TECNOECUATORIANO
INSTITUTO SUPERIOR TECNOLÓGICO

FOLLETO COMPLEMENTARIO DOCENTE:
CALIDAD DE SOFTWARE

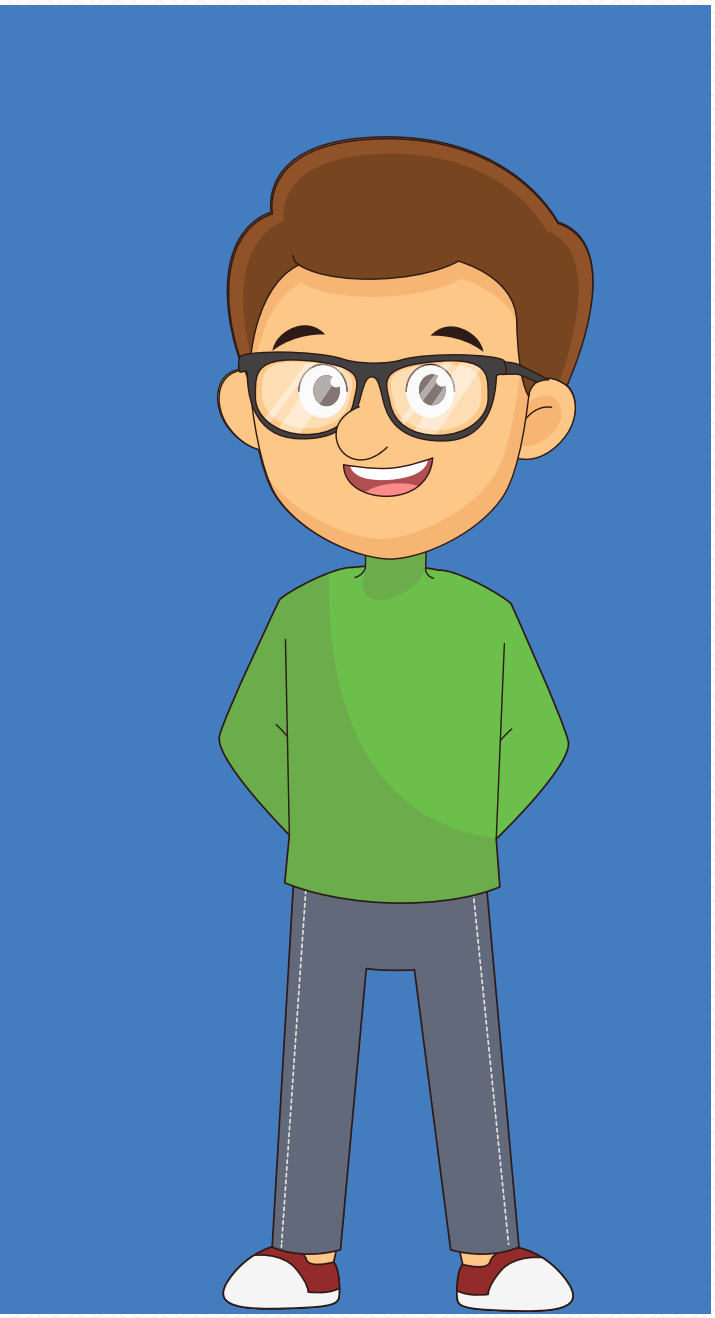


AUTORES

JENNY ELIZABETH
CABASCANGO CALDERÓN

PABLO DANILO
RAMOS PUMAYUGRA

Quito – Ecuador
Septiembre 2024



Folleto complementario docente:
Calidad de software

Jenny Elizabeth Cabascango Calderón
Instituto Superior Tecnológico Tecnoecuatoriano
jcabascango@istte.edu.ec
https://orcid.org/0000-0002-3461-9296

Pablo Danilo Ramos Pumayugra
Instituto Superior Tecnológico Tecnoecuatoriano
pramos@istte.edu.ec
https://orcid.org/0009-0008-3166-8181

Este libro ha sido sometido a revisión de doble par académico:
Ing. Omar Chancusig Maisincho, M.Sc.
Instituto Superior Universitario Cotopaxi
SENESCYT

Ing. Marcelo Quimbita Quimbita
Developers
Desarrollo de Software y Seguridad Informática

Corrección de estilo: Ángel Velásquez Cajas
Diseño y diagramación: Juan Carlos Tapia Calama

Primera Edición
Instituto Superior Tecnológico Tecnoecuatoriano
Rimana Editorial
Quito – Ecuador
Septiembre 2024

ISBN: 978-9942-676-89-4

Agradecimientos

Este folleto instruccional se lo ha realizado con la dedicación y contribuciones de la comunidad educativa que han participado directa o indirectamente en su desarrollo. Los estudiantes han sido el principal motivo, cuya pasión por el aprendizaje y el deseo de superación constante son la verdadera inspiración detrás de este trabajo. Su entusiasmo y curiosidad nos motivan a seguir mejorando cada día.

A todos los docentes y colaboradores, quienes, con su conocimiento especializado y su experiencia en la industria del software, han brindado valiosas ideas y sugerencias para la estructuración y actualización de los contenidos. Su compromiso con la excelencia académica es el pilar fundamental de este proyecto. Un especial agradecimiento a las instituciones y organizaciones que nos han apoyado en la implementación de los estándares de calidad en el desarrollo de software, proporcionando el marco práctico necesario para alinear la enseñanza con las exigencias reales del mercado.

Finalmente, queremos expresar gratitud a los equipos de trabajo que hicieron posible la producción de este material, asegurando que cada detalle esté orientado a facilitar el aprendizaje efectivo. Este folleto es el resultado del esfuerzo colectivo de todos aquellos que creen firmemente en la importancia de la calidad en el desarrollo de software. A todos, gracias por su compromiso y contribuciones.

Los autores

Biografía de los autores

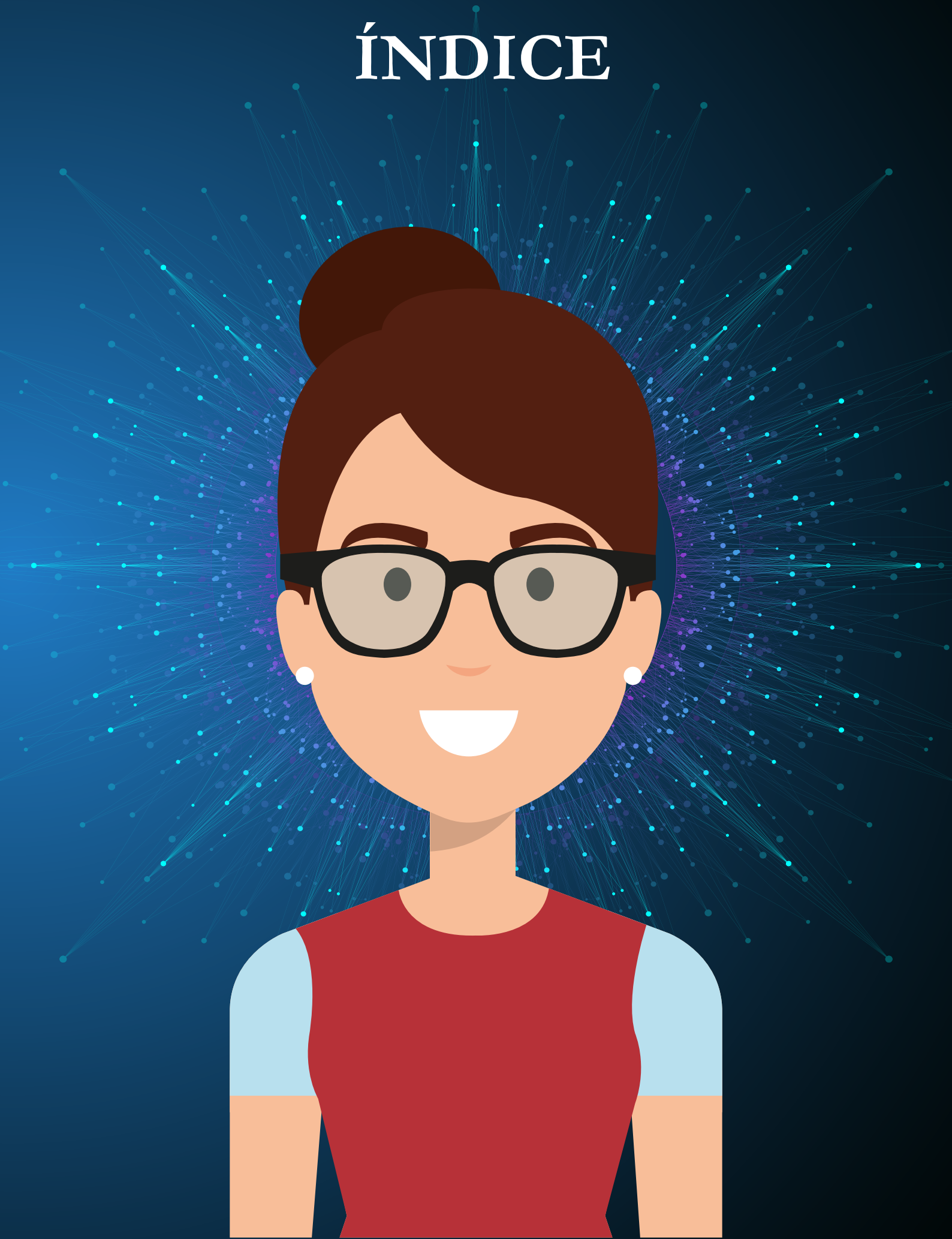
Jenny Elizabeth Cabascango Calderón

Actualmente, se desempeña como Coordinadora de la carrera de Desarrollo de Software en el Instituto Superior Tecnológico Tecnoecuatoriano. Además, la autora ejerce funciones de docencia en asignaturas básicas dentro de la misma carrera. Es Ingeniera en Electrónica y Control, graduada de la Escuela Politécnica Nacional y ha continuado su formación académica en España, con una especialización en Eficiencia Energética por la Universidad de Castilla - La Mancha y una Maestría Universitaria en Energías Renovables en Sistemas Eléctricos por la Universidad Carlos III de Madrid. En el ámbito académico y profesional, ha liderado y colaborado en proyectos de desarrollo de software, enfocados en la automatización de procesos industriales, integración de sistemas energéticos y eficiencia en el consumo energético. También, ha formado parte de equipos multidisciplinarios de investigación, combinando la innovación en software en diversas aplicaciones tecnológicas.

Pablo Danilo Ramos Pumayugra

Es un desarrollador en tecnología de la información con más de 10 años de experiencia en el desarrollo de software y soluciones tecnológicas innovadoras. El autor nació en Quito y ha trabajado en diversas empresas de tecnología, contribuyendo en proyectos que van desde mantenimiento y reparación de elementos de red y cómputo hasta seguridad electrónica. Actualmente, colabora en el Instituto Tecnológico Superior Tecnoecuatoriano; Pablo es un apasionado por el aprendizaje continuo, por ello, ha obtenido varias certificaciones en Redes Cisco, además de certificaciones Hikvision en seguridad electrónica. Este profesional es un ferviente defensor de la inclusión digital y colabora con organizaciones que promueven el acceso a la tecnología en comunidades desfavorecidas. En su tiempo libre, disfruta explorando nuevas tendencias tecnológicas, compartiendo su conocimiento y aprendiendo cada día un poco más. Su lema es: “la tecnología debe ser una herramienta para todos”.

