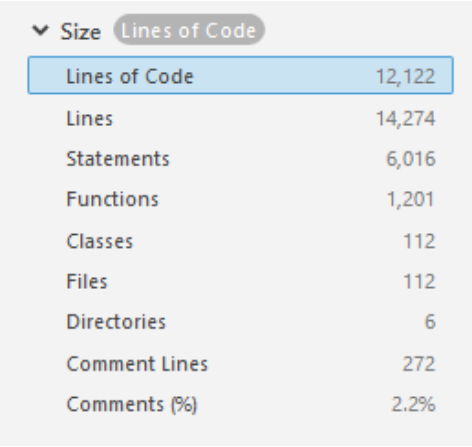


Figura 34: *Tamaño del Software Informe-Viáticos*

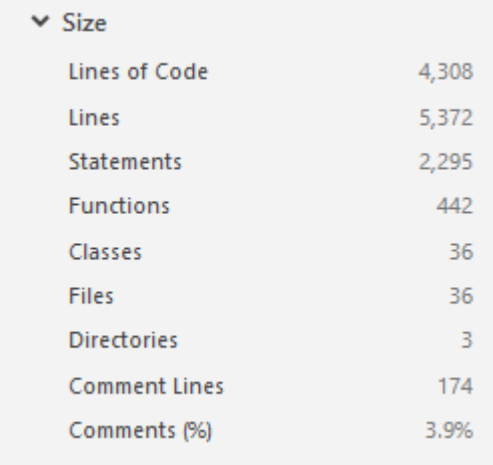


Figura 35: *Tamaño del Software Licencia-Viáticos*

Hay cerca de 20.000 líneas escritas, de ellas unas 17.000 son líneas de código (sin comentarios), están organizadas en 148 ficheros y 9 directorios. Además, hay 1.643 funciones y 148 clases.