ÍNDICE



ÍNDICE DE CONTENIDOS

Prólogo 11

12

CAPÍTULO I

Calidad de software

Definiciones de calidad. 13

Factores de calidad. 13

Factores de calidad según McCall. 14

Factores de calidad según Boehm. 16

Modelos y normas de calidad. 16

Normas y definiciones. 18

21

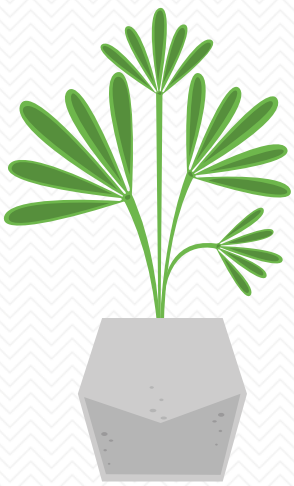
CAPÍTULO II

Estándares y métricas de calidad en la ingeniería de software

Estándares. 22

Factores de calidad. 22

Técnicas de revisión. 22

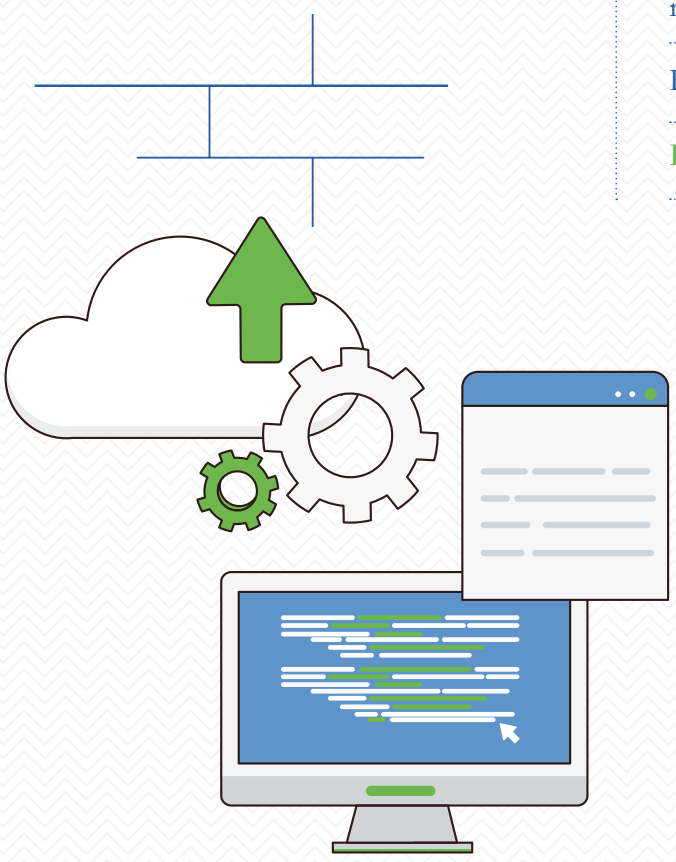


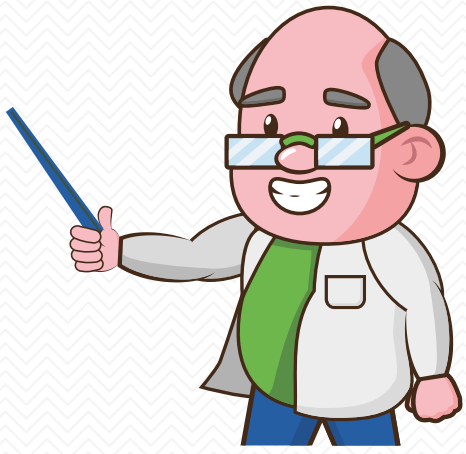
26
CAPÍTULO III
Aseguramiento de la Calidad del Software (ACS)

Elementos del Aseguramiento de la Calidad del Software (ACS). 27
Procesos de aseguramiento de la calidad. 27
Métricas de Aseguramiento de la Calidad del Software (ACS). 28
Enfoques formales al ACS. 29
Confiabilidad de software. 30
Las normas de calidad ISO 9000. 30

32
CAPÍTULO IV
Estrategias de prueba de software

Aspectos estratégicos para la prueba del software. 33
Pruebas de software. 33
Ejercicios de aplicación. 34
Ejemplo 1: Software para el control de inventario de un comercio pequeño. 34
Ejemplo 2: Sistema de gestión de inventarios para una cadena de suministro. 39
Ejemplo 3: Plataforma de monitoreo de sostenibilidad para la gestión de recursos hídricos. 41
Recomendaciones finales. 44
Referencias. 49





ÍNDICE DE FIGURAS

Figura 1

Perspectivas de los elementos de calidad según McCall. 14

Figura 2

Factores de calidad según Boehm. 16

Figura 3

Representación de la estructura de la norma ISO con el ciclo PHVA. 17

Figura 4

Factores ISO 9126. 20

Figura 5

Enfoques de Gestión de la Calidad. 30

Figura 6

Pruebas realizadas al software desarrollado por estudiantes de la carrera de Desarrollo de Software del Instituto Superior Tecnológico Tecnoecuatoriano. 34

Figura 7

Pantalla de inicio. 36

Figura 8

Menú disponible del programa. 37

Figura 9

Ejemplo del ingreso de nuevos clientes. 37

Figura 10

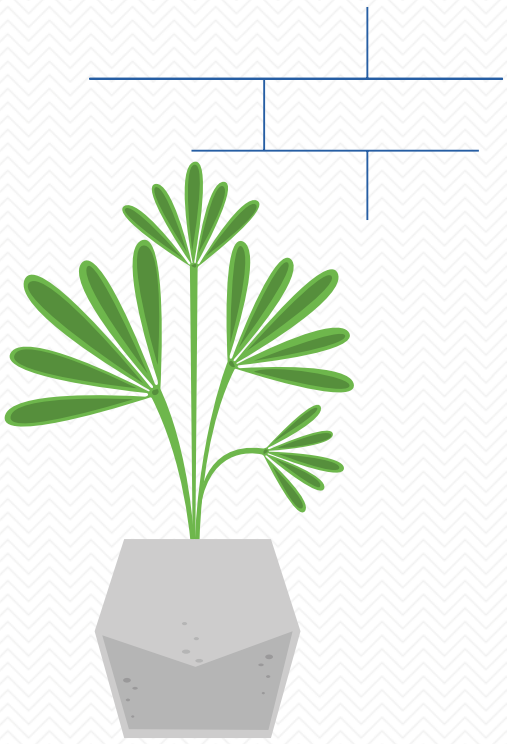
Ejemplo de una consulta de "Historial de Ventas". 38

Figura 11

Características del software al aplicar ISO/IEC 25010. 40

Figura 12

Modelo de calidad ISO/IEC 25010 aplicado en el ejemplo 3. 42



ÍNDICE DE TABLAS

Tabla 1

Desarrollo de ejemplo 1. 34

Tabla 2

Desarrollo del ejemplo 2. 39

Tabla 3

Desarrollo del ejemplo 3. 41

Tabla 4

Resumen de certificaciones en el tema de calidad de software. 48

Figura 13

Modelo según la metodología CMMI para el ejemplo 3. 42

Figura 14

Aplicación de las metodologías ágiles en el ejemplo 3. 43

Figura 15

Logo de la certificación CSQA. 44

Figura 16

Logo de la Certificación ISTQB®. 45

Figura 17

Logo de la certificación CSQM. 45

Figura 18

Logotipo de la certificación ISO 9001 Lead Auditor. 46

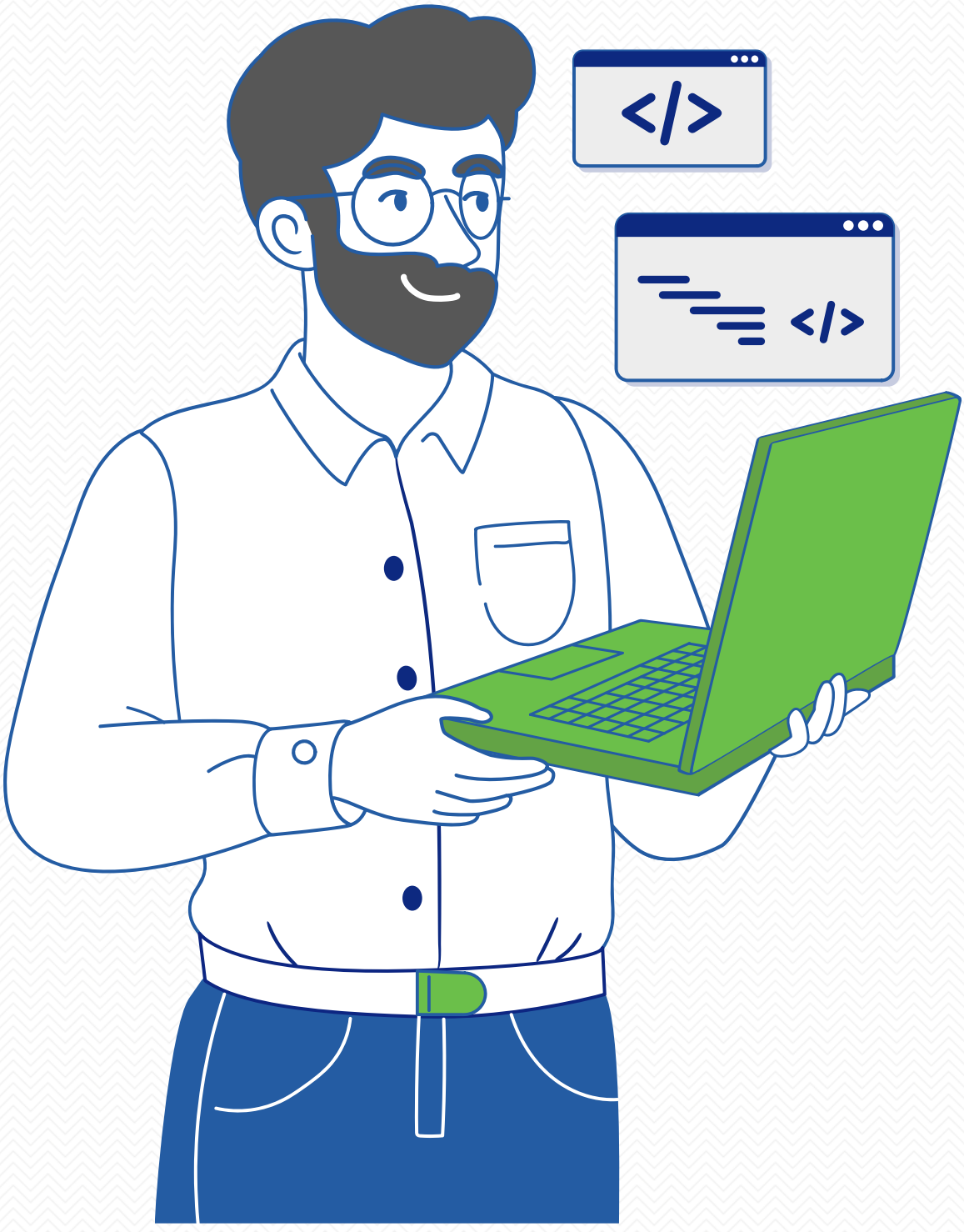
Figura 19

Logos representativos según las metodologías de las academias que emiten las certificaciones. 47

Figura 20

Logo de la certificación CATF a nivel fundamental para iniciar con las metodologías ágiles. 47

PRÓLOGO



Prólogo

El desarrollo de software va de la mano con la actualización de tecnologías. Así que, siempre existe una evolución en sus conceptos y mejoras en su implementación. Esto es impulsado por la creciente demanda de soluciones tecnológicas confiables y eficientes. En este contexto, la calidad de software ha emergido como un pilar fundamental para garantizar que los productos desarrollados no solo cumplan con los requisitos técnicos. Por lo tanto, se debe recordar que tanto el desarrollador y el cliente tienen una estrecha comunicación para que el producto final sea de calidad, donde se asegure su usabilidad y seguridad.

Este folleto instruccional ha sido diseñado como una guía base para la asignatura de calidad de software, con el objetivo de proporcionar a los estudiantes una comprensión profunda de los principios, técnicas y estándares que subyacen en la creación de software de alta calidad. Los contenidos están estructurados en unidades didácticas que tratan de llegar a los conocimientos tanto de fundamentos teóricos como las aplicaciones prácticas de los conceptos de calidad. De esta manera, cubre un amplio espectro de temas que van desde las definiciones básicas de calidad hasta las estrategias avanzadas de pruebas y auditorías de software.

La asignatura se organiza en cuatro unidades principales:

Unidad 1: Introducción - Conceptos, donde se exploran las definiciones clave de calidad y los principales modelos y normas utilizados en la industria.

Unidad 2: Técnicas de revisión, que introduce a los estudiantes en el uso de métricas, revisiones informales y revisiones técnicas formales, siendo este último un proceso de evaluación.

Unidad 3: Aseguramiento de la Calidad de Software (SQA), que abarca los elementos críticos del SQA, sus metas y métricas, así como su confiabilidad y las normas ISO 9000.