- **Análisis general de los subsistemas.**

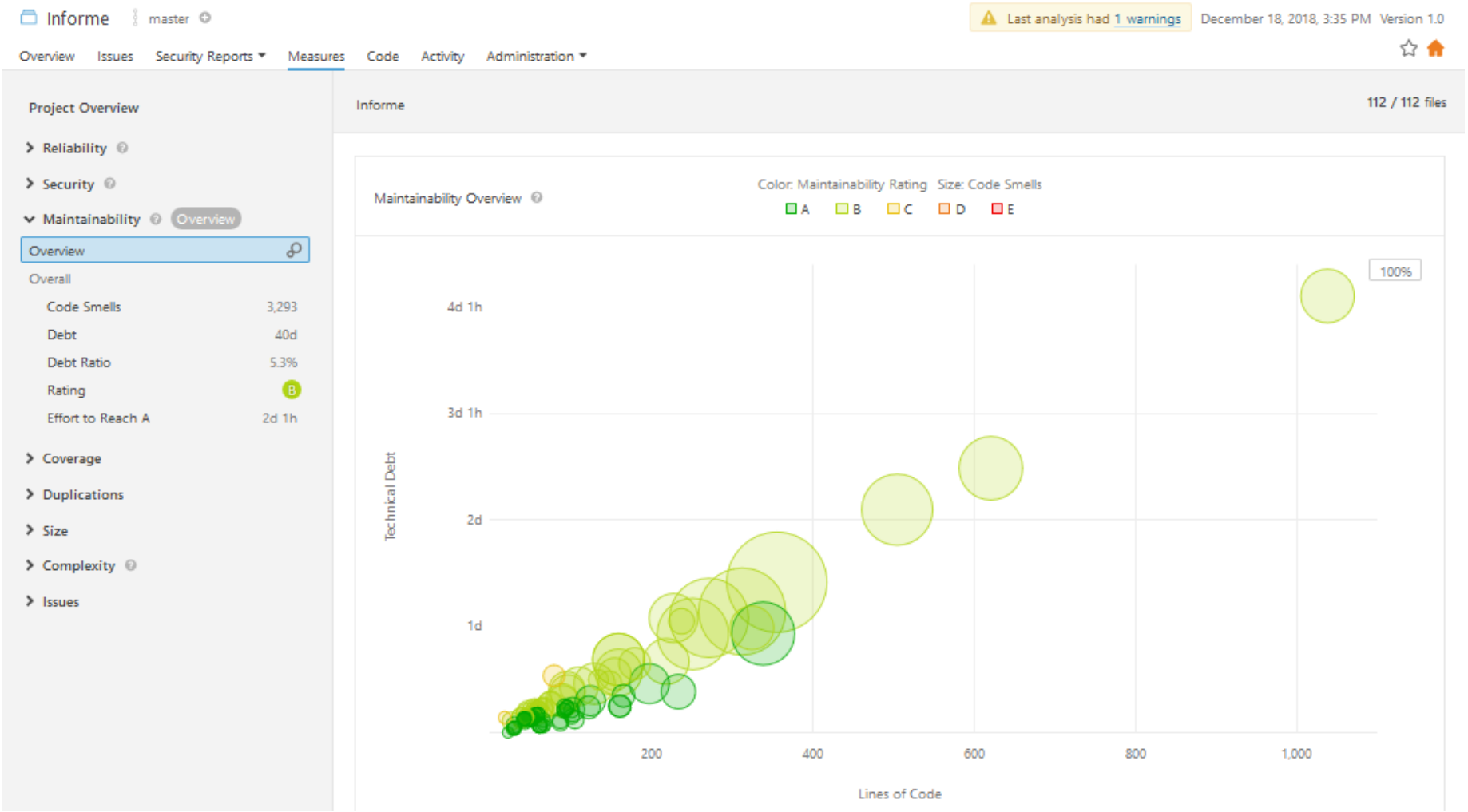


Figura 36:*Resumen Análisis General Informe-Viáticos*

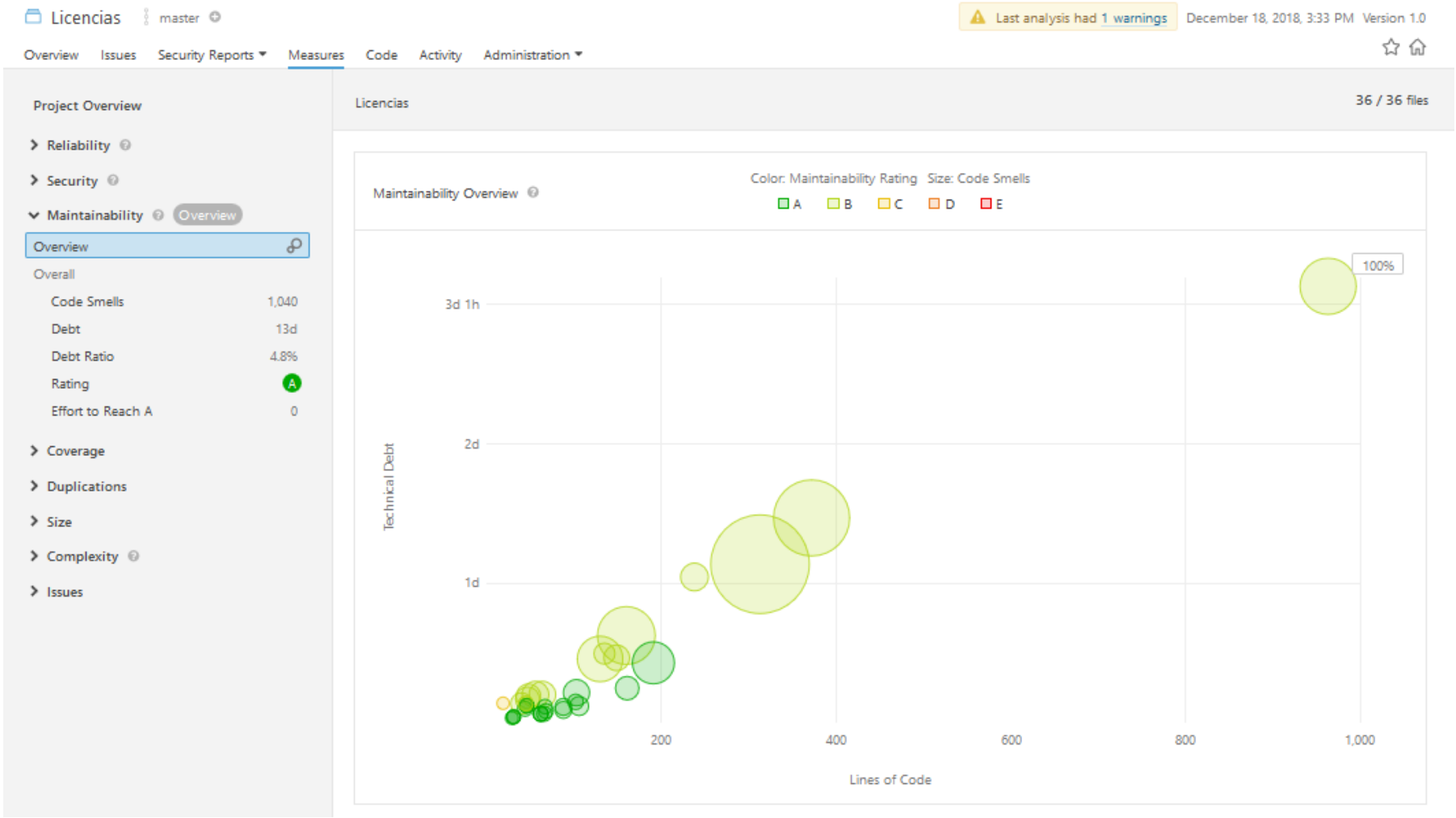


Figura 37: *Resumen Análisis General Licencias-Viáticos*

Función de Medición:

SonarQube mantiene una cuadricula de clasificación de mantenimiento SQALE (Evaluación de la calidad del software basada en las expectativas del ciclo de vida) es un método para respaldar la evaluación de un código fuente de una aplicación de software expresada en la siguiente Tabla [37]:

A = 0-0.05, B = 0.06-0.1, C = 0.11-0.20, D = 0.21-0.5, E = 0.51-1

Costo de remediación en porcentaje	Calificación de Mantenibilidad
0-0.05	A
0.06-0.1	B
0.11-0.20	C
0.21-0.5	D
0.51-1	E

Tabla 20: *Función de Medición de Mantenibilidad*

Es decir que si el costo de remediación es [37]:

- <= 5% del tiempo que ya ha entrado en la aplicación, la calificación es A
- Entre el 6 y el 10% la calificación es una B.
- Entre el 11 y el 20% la calificación es una C.
- Entre 21 y 50% la calificación es una D
- algo más del 50% es una E

Resultado por subsistema:

Costo de remediación en porcentaje	Calificación de Mantenibilidad
0-0.05	A
0.06-0.1	B
0.11-0.20	C
0.21-0.5	D
0.51-1	E

Tabla 21: *Resumen Análisis de Mantenibilidad Informe-viáticos*

Costo de remediación en porcentaje	Calificación de Mantenibilidad
0-0.05	A
0.06-0.1	B
0.11-0.20	C
0.21-0.5	D
0.51-1	E

Tabla 22: *Resumen Análisis Mantenibilidad Licencia-Viáticos*

De acuerdo a los resultados obtenidos del análisis del software, se resume en que la aplicación de Viáticos ha pasado la auditoría de mantenibilidad debido al cumplimiento de las métricas establecidas para la característica de mantenibilidad del software. La calificación de mantenibilidad del subsistema Informe – Viáticos dio como resultado B (Figura 36) según el análisis en SonarQube, y este se aplica cuando el índice de deuda técnica se encuentra entre el 6 – 10%; por lo tanto, el subsistema Informe - Viáticos sí es un software con mantenibilidad Buena que está listo para su producción.