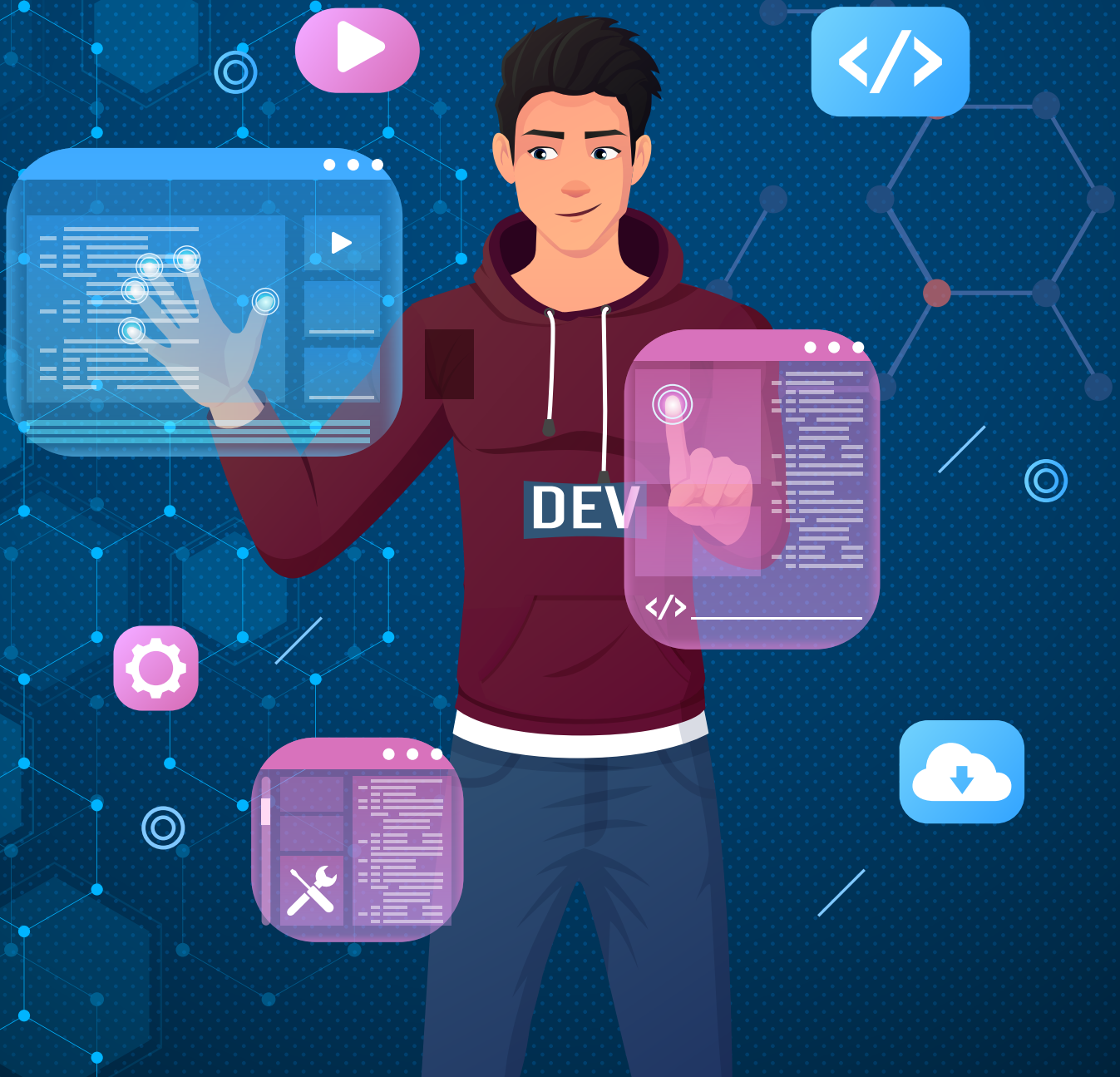
Unidad 4: Estrategias de Prueba del Software, centrada en las estrategias fundamentales que aseguran que el software (producto final) sea probado de manera exhaustiva y estratégica.

El aprendizaje práctico complementa los contenidos teóricos, permitiendo a los estudiantes aplicar sus conocimientos en situaciones reales. Entre estas actividades, se incluyen la revisión del código y su funcionalidad, la automatización de pruebas, la implementación de estándares de calidad y la auditoría de software.

En este sentido, este material está diseñado no solo para proporcionar conocimientos técnicos, sino también para fomentar en los estudiantes una mentalidad crítica y orientada a la mejora continua, tan necesaria en un entorno de desarrollo de software que evoluciona rápidamente. Es el deseo que este folleto no solo sea una herramienta útil durante el curso, sino también una referencia continua para el ejercicio profesional en el área de calidad de software.

CAPÍTULO I

CALIDAD DE SOFTWARE



Definiciones de calidad

La calidad se define como la habilidad de un objeto para satisfacer requerimientos, ya sean explícitos o implícitos, conforme a determinados estándares o requisitos. Este concepto posee un elemento subjetivo, dado que se basa en las percepciones personales, que fluctúan de acuerdo a elementos como la cultura, el tipo de producto o servicio, las expectativas y requerimientos de cada individuo.

El concepto de calidad se deriva del término latino qualitates o qualitatis. En un escenario más general, el término puede referirse, por ejemplo, a la calidad de vida de una nación, que se evalúa con base en el acceso a bienes y servicios fundamentales. Igualmente, se evalúa la calidad del agua o del aire a través de la comparación con parámetros ideales o en comparación con las normas de otras naciones.

La calidad de los servicios proporcionados por una compañía, generalmente, se relaciona con la satisfacción del cliente. Mientras que la calidad de un producto, usualmente, se relaciona con sus atributos, tales como su durabilidad y funcionalidad. Respecto a bienes y servicios, el concepto de calidad puede ser interpretado de diversas maneras, tales como la satisfacción de las necesidades del cliente, la generación de valor añadido o la correlación entre el costo y el beneficio (ISO, 2015). En el campo del marketing, una visión moderna de la calidad sugiere que no se trata solo de entregar lo que el cliente espera, sino de sorprenderlo con algo que ni siquiera sabía que necesitaba, pero que una vez que lo recibe, reconoce esto como esencial.

Por otro lado, conceptos como control de calidad, garantía de calidad y gestión de calidad son fundamentales en la industria y los servicios. Estos se gestionan a través de diversos indicadores, como las normas internacionales de calidad, tales como ISO 9000 e ISO 14000, que han sido establecidas por la Organización Internacional de Normalización (ISO) desde 1947 (López, 2014).

Factores de calidad

Existen dos grandes grupos de factores que influyen en la calidad del software que se describen a continuación:

Factores que pueden medirse de forma directa, como, por ejemplo, la cantidad de errores detectados por cada mil líneas de código (KLOC) o la cantidad de fallos que ocurren dentro de una línea de tiempo específica. Estos factores obtienen datos objetivos y cuantificables, lo que facilita el seguimiento continuo del rendimiento de su calidad en el software a lo largo del desarrollo y mantenimiento.

Factores que requieren medición indirecta, tales como la usabilidad del sistema o la fluidez del mantenimiento. Estos aspectos dependen más de la experiencia del usuario final y del equipo técnico y no siempre se pueden cuantificar con precisión. Para evaluarlos, a menudo se utilizan técnicas cualitativas, como encuestas, entrevistas o análisis de experiencias durante la interacción con el sistema.

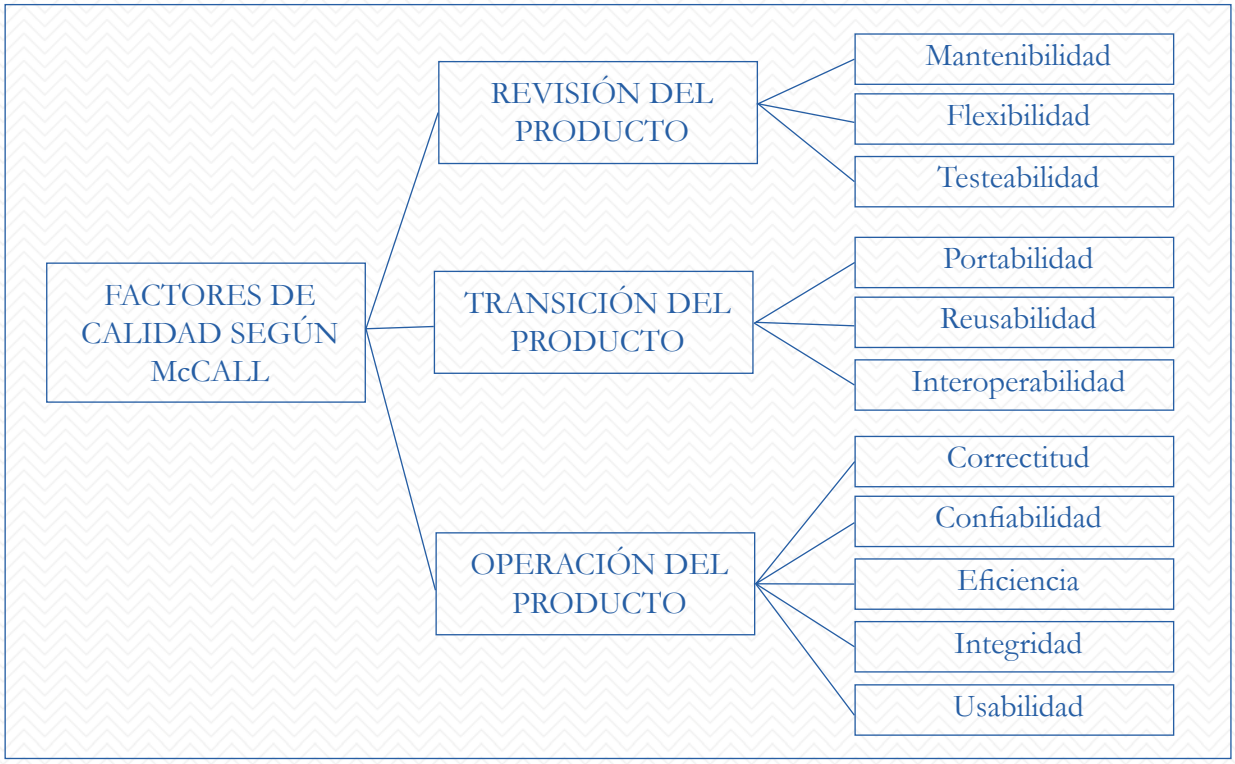
Para estimar la calidad del software en su totalidad, es necesario comparar el rendimiento y las características del mismo con estándares o modelos de referencia previamente definidos. Esto permite obtener una evaluación más completa que abarque tanto los aspectos técnicos como los relacionados con la experiencia del usuario, proporcionando una visión global del nivel de calidad alcanzado.

Factores de calidad según McCall

El tipo de calidad del software de McCall, propuesto por el ingeniero estadounidense Jim McCall en la década 70, es el pionero y más reconocido modelo para evaluar la calidad en los productos de software. El objetivo principal del modelo es proporcionar una guía que permita a los desarrolladores, testers y usuarios evaluar y garantizar que el software cumpla con ciertos criterios de calidad de manera integral. El modelo de McCall se centra en tres ejes clave para evaluar la calidad del software como de describe en la Figura 1.

Figura 1

Perspectivas de los elementos de calidad según McCall.



Fuente: autoría propia.

Los factores de calidad de McCall están divididos en 11 elementos clave agrupados en tres grandes categorías: comprobación del producto, transformación del producto y ejecución del producto. Estos factores sirven como una métrica para evaluar qué tan bien un software cumple con los requisitos de calidad (Mejía, 2024).

Revisión del producto. Esta categoría está enfocada en la aptitud del software de ser entendido, revisado y modificado por los desarrolladores y el equipo de mantenimiento.

- Mantenibilidad (maintainability): refleja la facilidad con que el software puede ser corregido, modificado y mejorado para satisfacer nuevos requisitos o para corregir errores que puedan surgir durante su vida útil.

- Flexibilidad (flexibility): es la capacidad de evolucionar a partir de nuevas circunstancias o requisitos sin necesidad de cambios estructurales esenciales.

- Testeabilidad (testability): es la facilidad con la que se pueden hacer pruebas para detectar errores en el software. Un software fácil de probar es más probable que tenga problemas identificados y, por lo tanto, es más probable que pueda abordarse de manera efectiva.

Transición del producto. Esta categoría aborda la adaptabilidad del software de adecuarse a diferentes entornos tecnológicos y operacionales.

- Portabilidad (portability): mide la capacidad del software para correr en diferentes plataformas, sistemas operativos o entornos sin requerir cambios significativos.

- Reusabilidad (reusability): indica qué tan bien pueden reutilizarse los componentes o tipos del software en otros proyectos o sistemas. La reusabilidad promueve la eficiencia al reducir la necesidad de escribir código desde cero.

- Interoperabilidad (interoperability): en términos de interoperabilidad, esta categoría evalúa la capacidad del software para interactuar o funcionar conjuntamente con otros sistemas o programas. Operación del producto. La interoperabilidad se refiere a si el software puede integrarse y comunicarse eficazmente con otros sistemas. Esta categoría mide cómo se comportó realmente el software desde el punto de vista del usuario. Los indicadores incluidos en esta clasificación evalúan la experiencia del usuario al interactuar con este o aquel software.

- Correctitud (correctness): mide el grado en que el software asiste con su funcionalidad especificada. Un sistema es correcto si realiza las funciones para las que fue diseñado de manera precisa y libre de errores.

- Confiabilidad (reliability): indica la capacidad del software para mantener su desempeño y ofrecer resultados correctos, bajo condiciones operacionales específicas, durante un periodo de tiempo definido. Un software fiable es capaz de mantenerse alejado de errores y evitar fallos críticos.

- Eficiencia (efficiency): evalúa el uso óptimo con los recursos, como el tiempo de procesamiento, la memoria y la capacidad de almacenamiento. Un software eficiente maximiza el rendimiento utilizando la menor cantidad de recursos disponibles.

- Integridad (integrity): es la capacidad del software de protegerse contra accesos no autorizados o daños, garantizando la seguridad de los datos y del sistema.

- Usabilidad (usability): mide la sustentabilidad con la que los usuarios pueden aprender a utilizar el software y operarlo de manera efectiva. Un software con alta usabilidad es intuitivo.

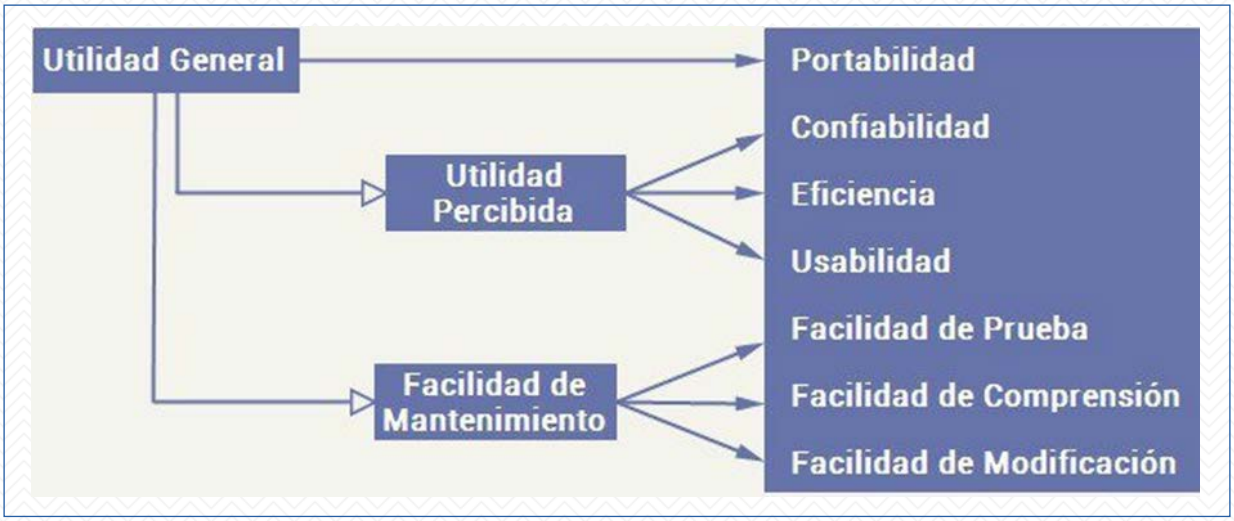
La relevancia del modelo de McCall reside en su uso extenso en el sector del desarrollo de software, gracias a su enfoque holístico para asegurar la calidad en todas las fases del ciclo de vida del software. Este ofrece un esquema que posibilita a los equipos de desarrollo valorar tanto elementos internos como externos de la calidad del software, garantizando que el producto final sea sólido, fiable y sencillo de conservar. El uso de estos elementos en la valoración y mejora del software también influye directamente en la satisfacción del cliente, dado que, al garantizar la exactitud, confiabilidad y usabilidad del producto, se reducen los errores y se asegura una experiencia de usuario positiva.

Factores de calidad según Boehm

El modelo sugerido por Boehm establece una jerarquía de funciones que aportan a la calidad global del software. Dentro del modelo de McCall, también se pueden hallar gran cantidad de los elementos contemplados en el de Boehm. En la Figura 2 se detalla algunas de las estructuras de este último, poniendo especial atención en los elementos fundamentales que lo conforman. En total, el modelo de Boehm presenta siete factores principales que influyen en la calidad del software. Este modelo se enfoca en analizar cómo estos factores específicos impactan la calidad global y la efectividad del software.

Figura 2

Perspectivas de los elementos de calidad según McCall.



Fuente: tomado de Zimbrón (2017).

Modelos y normas de calidad

Los modelos de calidad son herramientas importantes que las organizaciones utilizan para mejorar la gestión interna de procesos y productos. A diferencia de las normas, los modelos no establecen requisitos estrictos que un sistema de gestión de calidad debe cumplir. En cambio, estos proporcionan pautas que ayudan a las organizaciones a mejorar continuamente. Es así que existen muchos modelos diferentes, algunos centrados en la calidad general y el conocimiento, otros en la mejora de procesos específicos o la adaptación a sectores industriales específicos.

Muchas veces, las organizaciones desarrollan sus propios modelos internos de calidad. La principal diferencia entre los modelos y las normas de calidad es que, si bien normas como la ISO 9001 establecen requisitos formales para la certificación, los modelos ofrecen un enfoque más flexible. Estos pueden servir como referencia para mejorar la gestión sin ceñirse a normas estrictas. Modelos como el de Excelencia EFQM se centran en la autoevaluación y la mejora continua, mientras que normas como ISO 9001 se centran más en la certificación formal y la auditoría. En este contexto, existen varios modelos de calidad:

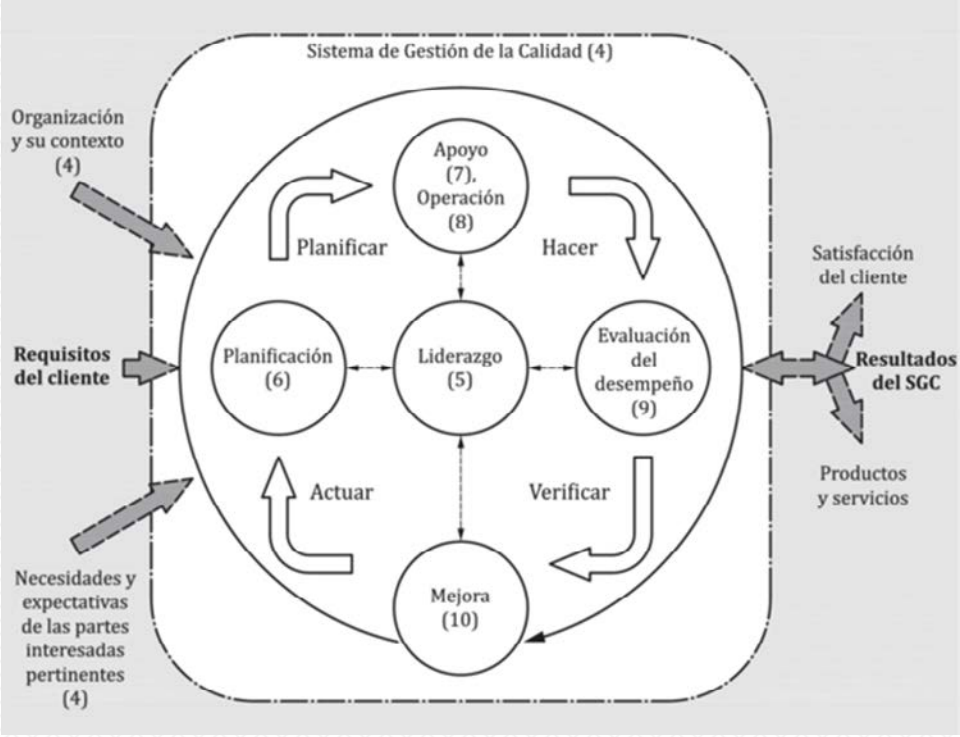
Modelos orientados a la excelencia. Un ejemplo es el modelo EFQM, que ayuda a las organizaciones a comprender sus fortalezas y áreas de mejora para alcanzar la excelencia organizacional. El modelo evalúa a las organizaciones desde una perspectiva integral, tomando en cuenta factores como el liderazgo, la estrategia, la gestión de recursos humanos y los resultados.

Modelos industriales. Algunas industrias específicas han desarrollado sus propios modelos de calidad para satisfacer sus necesidades específicas. Por ejemplo, en la industria automotriz existen modelos como el IATF 16949, que se centran en la calidad de fabricación de los automóviles y sus piezas.

Modelo de mejora continua. Un ejemplo clásico es el ciclo PHVA (Planificar, Hacer, Verificar, Actuar), también conocido como ciclo de Deming. Este modelo se centra en lograr la mejora continua mediante la realización de ciclos repetidos de operaciones y el ajuste de los procesos en función de los resultados obtenidos (ISO, 2015).

Figura 3

Representación de la estructura de la norma ISO con el ciclo PHVA.



Fuente: tomado de ISO (2015).

Modelos de desarrollo organizacional en sí mismos. Muchas empresas crean sus propios modelos de calidad basados en sus valores, cultura y necesidades. Estos modelos les permiten adaptar las propuestas de mejora a sus circunstancias y crear sistemas de gestión más personalizados. Actualmente, la tendencia de implementar modelos de gestión de calidad tanto en el sector público como privado es clara. Esto permite cubrir la necesidad de desarrollar propuestas estructuradas para la mejora continua de los productos y servicios ofrecidos. La implementación de estos modelos, no solo asegura el cumplimiento de estándares de calidad reconocidos globalmente, sino que también aumenta la competitividad en el mercado globalizado (Lasso, 2021).

Los modelos de calidad también tienen implicaciones estratégicas importantes. Al proporcionar una estructura clara y un enfoque objetivo para analizar y mejorar la gestión organizacional, permiten a las empresas identificar áreas de mejora. Esto es necesario para crear un proceso de mejora continua que no sólo asegure la supervivencia de la empresa en un entorno competitivo, sino que también ayude a la empresa a destacarse en el mercado. Al aplicar modelos de calidad, las organizaciones pueden:

- Evitar la necesidad de crear indicadores nuevos, ya que estos están predefinidos dentro del propio modelo de calidad.

- Ofrecer un marco conceptual integral, que permite a las organizaciones entender mejor sus procesos y áreas de mejora.

- Establecer objetivos y estándares uniformes para todos los procesos; esto facilita comparaciones consistentes ya que estos estándares han sido validados en diversos contextos.

- Organizar de manera coherente las actividades de mejora, permitiendo una estructura clara para implementar cambios de forma ordenada y eficiente.

- Facilitar la medición con criterios consistentes a lo largo del tiempo, lo que permite a la organización identificar claramente si está avanzando hacia sus metas de mejora continua.

Además, estos modelos favorecen la comparación con otras organizaciones, lo que promueve el intercambio de buenas prácticas y experiencias entre industrias.

Normas y definiciones

La Organización Internacional de Normalización (ISO) es una federación mundial de organismos nacionales de normalización (organismos miembros de la ISO). Los comités técnicos de la ISO suelen ser responsables del desarrollo de normas internacionales. Los comités nacionales miembros interesados en un tema en particular, tienen derecho a participar en el comité técnico creado para tal efecto. En esta labor, también participan organizaciones internacionales, tanto gubernamentales como no gubernamentales, que cooperan con la ISO. Adicionalmente, esta entidad coopera estrechamente con la Comisión Electrotécnica Internacional (IEC) en materia de normalización electrotécnica.

Como punto inicial, la publicación en 1987 de las Normas Internacionales de la serie ISO 9000 obedeció a exigencias básicas de los programas genéricos de gestión de calidad. Estas normas

están redactadas en términos genéricos y son igualmente aplicables a empresas de servicios tales como bancos, hospitales, hoteles y restaurantes. Dichas normativas se desarrollaron principalmente para ser usadas dentro de las empresas y en las relaciones entre comprador y vendedor. Esta última aplicación ayudaba a que las empresas tengan la posibilidad de evaluaciones múltiples. En cierto número de países, la práctica de confiar la evaluación de sistemas de calidad de proveedores a organismos terceros se ha adoptado rápidamente.

En la actualidad, no existe ningún mecanismo de la ISO que rija el reconocimiento mutuo de certificados de registro, emitidos por organismos terceros a empresas cuyos sistemas de calidad han sido evaluados. Sin embargo, para promover la convergencia entre las normas nacionales, el Comité del Consejo para la evaluación de la conformidad (ISO/ CASCO) ha preparado y publicado la colección de guías ISO/CEI. Algunas de estos textos son aplicables directamente, como es el caso de la guía ISO/CEI 40, "Requisitos Generales para la Aceptación de Organismos de Certificación", o la guía ISO/CEI 48, "Requisitos para la Evaluación y el Registro por Terceros del Sistema de Calidad de un Proveedor". Los dos documentos anteriormente mencionados han sido adoptados en muchos países, tanto en sus reglamentos sobre programas de certificación como en normas nacionales o en normas regionales, como es el caso de las Normas Europeas de la serie EN 45000 de CEN/CENELEC.

Por otro lado, cabe recalcar que los certificados ISO 9000 no se entregan en nombre de la ISO, sino que se ha considerado necesario que esta juegue un papel en la diseminación de información sobre los programas nacionales que operan en los países miembros. En este sentido, la serie ISO 9000 es la norma internacional equivalente para los sistemas de calidad que está disponible en una serie de documentos. Aparte de algunas diferencias sobre la forma en que se expresa, el contenido de la serie ISO 9000 es igual a muchas otras normativas, como la BS 5750. Estos equivalentes no son coincidencia. La serie ISO 9000 se modeló sobre la BS 5750, que fue la pionera de los sistemas de calidad a nivel internacional.

Factores de calidad según ISO 9126. La ISO 9126 es una norma que define un marco para evaluar la calidad del software, enfocándose en diferentes aspectos que contribuyen a su buen desempeño. En esta norma, se identifican varios factores de calidad que son características clave que permiten medir la calidad de un sistema o software.