En cuanto al subsistema Licencia – Viáticos dio como resultado A (Figura 37) según el análisis en SonarQube, y este se aplica cuando el índice de deuda técnica es inferior al 5.0%; por lo tanto, el subsistema Licencia - Viáticos sí es un software con mantenibilidad satisfactoria que está listo para su producción. En conclusión, el sistema viáticos en general se encuentra en una mantenibilidad buena y puede realizar cambios en errores dentro de un tiempo moderado con un poco de esfuerzo.

CAPÍTULO IV: DISCUSIÓN

Los estándares son un conjunto de reglas pre establecidas que sirven como modelo para poder alcanzar un objetivo. Para la evaluación de calidad de software está la ISO 25000 la cual propone un modelo detallado, en el que describe las siguientes características: Adecuación Funcionalidad, Usabilidad, Fiabilidad, Eficiencia en el desempeño, Mantenibilidad, Portabilidad, Seguridad, Compatibilidad [26].

La Evaluación de software es un proceso que determina en qué medida se han logrado los objetivos previamente establecidos que supone un juicio de valor sobre la programación establecida, y que se emite al contrastar esa información con dichos objetivos [10].

"Son un componente crítico del ciclo de vida del desarrollo de software, cuyo objetivo es ayudar a mejorar la calidad y fiabilidad del producto mediante la identificación de defectos, la prevención de errores, y la observación y presentación de reportes" [11]. Se considera que evaluar un producto de software es la única forma precisa para determinar fallos en la programación o incluso en otra parte del proceso del software, que presenta el producto que se esté desarrollando.

En esta investigación se realizó el análisis de comparación de los sistemas establecidos del departamento de Tics del GADPE, en donde se evaluó el sistema de Vináticos que, de acuerdo a los resultados de la evaluación del software, cubre con la característica de mantenibilidad que plantea la ISO 25000.

Finalmente se puede decir que el estándar ISO/IEE 25000 puede ser utilizada en cualquier organización, teniendo en cuenta que son estándares que viven en constante cambio y que fueron creadas para mecanizar procesos, facilitando la administración y disminución de costos.

CAPÍTULO V: CONCLUSIONES

- La norma ISO/IEE 25000 presenta un modelo SQUARE (Software product Quality Requirements and Evaluation - Requisitos de calidad y evaluación de productos de software) agrupando dos estándares anteriores como son la ISO/IEE 9126 y la ISO/IEE 14598 brindando la unificación y organización de dos técnicas importantes como es la evaluación y especificación de requerimientos de calidad del software, especialmente el modelo SQUARE va dirigido a las empresas dedicadas al desarrollo de software y a las empresas que de manera interna crean sus propios sistemas para el perfeccionamiento de su negocio.
- La aplicación de Viáticos ha pasado la auditoría de mantenibilidad debido al cumplimiento de las métricas establecidas para la característica de mantenibilidad del software. Sin embargo, es importante inspeccionar algunos aspectos del programa como la duplicación de código, el tratamiento de excepciones y los errores en el código que permite que el coste de la deuda técnica incremente debido a la cantidad de días y horas establecidas para reestructurar la aplicación.
- La predisposición de aceptar los cambios que mejoren el producto de software no son parcialmente altas; esto se ve evidenciado en los resultados obtenidos en el análisis del sistema viáticos en donde este posee un moderado potencial de mantenibilidad en el software, pero es importante perfeccionar la calidad del software con el fin de corregir los moderados errores establecidos con la herramienta SonarQube.

CAPÍTULO VI: RECOMENDACIONES

- Se recomienda revisar el código fuente de la aplicación para eliminar los errores descritos durante la ejecución de auditoría.
- Las consideraciones descritas en la ejecución de auditoría se deberán tener en cuenta para posibles modificaciones futuras de la aplicación con el fin de conservar su capacidad de mantenimiento.
- Es importante invitar a los desarrolladores del GADPE que realicen también pruebas a sus códigos con el fin de mejorar la calidad interna del software, pero que no sea el mismo quien ponga el visto bueno con satisfacción, sino que se recomienda contar con un grupo de testeadores que se encarguen de preparar los ambientes de pruebas, y cumplan con la revisión del software antes de la puesta en producción.