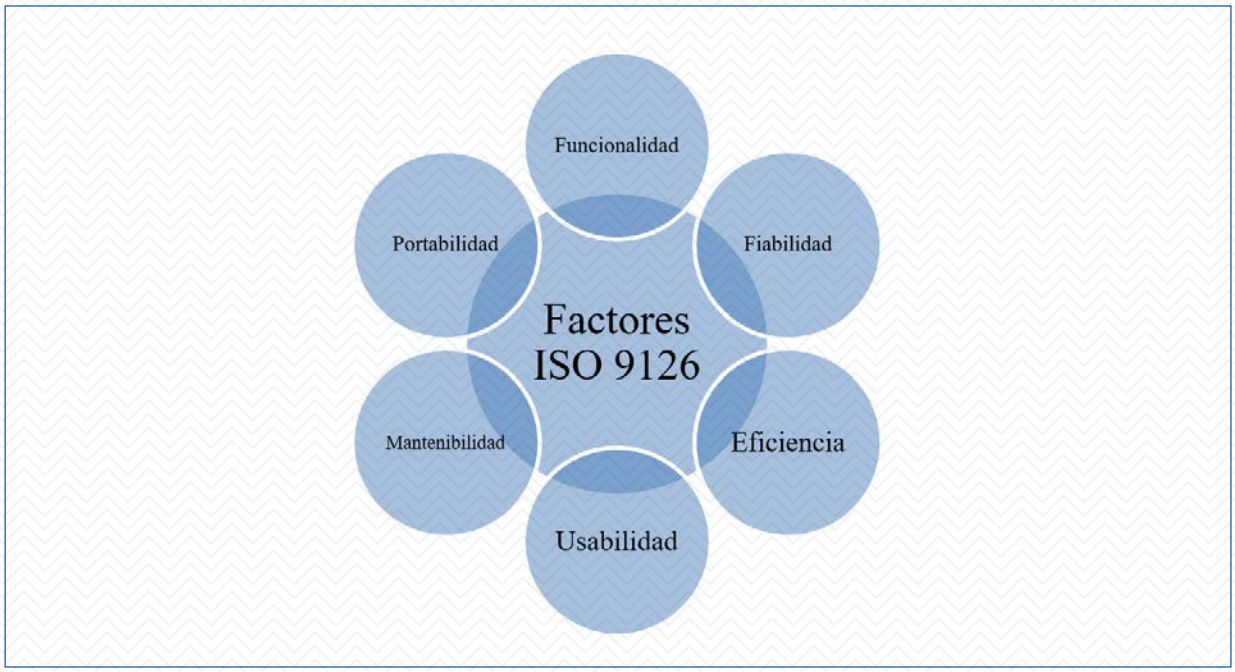
- Funcionalidad: se refiere a la capacidad del software para cumplir con los requisitos especificados, es decir, qué tan bien realiza las funciones para las que fue diseñado. Esta incluye sub-características como adecuación, exactitud, interoperabilidad, cumplimiento y seguridad.
- Fiabilidad: mide la capacidad del software para mantener su rendimiento bajo condiciones específicas durante un período de tiempo determinado. Esta incluye sub-características como madurez, tolerancia a fallos y recuperabilidad.
- Usabilidad: refleja cuán fácil y accesible es el software para los usuarios, tanto en términos de aprendizaje como de uso. Sus sub-características incluyen la comprensión, operabilidad y atractivo visual.
- Eficiencia: evalúa cómo el software utiliza los recursos del sistema (como tiempo de procesamiento y memoria) en relación con la cantidad de trabajo realizado. Las sub-características son el comportamiento en tiempo, utilización de recursos y capacidad de adaptación.
- Mantenibilidad: mide cuán fácil es para los desarrolladores mantener el software, como corregir defectos, realizar mejoras o adaptarlo a nuevas condiciones. Esta incluye sub-características como analizabilidad, modificabilidad, estabilidad y facilidad de prueba.

- Portabilidad: evalúa la capacidad del software para ser transferido y adaptado a diferentes entornos o plataformas. Las sub-características son la capacidad de adaptación, instalación, conformidad y reemplazo.

En la Figura 4, se grafican cada uno de los factores ISO 9126.

Figura 4

Factores ISO 9126.



Fuente: autoría propia.

CAPÍTULO II

ESTÁNDARES Y MÉTRICAS DE CALIDAD EN LA INGENIERÍA DE SOFTWARE

Estándares

Los estándares de calidad de software son normas emitidas por organismos específicos, que se utilizan como un marco de referencia de medición, para saber si un proceso de desarrollo es de calidad. Las normas de calidad del software más conocidas son las que ha elaborado ISO y son la serie ISO-9000.

ISO 9000. Las normas ISO son un estándar de calidad aplicable a cualquier industria. La ISO-9003 tiene relación directa con el desarrollo de un software. Una normativa ISO-9003 es un conjunto de cláusulas o puntos que se deben lograr por medio del trabajo y vida del proyecto y las actividades en torno al proyecto.

CMMI *(Capability Maturity Model Integration).* Este fue desarrollado por el Software Engineering Institute en Estados Unidos y sirve para comprobar la habilidad de los procesos de las organizaciones para realizar determinados proyectos.

SPICE *(Software Process Improvement and Capability Determination).* También conocido como ISO/IEC 15504, es un estándar internacional que proporciona un marco para la evaluación y mejora de los procesos de desarrollo de software. Este es el modelo de madurez propuesto por ISO, similar a CMMI.

Factores de calidad

Los factores de calidad sirven para descomponer el concepto genérico de calidad y facilitar su control y su medición. Se clasifican en:

- Factores operativos: son aquellos que afectan al uso del software.
- Factores de mantenimiento: son aquellos que se aplican a la capacidad de modificación del software.
- Factores evolutivos: son aquellos que indican si el software se puede trasladar con facilidad a otra máquina o a otro producto de base (SO, SGBD).

Técnicas de revisión

La calidad es un concepto relativo y multidimensional, referido a las expectativas y cualidades solicitadas por el cliente. A su vez, está ligado a restricciones y compromisos (presupuesto, tiempo de desarrollo, entre otros). Sin embargo, existe algo que nadie puede negar: cuando algo es de calidad suele pasar desapercibido, pero, al contrario, la mala calidad es algo que destaca negativamente.

Durante el proceso de desarrollo de una aplicación, es muy probable la aparición de bugs causados por fallos humanos. Además, las exigencias del usuario cada vez se incrementan, exigiendo una mejor performance de las aplicaciones creadas. ¿Cómo se puede mejorar la calidad de nuestras aplicaciones? Justamente, para producir software de calidad y cumplir con las expectativas de los usuarios es que existen diferentes métricas de calidad de software. Estas, son un conjunto de medidas utilizadas para estimar la calidad de un proyecto a desarrollar y que permiten comparar o planificar

estas aplicaciones.

Las métricas son de aplicación compleja, ya que no existe una precisión absoluta para determinar cuáles son los elementos o procedimientos relevantes que deben ser aplicados a la hora de escribir código. Por lo tanto, es necesario definir las acciones a tomar, dependiendo de los resultados de estas métricas. Adicional, estas mediciones del producto son una medida cuantitativa que permiten a la gente del software tener una visión profunda de la eficacia del proceso del software y de los proyectos que dirigen, utilizando el proceso como un marco de trabajo, es decir, que son analizadas y evaluadas por los administradores del software. Entre las ventajas de su uso se tienen:

- Determina la calidad del producto.
- Evalúa la productividad de los desarrolladores.
- Se tiene conocimiento cuantitativo de las características del proceso y del producto.
- Se tiene un soporte para la estimación y la planificación.
- Se evalúan los beneficios (en cuanto a calidad y productividad) derivados del uso de nuevos métodos y herramientas de ingeniería del software.
- Establece una línea base para la estimación.

En este sentido, las métricas deben cumplir ciertas características para su funcionalidad y operabilidad:

- Exactas y precisas: no se debe perder información en los redondeos, ya que la información se desvirtúa.
- Consistentes: una medición de un atributo debe dar el mismo valor independientemente de la medición.
- Validez: las métricas deben medir lo que realmente se pretende medir. Es decir, la métrica debe ser relevante y estar alineada con los objetivos del proyecto o proceso.
- Fiabilidad: las métricas deben ser capaces de producir resultados consistentes cuando se aplican en las mismas condiciones, lo que garantiza la confianza en los datos recogidos.
- Accesibilidad: las métricas deben ser fáciles de entender y accesibles para todas las partes interesadas. Esto asegura que los resultados puedan ser interpretados correctamente y utilizados para la toma de decisiones.
- Sensibilidad: las métricas deben detectar cambios pequeños o grandes en las características del software, lo que permite identificar mejoras o deterioros de manera oportuna.
- Comparabilidad: las métricas deben permitir la comparación entre diferentes proyectos, versiones de software o periodos de tiempo. Esto facilita la evaluación de mejoras o problemas a lo largo del tiempo.
- Objetividad: las métricas deben estar basadas en hechos y datos medibles, evitando que las interpretaciones subjetivas influyan en los resultados.
- Eficiencia: las métricas deben ser prácticas de obtener y no deberían requerir grandes recursos o tiempo para su cálculo, especialmente si son utilizadas de manera recurrente.

Para ello, existen diferentes herramientas que ayudan en la tarea de mejorar el desarrollo de las aplicaciones. Como lenguaje de programación interpretado está el Ruby. Este es de alto nivel, orientado a objetos y utiliza herramientas como Rubocop, Rubicritic, Bullet, que ayudan a optimizar el código. Pero lo que hacen estas herramientas no es magia. A continuación, se describen algunas de las principales métricas utilizadas por estos instrumentos para determinar la calidad del código producido.

Acoplamiento *(coupling).* Esta herramienta se refiere al nivel de "conectividad" de un módulo con otros módulos, datos globales y entorno exterior. Durante el desarrollo de la aplicación, uno de los objetivos es mantener una baja dependencia o acoplamiento. Esto quiere decir que, un módulo debe ser capaz de interactuar con otro a través de una interfaz estable y sin depender de otros, para su correcta implementación. Por el contrario, un sistema con un nivel de acoplamiento alto tendrá módulos que se encontrarán inter ligados entre sí y se comunicarán directamente unos con otros sin ningún tipo de barrera, por lo que el código será difícil de comprender, reutilizar y al no estar aislados, se deben considerar todas sus dependencias.

Un problema común con este tipo de aplicaciones es que, en caso de un cambio en algún módulo, esto puede desencadenar un efecto dominó en otros módulos, por lo que será necesario rehacer código en el resto de los módulos dependientes. Un código acoplado va directamente en contra del principio Tell, Don't Ask (TDA), que promueve el encapsulado conjunto de los datos y de las funciones que operan sobre estos datos.

Cohesión *(cohesion).* Esta define el grado de relación que existe entre los elementos de un módulo. Aquellos que sigan el Principio de Responsabilidad Única o SRP por sus siglas en inglés, deben realizar una única cosa. Si no se presta atención a esto, es muy habitual que se tengan clases de elementos realizando varias responsabilidades lógicas a la vez. Cuando se tienen clases con baja cohesión implica, entre otras cosas, falta de comprensibilidad, difícil mantención y difícil reutilización de código.

Una de las métricas más importantes en el desarrollo es la **complejidad ciclomática**, que puede ser usada en las fases de desarrollo o mantenimiento. Esta métrica, propuesta por Thomas McCabe en 1976, se basa en el diagrama de flujo determinado por las estructuras de control de un código en concreto. Del análisis de esta estructura, se obtendrán medidas cuantitativas que facilitarán su comprensión y mejora. La complejidad ciclomática está fuertemente relacionada a un algoritmo claro y eficaz que, a su vez, se relaciona a otro tipo, **la complejidad cognitiva.** Es así que existen diferentes prácticas que ayudan a bajar esta complejidad cognitiva, más usada y funcional para los desarrolladores, son los comentarios en el código.

La complejidad cognitiva mide que tan difícil es entender intuitivamente un bloque de código, a diferencia de la complejidad ciclomática, que determina qué dificultad tiene probar el código. Algunos estudios han probado una correlación directa entre la complejidad ciclomática y la cantidad de bugs de un trozo de código, o el número de líneas de este. Por esto, se concluye que, a mayor complejidad y dimensión, se presentan mayores problemas y fallos en el código.

Revisiones informales. Una verificación de escritorio simple o una reunión casual realizada con un colega constituye una revisión. Sin embargo, como no hay una planeación o preparación por adelantado, ni agenda o estructura de la reunión y no se da seguimiento a los errores descubiertos, la eficacia de tales revisiones es mucho menor que la de los enfoques más formales.

Revisiones técnicas formales. Es una actividad del control de calidad del software realizada por ingenieros de software y otras personas con las cuales se pretende: descubrir los errores en funcionamiento, lógico o implementación, verificar que el software que se revisa cumple sus requerimientos, garantizar que el software está representado de acuerdo con estándares predefinidos, obtener software desarrollado de manera uniforme y hacer proyectos más manejables.

Las reuniones de revisión deben involucrar de tres a cinco personas. Estas tienen una preparación previa y no más de dos horas de trabajo. Estas sesiones se centran en una parte específica (y pequeña) del software general, realizando reportes y registros de la revisión. Sus documentos de salida son: una lista de pendientes de la revisión, reporte técnico formal de la revisión. El reporte de la revisión es una sola página (quizá con anexos). Con ello, se identifica las áreas de problemas en el producto.

En resumen, el objetivo de toda revisión técnica es detectar errores y descubrir aspectos que tendrían un efecto negativo en el software que se va a desarrollar. Entre más pronto se descubra y corrija un error, menos probable es que se propague a otros productos del trabajo de la ingeniería de software y que se amplifique, lo que provocaría un mayor esfuerzo para corregirlo.

CAPÍTULO III

ASEGURAMIENTO DE LA CALIDAD DEL SOFTWARE (ACS)

Elementos del Aseguramiento de la Calidad del Software (ACS)

El Aseguramiento de la Calidad del Software (ACS) es un conjunto de actividades planificadas y sistemáticas necesarias, que cumplen la función de cerciorarse que el producto sea construido de acuerdo con los estándares y normas establecidas, tanto en términos de gestión, como de funcionamiento, desde una perspectiva como cliente. Este proceso de la ingeniería del software se diseña y planea para cada aplicación antes de comenzar a desarrollarla y no después. El proceso de calidad considera los siguientes aspectos:

- Un enfoque de gestión de calidad.
- Métodos y herramientas de ingeniería del software.
- Revisiones técnicas formales en el proceso del software.
- Detección y documentación de errores y defectos.
- Una estrategia de prueba multiescala.
- El control de la documentación del software y de los cambios realizados.
- Procedimientos para ajustarse a los estándares de desarrollo del software.
- Aplicación de mecanismos de medición.
- Generación de informes de resultados y documentación.

El proceso es considerado como una actividad "sombrilla" dentro de un plan de trabajo, por lo que todas las personas involucradas, de alguna manera, participan en el aseguramiento de la calidad del producto. Sin embargo, existen dos grupos principales involucrados en este proceso:

- Los ingenieros en software desarrollan el trabajo técnico.
- Un grupo de SQA (Software Quality Assurance) se responsabiliza en la planificación de aseguramiento de la calidad, supervisión, mantenimiento de registros, análisis e informes.

Procesos de aseguramiento de la calidad

El proceso de aseguramiento de calidad comprende la mayor parte del proceso general de desarrollo del sistema. Las actividades se definen para aplicarse a diferentes aspectos de la visión general del sistema como producto final, con la certificación que todos los procesos y componentes construidos fueron evaluados y probados.

Elementos en el aseguramiento de calidad. Los elementos que intervienen en el aseguramiento de calidad se pueden resumir en los siguientes:

- Estándares: definen un conjunto de criterios que guían la forma en que se aplican procedimientos y metodologías al software desarrollado. La certificación de calidad permite una valoración independiente de la organización, donde se demuestra la capacidad de desarrollar productos y servicios de calidad. IEEE *(Institute of Electrical and Electronics Engineers)*, ISO y otras organizaciones que establecen estándares han producido una amplia variedad de ellos para ingeniería de software y documentos relacionados. Una organización de software adopta los estándares de manera voluntaria o los impone el cliente u otros participantes.

- Revisiones y auditorías: las revisiones técnicas son una actividad de control de calidad que realizan profesionales de software para otros profesionales de software. Su objetivo es detectar errores, por ejemplo, se realizan auditorías con el fin de asegurar que las revisiones se lleven a cabo de manera que tengan la máxima probabilidad de descubrir errores.

- Pruebas: son una función del control de calidad que tiene un objetivo principal detectar errores. Por ejemplo, se realizan pruebas de funcionamiento, de usabilidad, de fiabilidad, de rendimiento y de la capacidad de soporte.

- Colección y análisis de los errores: el Aseguramiento de la Calidad del Software (ACS) reúne y analiza errores y datos acerca de los defectos, para entender mejor cómo estos se cometen y qué actividades de la ingeniería de software son más apropiadas para eliminarlos.

- Administración del cambio: si no se administra de forma adecuada, lleva a la confusión y esta a su vez genera mal calidad.

- Educación: toda organización de software quiere mejorar sus prácticas de ingeniería de software. Un contribuyente clave de la mejora es la educación de los técnicos de software, de sus gerentes y de otros participantes.

- Administración de proveedores: son tres las categorías de software que se adquieren a proveedores externos: paquetes contenidos en una caja (por ejemplo, Office de Microsoft) que da una estructura básica; tipo esqueleto, que se adapta de manera única a las necesidades del comprador; software contratado, que se diseña y construye especialmente a partir de especificaciones provistas por la organización cliente.

- Administración de seguridad: con el aumento de los delitos cibernéticos, toda organización de software debe instituir políticas para proteger los datos en todos los niveles.

- Seguridad: debido a que el software casi siempre es un componente crucial de los sistemas humanos, la consecuencia de defectos ocultos puede ser catastrófica. Por ejemplo, fallas en aplicaciones automotrices o aeronáuticas.

- Administración de riesgos: el análisis y la mitigación de riesgos es asunto de los profesionales de software. La organización del ACS garantiza que las actividades de administración de riesgos se efectúen en forma apropiada y que se establezcan planes de contingencia relacionados con los riesgos.

Métricas de Aseguramiento de la Calidad del Software (ACS)

El objetivo del equipo de Aseguramiento de Calidad del Software (ACS) es auxiliar al equipo del software para lograr un producto final de alta calidad. En este contexto, se determinan las principales tareas para su estructuración:

- Preparación del plan de ACS para un proyecto.
- Participación en el desarrollo de la descripción del software del proyecto.
- Revisión de las actividades de la ingeniería de software, a fin de verificar el cumplimiento mediante el proceso definido para el software.
- Auditoría de los productos del trabajo de software designados, para verificar que se cumplan con aquellos definidos como parte del proceso de software.
- Aseguramiento de las desviaciones en el trabajo de software y que sus productos se documenten y manejen de acuerdo con un procedimiento documentado.
- Registro toda falta de cumplimiento y la reporte a la alta dirección.

Las desviaciones detectadas en las actividades del software y en los productos del trabajo de software son documentadas y manejadas de acuerdo a procedimientos previamente determinados. El grupo de ACS conduce periódicamente revisiones de sus actividades y reuniones con el personal de ACS del cliente, según sea necesario. Las métricas del software comprenden un amplio rango de actividades diversas, como: aseguramiento y control de calidad, modelos de fiabilidad, modelos y evaluación de ejecución, modelos y medidas de productividad.

Según Juran (1992), la calidad, para ser entendida de una mejor manera y posteriormente ser medida con eficacia, debe expresarse por medio de otros términos que tengan más sentido para el usuario. En el caso del software, estos factores son el medio por el que se traduce el término calidad al lenguaje de las personas que manejan la tecnología.

Los factores de calidad que afectan a la calidad del software se dividen en dos grandes grupos: los que miden directamente (defectos descubiertos en las pruebas) y los que se miden directamente (facilidad de uso o de mantenimiento). En cada caso debe presentarse una medición y se debe comparar el software con algún conjunto de datos y para obtener algún indicio sobre la calidad. manera voluntaria o los impone el cliente u otros participantes.

Enfoques formales al ACS

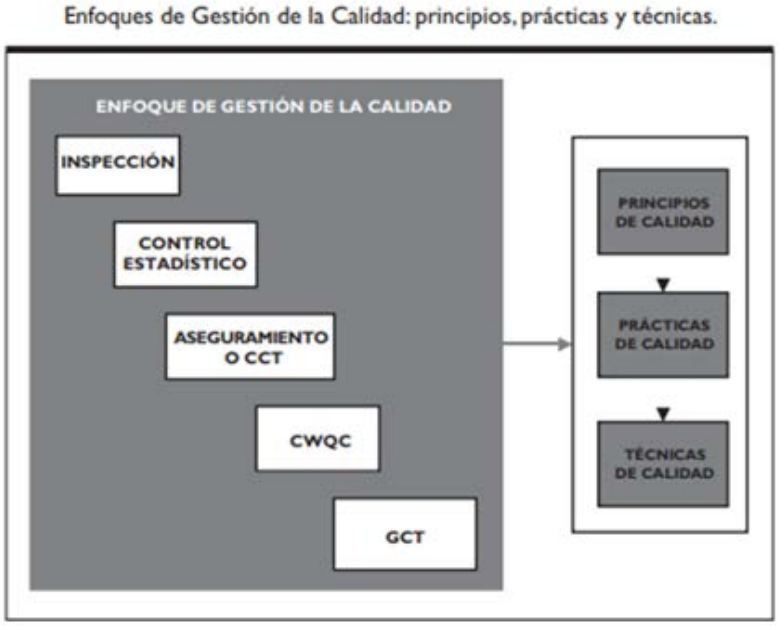
El enfoque de Gestión de la Calidad se utiliza para describir un sistema que relaciona un conjunto de variables relevantes para la puesta en práctica de una serie de principios, prácticas y técnicas para la mejora de la calidad. Así pues, el contenido de los distintos enfoques de Gestión de la Calidad se distingue por tres dimensiones.

- Los principios que asumen y que guían la acción organizativa.
- Las actividades que incorporan para llevar a la práctica estos principios.
- Las técnicas que intentan hacer efectivas estas prácticas.

Por ejemplo, un principio como la orientación hacia el cliente, asumido en diversos enfoques, puede conducir a que la organización lleve a cabo prácticas como la recogida sistemática de información sobre las necesidades, expectativas y satisfacción del cliente. Estas se hacen efectivas a través de estudios de mercado y pruebas de gusto en mercados seleccionados antes del lanzamiento de un nuevo producto.

Figura 5

Enfoques de Gestión de la Calidad.



Fuente: autoría propia.

Confiabilidad de software

La confiabilidad del software se mide y estima directamente mediante el uso de datos históricos del desarrollo. Según la IEEE (2025) la define como "la probabilidad que tiene un programa de cómputo de operar sin fallas en un ambiente específico por un tiempo específico". Si se considera un sistema basado en computadora, una medida sencilla de su confiabilidad es el Tiempo Medio Entre Fallas (TMEF):

$$TMEF = TMPF + TMPR$$

Donde las siglas TMPF y TMPR significan Tiempo Medio Para la Falla y Tiempo Medio Para la Reparación. Como cada defecto contenido en un programa no tiene la misma tasa de fallas, la cuenta total de defectos indica muy poco acerca de la confiabilidad del sistema. En resumen, a un usuario final le preocupan las fallas, no la cuenta total de los defectos.

Las normas de calidad ISO 9000

Las normas de la serie ISO 9000 fueron establecidas por la Organización Internacional de Normalización (ISO) para dar respuesta a una necesidad de las organizaciones: precisar los requisitos que debería tener un sistema de gestión de la calidad. La primera edición de estas normas se publicó en 1987. Posteriormente, ha sido modificada en 1994, 2000, 2008 2015; esta última versión es la vigente.

Las normas ISO nacieron aglutinando los principios que existían en multitud de normas de sistemas de calidad en distintos países. Desde su primera edición, se pretendió que fueran normas de aplicación a cualquier tipo de organización, independientemente de su tamaño o sector de actividad.

Las normas ISO 9000 están basadas en el principio de mejora continua. Según Alvarado (2019): "es importante que la gestión de calidad se fundamente en una cultura de calidad y trabajo en equipo, garantizando una gestión de mejora continua para la organización". Al efectuar las normas de forma efectiva, se logra aumentar sostenidamente el valor económico y la calidad de lo ofrecido a los clientes. Para la implementación de este sistema, las empresas deben realizar una serie de pasos que comprueban la gestión eficiente de su calidad:

- Definición de las necesidades y expectativas de los proyectos de la compañía.
- Identificación de las políticas y objetivos de calidad de cada producto.
- Determinación de los procesos de producción e identificación de los cuellos de botella que pueden surgir durante el proceso.
- Suficiencia de los recursos necesarios para el alcance de objetivos.
- Definición de las herramientas que permitirán medir la eficiencia y eficacia de los proyectos.
- Establecimiento de los métodos para evitar las no conformidades durante una auditoría.
- Determinación de las correcciones que se llevarán a cabo para la mejora continua de los procesos de producción.

Un sistema de gestión de calidad ayudará a las empresas a optimizar sus procesos, gracias a la consigna de mejora continua. La certificación ISO 9000 brinda reputación y confianza a las organizaciones, por lo que son más capaces de cerrar alianzas estratégicas con otras compañías o proveedores que les facilitarán la expansión.

CAPÍTULO IV

ESTRATEGIAS DE PRUEBA DE SOFTWARE

Aspectos estratégicos para la prueba del software

La estrategia proporciona la descripción de los pasos que hay que llevar a cabo como parte de la prueba, cuándo se deben planificar y realizar esos pasos y cuánto esfuerzo, tiempo y recursos se van a requerir. Una estrategia de prueba contiene: planificación de la prueba, diseño de casos de prueba, ejecución de las pruebas, agrupación y evaluación de datos.

En este contexto, existen muchas estrategias que pueden usarse para probar el software. En un extremo, se puede esperar hasta que el sistema esté completamente construido y luego realizar las pruebas sobre el sistema total, con la esperanza de encontrar errores. Este enfoque, aunque atractivo, simplemente, no funciona. En el otro extremo, podrían realizarse pruebas diariamente, siempre que se construya alguna parte del sistema. Este enfoque, aunque menos atractivo para muchos, puede ser muy efectivo.