REFERENCIAS

[1] J. Villanes, Baldeón; Edú, "Método Para La Evaluación De Calidad De Software Basado En Iso/Iec 25000," 2015.

[2] L. E. López Echeverry, Ana Maria; Cabrera, Cesar; Valencia Ayala, "Introducción a la calidad de software," *Sci. Tech.*, vol. 2, no. 39, pp. 326–331, 2008.

[3] H. T. Punter, R. J. Kusters, J. J. . Trienekens, T. M. A. Bemelmans, and A. C. Brombacher, "The W-Process for software product evaluation: A method for goal-oriented implementation of the ISO 14598 standard," *Softw. Qual. J.*, vol. 12, no. 2, pp. 137–158, 2004.

[4] N. León, L. Gómez, and J. Pimentel, "PRODUCTOS SOFTWARE ENMARCADOS EN ACTIVIDADES DE INVESTIGACIÓN Computational Tool for Assessment of Software Products Under Research Activities," *Sci. Tech.*, vol. 47, no. 48, pp. 93–98, 2011.

[5] R. S. Pressman, *Ingeniería del software. Un enfoque práctico*, ISBN: 978-. Mexico, 2010.

[6] R. S. Pressman, *Ingeniería del software. Un enfoque práctico*, Séptima Ed. Mexico, 2010.

[7] S. Piedrahita Mesa, "Construcción de una herramienta para evaluar la calidad de un producto software," Universidad EAFIT/Medellin, 2007.

[8] P. L. Alfonzo, "Revisión de modelos para evaluar la calidad de productos Web. Experimentación en portales bancarios del NEA.," Universidad Nacional de La Plata, Facultad de Informática, 2012.

[9] M. Rodriguez-Monje, *Calidad de Procesos y Productos Software. Calidad Del Producto Software ISO/IEC 25000*. 2010.

[10] "EVALUACIÓN." [Online]. Available: https://previa.uclm.es/profesorado/ricardo/practicum/relieve/evaluacion.htm. [Accessed: 27-Jun-2018].

[11] I. Gallesdic and G. Killiospy, “Software Testing as Science La Prueba del Software como Ciencia.,” *Revista antioqueña de las ciencias computacionales y la ingeniería de software*, vol. 3, no. 1, p. 33, 2013.

[12] I. Sommerville, *Software Engineering*, Novena Edi. Mexico: Mexico: Pearson Educación de Mexico, S.A. de C.V., 2011.

[13] J. A. Mera Paz, “Análisis del proceso de pruebas de calidad de software,” *Ing. Solidar.*, vol. 12, no. 20, pp. 163–176, 2016.

[14] A. Yagüe and J. Garbajosa, “Comparativa práctica de las pruebas en entornos tradicionales y ágiles,” *REICIS. Rev. Española Innovación, Calid. e Ing. del Softw.*, vol. 5, no. 4, pp. 19–32, 2009.

[15] Manuel Blázquez Ochando, “Evaluación de Sistemas de Información y Usuarios,” *Dpto. Biblioteconomía y Documentación Universidad Complutense de Madrid*, 2012. [Online]. Available: http://ccdoc-evaluacionsistemasinformacion.blogspot.com/2011/02/03-que-es-la-evaluacion.html. [Accessed: 27-Jun-2018].

[16] J. A. Lugo García, S. Torres López, P. Piñero Pérez, and R. Delgado Victore, “CONTROL DE LA EJECUCIÓN DE PROYECTOS BASADO EN INDICADORES Y LÓGICA BORROSA PROJECTS EXECUTION CONTROL BASED ON INDICATORS AND FUZZY LOGIC,” vol. 4, no. 1, pp. 15–35, 2013.