Una estrategia de prueba que eligen la mayoría de los equipos de software se coloca entre los dos extremos. Toma una visión incremental de las pruebas, comenzando con la de unidades de programa individuales, avanza hacia pruebas diseñadas para facilitar la integración de las unidades y culmina con pruebas que ejercitan el sistema construido.

Pruebas de software

Los 5 tipos de prueba del software son:

Especificación. Este tipo de prueba incluye probar la aplicación en contra de la documentación que se hizo antes, por ejemplo, que los procesos concuerden con los algoritmos hechos a papel, o que la aplicación tenga todas las funciones que se habían planeado.

Usabilidad. Este tipo de prueba se refiere a asegurar que la interfaz de usuario (GUI) sea intuitiva, amigable y funcione correctamente.

Unidad. Este tipo de prueba solo aplica a proyectos grandes. En este sentido, se divide el proyecto a unidades y cada unidad es sometida a prueba individualmente.

Integración. Esta prueba varias unidades juntas para asegurar que funcionen bien.

También se asegura que las nuevas aplicaciones se integren con aplicaciones antiguas o aplicaciones complementarias.

Regresión. Esta prueba incluye todas las pruebas anteriores, en caso que se le haga algún cambio a algún modulo después de haber sido puesto en ambiente de producción.

La prueba del software es un elemento de un tema más amplio que suele denominarse verificación y validación.

Verificación. Es el conjunto de tareas que garantizan que el software implementa

correctamente una función específica. ¿Se está construyendo el producto correctamente?

Validación. Es un conjunto de actividades que aseguran que el software que se construye sigue los requerimientos del cliente. ¿Se está construyendo el producto correcto?

Posteriormente, la prueba alfa es conducida por un cliente en el lugar de desarrollo. Se usa el software de manera natural, con el encargado de desarrollo "mirando por encima del hombro del usuario" y registrando errores y problemas de uso. Las pruebas alfa se llevan a cabo en un entorno controlado.

Por otro lado, la prueba beta se lleva a cabo en uno o más lugares de clientes por los usuarios finales del software. A diferencia de la prueba alfa, el encargado de desarrollo, normalmente, no está presente. La prueba beta es una aplicación "en vivo" del software en un entorno que no puede ser controlado por el equipo de desarrollo. El cliente registra todos los problemas (reales o imaginarios) que encuentra y los informa a intervalos regulares. Como resultado, el equipo de desarrollo lleva a cabo modificaciones y así prepara una versión del producto para toda la base de clientes.

Ejercicios de aplicación

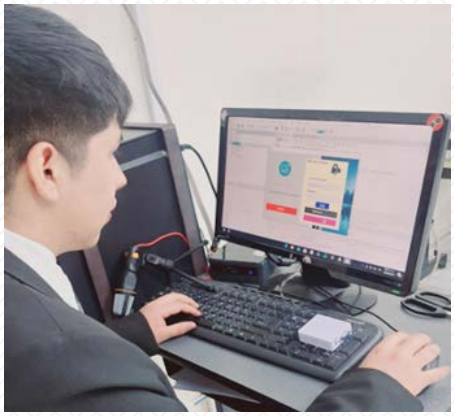
En este apartado, se presentan algunos ejemplos de aplicación de calidad de software.

Ejemplo 1: Software para el control de inventario de un comercio pequeño

A continuación, se desarrollan los pasos para la prueba de una aplicación hecha en NetBeans para un entorno de escritorio con Java.

Tabla 1

Desarrollo de ejemplo 1.

Aplicación de escritorio para la gestión de inventarios de un comercio pequeño	
Descripción del software por probar	**Figura 6** *Pruebas realizadas al software desarrollado por estudiantes de la carrera de Desarrollo de Software del Instituto Superior Tecnológico Tecnoecuatoriano.*  *Fuente:* autoría propia.

Las principales funcionalidades incluyen:

- Registro de productos: crear, actualizar y eliminar productos del inventario.
- Consulta de inventarios: verificar la cantidad de productos disponibles.
- Reporte de productos agotados: generar un reporte de productos con stock bajo.
- Sistema de autenticación: controlar el acceso a la plataforma a través de usuarios autorizados.

Objetivos de la prueba

El objetivo de la prueba es garantizar que el software cumpla con los requisitos funcionales y no funcionales establecidos, evaluando la integridad y calidad de la aplicación en términos de:

- Funcionalidad: verificar que cada función del sistema cumpla con los requisitos especificados.
- Usabilidad: asegurar que la aplicación sea fácil de usar y que la interfaz sea intuitiva.
- Seguridad: validar que el sistema de autenticación funcione correctamente.
- Rendimiento: verificar la velocidad y eficiencia del sistema bajo diferentes cargas.

Plan de pruebas

El objetivo de la prueba es garantizar que el software cumpla con los requisitos funcionales y no funcionales establecidos, evaluando la integridad y calidad de la aplicación en términos de:

- Funcionalidad: verificar que cada función del sistema cumpla con los requisitos especificados.
- Usabilidad: asegurar que la aplicación sea fácil de usar y que la interfaz sea intuitiva.
- Seguridad: validar que el sistema de autenticación funcione correctamente.
- Rendimiento: verificar la velocidad y eficiencia del sistema bajo diferentes cargas.
- Pruebas de seguridad: validarán que solo los usuarios autorizados puedan acceder al sistema.
- Pruebas de regresión: asegurarán que nuevas funcionalidades o correcciones no introduzcan errores en otras partes del sistema.

b. Entorno de pruebas

- Sistema operativo: Windows 11.
- Navegadores: Google Chrome, Mozilla Firefox.
- Base de datos: MySQL.
- Servidor de aplicación: Apache Tomcat.

c. Criterios de aceptación

- El software debe registrar y actualizar productos correctamente.
- Los usuarios no autorizados no deben poder acceder a la plataforma.
- La interfaz debe ser accesible desde dispositivos móviles y computadoras de escritorio.
- La aplicación debe soportar al menos 100 usuarios simultáneos sin errores.

Casos de prueba

Caso de prueba 1: registro de un nuevo producto y registro de clientes

Descripción: verificar que el sistema permite agregar un nuevo producto al inventario.

Pasos:

- Iniciar sesión con un usuario válido.
- Navegar a la sección "Productos".
- Ingresar los datos del producto (descripción, cantidad, precio).
- Hacer clic en "Guardar".
- Navegar a la sección "Clientes".
- Ingresar los datos del cliente (RUC, nombre, teléfono, dirección).
- Hacer clic en "Guardar".

Resultado esperado: el sistema debe mostrar un mensaje de confirmación y el producto debe aparecer en el inventario.

Resultado real:

Figura 7

Pantalla de inicio.

Fuente: autoría propia.

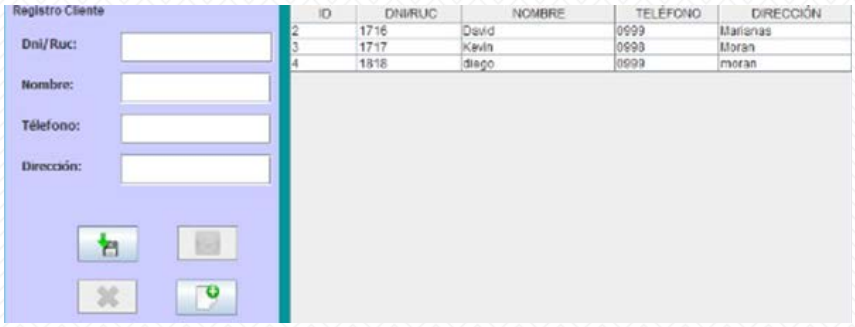
Figura 8

Menú disponible del programa.

Fuente: autoría propia.

Figura 9

Ejemplo del ingreso de nuevos clientes.



Fuente: autoría propia.

Caso de prueba 2: consulta de historial de ventas

Descripción: verificar que el sistema permita buscar el historial de ventas.

Pasos:

- Iniciar sesión con un usuario válido.
- Navegar a la sección "Ventas".
- Hacer clic en "Historial".

Resultado esperado: el sistema debe mostrar el historial de ventas.
Resultado real:

Figura 10

Ejemplo de una consulta de "Historial de Ventas".

Fuente: autoría propia.

Caso de prueba 3: acceso al sistema con un usuario no autorizado

Descripción: verificar que un usuario no autorizado no pueda acceder al sistema.
Pasos:
- Intentar iniciar sesión con un usuario no registrado.

Resultado esperado: el sistema debe mostrar un mensaje de error indicando que las credenciales no son válidas.

Caso de prueba 4: reporte de productos agotados

Descripción: verificar que el sistema genere un reporte de productos con stock bajo.
Pasos:
- Iniciar sesión con un usuario válido.
- Navegar a la sección "Reporte de productos agotados".
- Hacer clic en "Generar reporte".

Resultado esperado: el sistema debe generar un reporte con los productos cuyo stock sea menor al mínimo definido.

Pruebas de rendimiento

Se realizarán pruebas de carga utilizando herramientas como JMeter para simular 100, 200 y 500 usuarios simultáneos. Se medirá el tiempo de respuesta del servidor y se verificará que la aplicación siga funcionando sin errores críticos bajo estas condiciones.

Reporte de resultados	Al final de las pruebas, se generará un reporte detallado que incluirá: - Resumen de casos de prueba: con los resultados esperados y reales. - Bugs encontrados: listado de errores encontrados, priorizados por su impacto en el sistema. - Recomendaciones: sugerencias para mejorar la funcionalidad o rendimiento del sistema. - Cierre de pruebas: se indican si el software cumple con los criterios de aceptación y está listo para su lanzamiento.

Ejemplo 2: Sistema de gestión de inventarios para una cadena de suministro

En el Ejemplo 2, se indican las metodologías de calidad implementadas a lo largo de un proyecto, donde se identifica lo más relevante de la aplicación.

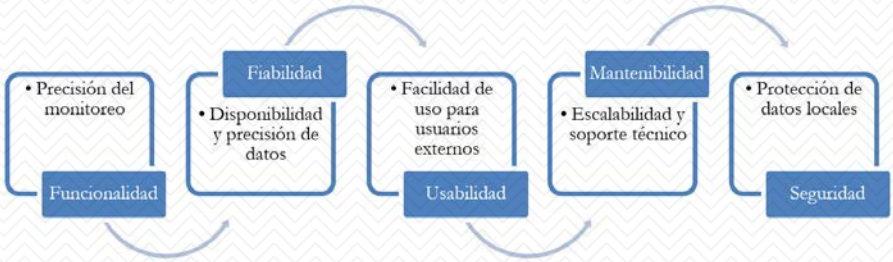
Tabla 2

Desarrollo del ejemplo 2.

Descripción del software	Una cadena de suministro internacional necesita un sistema de gestión de inventarios que optimice el control de existencias en sus bodegas distribuidas por diferentes países. Este software debe integrarse con los sistemas de gestión existentes (ERP) y proporcionar interfaces de usuario tanto para la gestión interna como para los clientes corporativos. El objetivo es reducir costos operativos, mejorar la precisión de los inventarios y automatizar el reabastecimiento en función de la demanda.
Metodologías de calidad aplicadas a lo largo del proyecto	- ISO/IEC 25010: modelo de calidad del producto de software. - *Six Sigma:* mejora de la calidad. - CMMI (*Capability Maturity Model Integration*). - *Lean Software Development.* - TMMi (*Test Maturity Model Integration*): modelo de madurez de pruebas. - DevOps: integración y entrega continua. - ISO 9001:2015 Gestión de Calidad.
ISO/IEC 25010: Modelo de calidad del producto de software	Durante la fase de requisitos, se utiliza el estándar ISO/IEC 25010 para asegurar que el producto cumpla con los atributos de calidad esenciales, como funcionalidad, fiabilidad, usabilidad, eficiencia, mantenibilidad y seguridad (ISO/IEC 25010, 2011). Las métricas derivadas del estándar permiten evaluar el cumplimiento de estos atributos a lo largo del ciclo de vida del software. Como herramientas se emplean métricas de calidad de software recomendadas en el estándar para asegurar la funcionalidad y seguridad del sistema descritas en la Figura 11.

Figura 11

Características del software al aplicar ISO/IEC 25010.



Fuente: autoría propia.

Six Sigma: mejora de la calidad

Six Sigma se aplica durante la fase de análisis y diseño para reducir defectos en los procesos del sistema (Schroeder, et al., 2002). Se sigue el ciclo DMAIC para definir, medir, analizar, mejorar y controlar los procesos críticos, como la integración del software con el ERP existente.

Dentro de las herramientas que se utilizan están los diagramas de Pareto y análisis de causa-raíz para identificar problemas y posibles mejoras.

***CMMI (Capability Maturity Model Integration):* Gestión de procesos**

CMMI se utiliza para gestionar el proyecto asegurando procesos bien definidos y ejecutados de manera consistente (Chrissis et al., 2011). El equipo se enfoca en alcanzar los niveles de madurez 2 y 3, que establecen la gestión y definición de procesos estandarizados.