[17] J. Luque Claveras, “Métricas de productividad del software para la gestión de proyectos,” Universitat Oberta de Catalunya, 2015.

[18] B. Pereira, F. Ayaach, H. Quintero, I. Granadillo, and J. Bustamante, “Métricas de Calidad de Software,” in *Calidad del Software*, 2012, p. 51.

[19] E. A. Chisaguano Balseca, “Evaluación de Calidad de Productos Software en Empresas de desarrollo de Software aplicando la Norma ISO/IEC 25000,” ESCUELA POLITÉCNICA NACIONAL, 2014.

[20] P. L. Alfonzo and S. I. Mariño, “Los estándares internacionales y su importancia para la industria del software,” *Buenos Aires, 15-04-2013*, 2013. [Online]. Available:

http://www.cyta.com.ar/ta1202/v12n2a3.htm. [Accessed: 10-Jul-2018].

[21] "SQA consultoría de sistemas - Calidad Interna y Externa de Producto," 2011. [Online]. Available: http://www.sqa.es/webSQA/es/calidad-e-innovacion/calidad-interna-y-externa-de-producto.html. [Accessed: 25-Jun-2018].

[22] A. Calero, P. Castro, H. Mora, M. Á. Vicedo, and D. García, "Visión innovadora de la calidad del producto software," *REICIS Rev. Española Innovación, Calid. e Ing. del Softw.*, vol. 5, no. 2, pp. 49–55, 2009.

[23] C. M. Pardo Rivera, "Estandares y modelos de calidad del software," *marzo 17*, 2013. [Online]. Available: http://evaluaciondesoftware2013.blogspot.com/. [Accessed: 10-Jul-2018].

[24] P. G. Tello, "Evaluación de Calidad de un Producto de Software," Universidad Nacional de La Plata, 2016.

[25] ISO/IEC 25010, "ISO/IEC 25010 Calidad del Producto de Software." [Online]. Available: http://iso25000.com/index.php/normas-iso-25000/iso-25010?limit=3&limitstart=0. [Accessed: 23-Jun-2018].

[26] A. M. del C. García Oterino, "[Debate] ¿Los testers deben saber hacer de todo en el mundo ágil? ¿Qué cualidades debe tener un tester en un equipo ágil? - Javier Garzás," *2015-05-15*, 2015. [Online]. Available: http://www.javiergarzas.com/2015/05/debate-tester-agil.html. [Accessed: 20-Nov-2018].

[27] Subcomité 7, "NC ISO/IEC 25022 SQuaRE – Medición de la calidad en el uso," *Nov. 23, 2017*, 2017. [Online]. Available: https://subcomite7.cubava.cu/2017/11/23/nc-isoiec-25022-square-medicion-de-la-calidad-en-el-uso/#.W0PtDdIza02. [Accessed: 09-Jul-2018].

[28] ISO25000, "ISO 25040 - Evaluación de un producto software." [Online]. Available: http://iso25000.com/index.php/normas-iso-25000/iso-25040. [Accessed: 26-Nov-2018].

[29] Y. S. Estévez Carvajal and L. A. Esteban Villamizar, "Modelo De Calidad Para Evaluar El Software Desarrollado En El Centro De Investigación Aplicada Y

Desarrollo En Tecnologías De Información CIADTI," *TECKNE 12 de junio 2014*, vol. 12, no. 1, p. 18, 2014.

[30] "Sistemas de Calidad: Cuadro Comparativo de las Normas y Estándares para Proyectos de TI y para Desarrollo de Software.," *23-02-2015*, 2015. [Online]. Available: http://blogsistemasdecalidad.blogspot.com/2015/02/cuadro-comparativo-de-las-normas-y.html. [Accessed: 03-Sep-2018].

[31] "Herramientas de calidad software imprescindibles." [Online]. Available: http://www.javiergarzas.com/2012/03/herramientas-de-calidad-software.html. [Accessed: 25-Jan-2019].

[32] M. Arbenz, *SonarQube In Action*, no. November 2010. 2011.

[33] "Calidad Software - Los 7 ejes de la calidad del código fuente." [Online]. Available: https://www.panel.es/blog/calidad-software-los-7-ejes-de-la-calidad-del-codigo-fuente/. [Accessed: 11-Jan-2019].

[34] C. N. del Ecuador, "LEY DE COMERCIO ELECTRONICO, FIRMAS Y MENSAJES DE DATOS," *Leyes relativas a la PI Adopt. por el Pod. Legis.*, pp. 1–19, 2002.

[35] Asamblea Nacional del Ecuador, "CÓDIGO ORGÁNICO DE LA ECONOMÍA SOCIAL DE LOS CONOCIMIENTOS, CREATIVIDAD E INNOVACIÓN," *Regist. Of.*, vol. IV, no. 899, p. 113, 2016.

[36] Decreto Ejecutivo 1435, "REGLAMENTO CODIGO ORGANICO ECONOMIA SOCIAL DE LOS CONOCIMIENTOS," pp. 1–17, 2017.

[37] SonarQube, "Definiciones métricas," 2018. [Online]. Available: https://docs.sonarqube.org/latest/user-guide/metric-definitions/. [Accessed: 11-Jan-2019].

ANEXOS

Anexo 1. Matriz de Evaluación Calidad Interna del Software

MATRIZ DE CALIDAD INTERNA - EVALUACIÓN								
Característica	**Subcaracterística**	**Métrica**	**Fase del ciclo de vida de calidad del producto**	**Método de aplicación**	**Fórmula**	**Valor deseado**	**Tipo de Medida**	**Software Viáticos valor obtenido**
Mantenibilidad	**Modularidad**	Capacidad de condensación	Interna	Contar el número de componentes que no son afectados por cambios de otros componentes y el número total de componentes específicos	X= B/A	0<=X<=1 El más cercano a 0 es el mejor.	X=Contable/ Contable A=Contable B=Contable	
		Acoplamiento de clases	Interna	Contar el número de relaciones que tiene una función con respecto a otras clases.	X=A	1<=X<=5 El más cercano a 1, es el mejor.	X=Contable A=Contable	
	Reusabilidad	Ejecución de reusabilidad	Interna	Contar el número de elementos reutilizados y el número total de elementos de la biblioteca reutilizable	X = A/B	0<=X<=1 El más cercano a 1, es el mejor	X=Contable/ Contable A=Contable B=Contable	
	Capacidad de ser analizado	Capacidad de pistas de auditoría	Interna	Contar el número de datos realmente grabadas durante la operación y el número de datos previstos a grabarse para controlar el estado del sistema durante la operación	X = A/B	0<=X<=1 El más cercano a 1, es el mejor	X=Contable/ Contable A=Contable B=Contable	

	Código Repetido	Interna	La densidad de código repetido mide la relación entre la cantidad de código repetido de un producto y su tamaño.	DCR=LD /LT *100	1<=X<=100 El más cercano a 1, es el mejor	DCR=Contable/ Contable LD=Contable LC=Contable	
Capacidad de ser modificado	Complejidad ciclomática	Interna	Contar las instrucciones condicionales, bucles, salidas de métodos y clausulas AND y OR dentro de los condicionales.	X= A+1	1<=X<50 El más cercano a 1, es el mejor	X=Contable A=Contable	
	Grado de localización de corrección de impacto	Interna/Externa	Contar el número de fallas aparecidas después que se ha resuelto un fallo y contar el número de fallas resultas	X = A/B	0<=X<=1 El más cercano a 0, es el mejor	X=Contable/Contable A=Contable B = Contable	
Capacidad de ser probado	Capacidad de prueba autónoma	Interna	Contar el número de pruebas que están dependiendo de otros sistemas y contar el número total de pruebas dependientes con otros sistemas	X = A/B	0<=X<=1 El más cercano a 0, es el mejor	X=Contable/Contable A=Contable B=Contable	
TOTAL MANTENIBILIDAD=							