***Lean Software Development:* Eliminación de Desperdicios**

En la fase de desarrollo, Lean ayuda a eliminar desperdicios y mejorar la eficiencia del equipo, asegurando que se eviten la sobreproducción y las tareas innecesarias (Poppendieck & Poppendieck, 2003). Se busca la mejora continua a través de la entrega rápida y la retroalimentación constante de los usuarios.

Las herramientas que se utilizan son el uso de tableros Kanban para visualizar el flujo de trabajo.

***TMMi (Test Maturity Model Integration):* Modelo de madurez de pruebas**

Durante la fase de pruebas, se sigue el modelo TMMi para madurar los procesos de prueba a través de niveles predefinidos de calidad (van Veenendaal & Wells 2012). Esto garantiza pruebas sistemáticas y proactivas. El beneficio de aplicar esta metodología se evidencia en que las pruebas son más robustas y estructuradas, que ayudan a evitar problemas durante la operación.

DevOps: Integración y entrega continua	Se emplea DevOps para la entrega continua, integrando el software de manera eficiente y permitiendo la automatización del despliegue (Forsgren, Humble & Kim, 2018). Esto asegura una implementación sin errores y una actualización continua en producción. Las herramientas aplicadas consisten en Jenkins y Docker para automatización de despliegue y pruebas.
ISO 9001:2015 - Gestión de calidad	ISO 9001 se aplica a la gestión del proyecto, estableciendo procedimientos y estándares de calidad en los procesos de desarrollo y pruebas del software (ISO, 2015). Uno de los principales beneficios de aplicación de esta metodología es la mejora de la eficiencia operativa y garantía de la satisfacción del cliente.

Fuente: autoría propia.

Ejemplo 3: Plataforma de monitoreo de sostenibilidad para la gestión de recursos hídricos

En el Ejemplo 3, se muestra de igual forma las metodologías que se pueden implementar en un proyecto de monitoreo.

Tabla 3

Desarrollo del ejemplo 3.

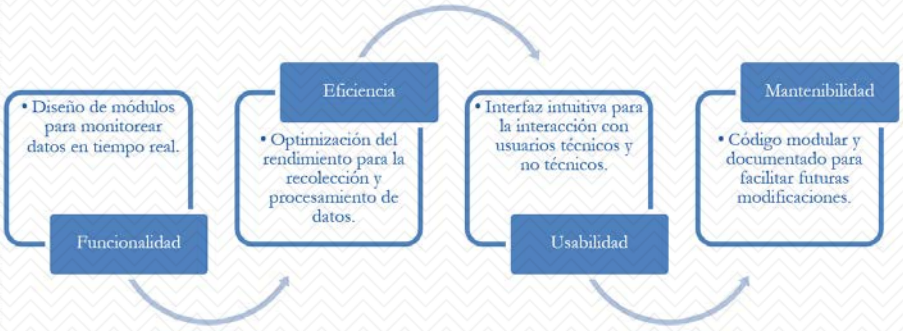
Descripción del software	Una organización sin fines de lucro está desarrollando una plataforma de software para monitorear el uso y conservación de recursos hídricos en áreas rurales, alineada con los Objetivos de Desarrollo Sostenible (ODS), específicamente con el ODS 6: Agua limpia y saneamiento. El software permite a los usuarios registrar datos sobre el consumo de agua, la calidad del agua y las infraestructuras locales. La plataforma tiene como objetivo apoyar a gobiernos y comunidades a gestionar el uso eficiente de los recursos hídricos, reportar problemas y hacer un seguimiento del progreso hacia el ODS 6. El acceso y uso eficiente del agua es un desafío global, particularmente en áreas que enfrentan escasez y riesgos ambientales.
Metodologías de calidad aplicadas en el proyecto	- ISO/IEC 25010: Modelo de calidad del producto - Lean Software Development. - TMMi: Modelo de madurez de pruebas. - DevOps: Integración continua - ISO 9001:2015: Gestión de calidad

Modelo de calidad ISO/IEC 25010

Este modelo, representado en la Figura 12, evalúa la calidad del software en términos de características como funcionalidad, eficiencia, compatibilidad, usabilidad, fiabilidad, seguridad, mantenibilidad y portabilidad (Martínez, García & Guerrero, 2018).

Figura 12

Modelo de calidad ISO/IEC 25010 aplicado en el ejemplo 3.



Fuente: autoría propia.

CMMI *(Capability Maturity Model Integration)*

Este modelo, representado en la Figura 13, proporciona un marco para la mejora continua del proceso de desarrollo de software, midiendo la madurez de los procesos en cinco niveles (Sánchez Pérez, 2019).

Figura 13

Modelo según la metodología CMMI para el ejemplo 3.

Fuente: autoría propia.

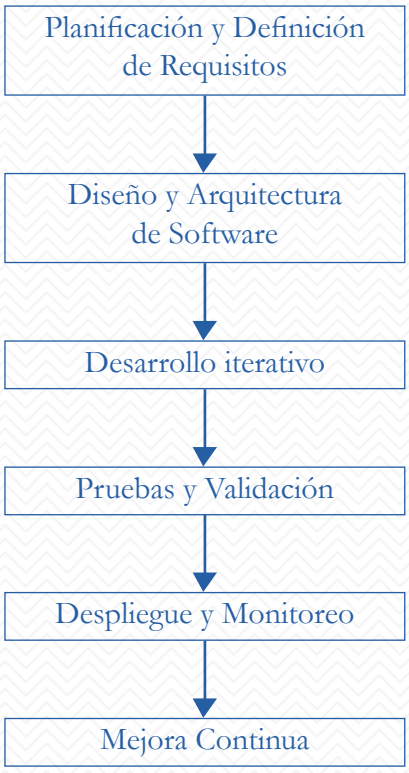
Metodologías ágiles

El enfoque ágil se centra en la entrega incremental de software y la adaptación rápida a cambios (López Gil, 2018).

- Se aplica el uso de Scrum para gestionar el desarrollo de sprints y asegurar la entrega continua de funcionalidades.
- Se incorporan pruebas automatizadas para asegurar la calidad en cada iteración.
- Es necesario un feedback constante de los usuarios para ajustar el desarrollo a las necesidades cambiantes, como se visualiza en la Figura 14.
- Planificación y definición de requisitos.
- Recopilación y análisis de necesidades de los stakeholders.
- Diseño y arquitectura del software.
- Definición de la estructura del sistema y la arquitectura de N capas.
- Desarrollo iterativo.
- Implementación y desarrollo de funcionalidades mediante sprints ágiles.
- Pruebas y validación.
- Verificación de la funcionalidad y calidad del software.
- Despliegue y monitoreo.
- Implementación en producción y monitoreo del rendimiento.
- Mejora continua.
- Análisis y retroalimentación para optimizar el sistema.

Figura 14

Aplicación de las metodologías ágiles en el ejemplo 3.



Fuente: autoría propia.

Recomendaciones finales

Para los desarrolladores de software que desean especializarse en calidad de software, es crucial obtener certificaciones que les permitan mejorar sus habilidades en la evaluación, control y mejora de la calidad de productos de software. A continuación, se presentan algunas de las certificaciones más importantes que abordan los requisitos mínimos para garantizar la calidad de software en entornos profesionales.

Certified Software Quality Analyst (CSQA). El CSQA, ofrecido por el QAI Global Institute, es una de las certificaciones más reconocidas en el área de la calidad de software. Esta certificación está diseñada para profesionales interesados en la planificación, gestión y evaluación de los procesos de calidad del software. De esta manera, el CSQA abarca una amplia gama de conceptos, como la medición de la calidad, el aseguramiento de la calidad y las mejores prácticas en pruebas.

Figura 15

Logo de la certificación CSQA.

Fuente: GAQM (2024).

Para su consecución, existen algunos requisitos mínimos como experiencia previa en el área de aseguramiento de la calidad o pruebas de software. Se recomienda tener conocimientos básicos sobre metodologías de desarrollo de software. Uno de sus principales beneficios es que proporciona una base sólida en principios de calidad del software y cómo implementarlos en proyectos de desarrollo para mejorar la productividad y minimizar errores.

ISTQB Certified Tester Foundation Level (CTFL). La I*nternational Software Testing Qualifications Board* (ISTQB) es una organización reconocida internacionalmente por su programa de certificación de pruebas de software. El nivel *Foundation* (CTFL) es ideal para aquellos que comienzan en el campo de pruebas de software y desean comprender los fundamentos del control de calidad, las técnicas de prueba y los procesos de revisión.

Figura 16

Logo de la Certificación ISTQB®.

Fuente: ISTQB (2024).

Esta certificación no requiere experiencia previa, lo que la convierte en una excelente opción para desarrolladores que deseen obtener una comprensión básica de las pruebas de software y su relación con la calidad. Además, esta introduce a los desarrolladores en las técnicas y prácticas fundamentales de pruebas de software, esenciales para asegurar la calidad del producto final.

Certified Manager of Software Quality (CMSQ). También ofrecida por QAI Global Institute, la certificación CMSQ está dirigida a aquellos que ocupan roles de liderazgo o supervisión en el ámbito de la calidad del software. Esta se centra en los aspectos de gestión de los procesos de calidad, proporcionando un enfoque más estratégico sobre cómo supervisar y mejorar la calidad dentro de un equipo o proyecto.

Figura 17

Logo de la certificación CSQM.

Fuente: GAQM (2024).

Para la certificación CSQM, generalmente, se requiere experiencia previa en la gestión de proyectos de software o calidad de software. Idealmente, los candidatos deben tener conocimientos sobre control y evaluación de calidad. Esta certificación permite a los profesionales liderar iniciativas de calidad de software, implementar estándares y guiar equipos hacia la mejora continua del producto.

ISO 9001 *Lead Auditor.* Aunque no está dirigida exclusivamente a software, la certificación ISO 9001 Lead Auditor es altamente relevante para garantizar que los procesos de desarrollo de software cumplan con los estándares internacionales de gestión de calidad. Los auditores certificados en ISO 9001 pueden evaluar sistemas de gestión de calidad, identificar deficiencias y proponer mejoras, asegurando que el software desarrollado cumpla con los requisitos normativos y de calidad.

Figura 18

Logotipo de la certificación ISO 9001 Lead Auditor.

Fuente: GAQM (2025).

Para conseguir esta certificación existen requisitos mínimos, como conocimientos previos sobre gestión de calidad y familiaridad con el estándar ISO 9001. Adcional, la experiencia en auditorías de calidad es deseable. Esta permite evaluar y auditar los sistemas de calidad de software en una organización, alineando los procesos con los estándares internacionales.

Six Sigma Green Belt. La certificación *Six Sigma Green Belt* es ampliamente reconocida en el campo de la mejora de procesos y calidad. Aunque no es específica para el desarrollo de software, sus principios de mejora continua y reducción de defectos pueden aplicarse directamente a los procesos de desarrollo y control de calidad de software. Esta certificación se enfoca en la recolección de datos, análisis estadístico y metodologías de mejora como DMAIC (Definir, Medir, Analizar, Mejorar y Controlar). Esta certificación no se requiere experiencia específica, pero es útil tener conocimientos básicos de mejora de procesos y análisis de datos.

Como beneficio, la *Six Sigma Green Belt* proporciona herramientas para mejorar los procesos de desarrollo de software, reducir defectos y aumentar la eficiencia en la entrega de productos de alta calidad. Según la institución el enfoque de la certificación varía en tres tipos: CLSSGB (*Certified Lean Six Sigma Green Belt)*, LSSGBPC (*Lean Six Sigma Green Belt Professional Certification*) o Lean 7 (metodologías para eliminar los desperdicios en los procesos, sin necesariamente enfocarse tanto en

la parte estadística o de variación que se asocia con S*ix Sigma*).

Figura 19

Logos representativos según las metodologías de las academias que emiten las certificaciones.

Fuente: GAQM (2024).

Certified Agile Tester (CAT). Para desarrolladores que trabajan en entornos ágiles, la certificación *Certified Agile Tester* (CAT) proporciona conocimientos sobre cómo realizar pruebas de software en equipos que usan metodologías ágiles como Scrum o Kanban. La calidad de software en entornos ágiles se evalúa a través de pruebas iterativas y continuas, asegurando que el software funcione correctamente a lo largo de todo el ciclo de desarrollo. Su logro requiere experiencia previa en desarrollo de software o participación en proyectos ágiles.

La *Certified Agile Tester* (CAT) mejora la capacidad de los desarrolladores para realizar pruebas de calidad continua y colaborar eficazmente en equipos ágiles, garantizando la entrega de software de alta calidad en ciclos cortos de desarrollo. En la Figura 20 se muestra el logo de la certificación inicial CATF (*Certified Agile Tester Foundation*). Esta no incluye tantas técnicas avanzadas de pruebas o automatización como la certificación CAT de nivel más avanzado.

Figura 20

Logo de la certificación CATF a nivel fundamental para iniciar con las metodologías ágiles.

Fuente: GAQM (2025).

En la Tabla 4 se realiza un resumen de las principales certificaciones con el costo aproximado de inversión y el tiempo que requieren para su obtención.

Tabla 4

Resumen de certificaciones en el tema de calidad de software.

Certificación	Costo aproximado (USD)	Tiempo aproximado (meses)	Ranking de popularidad	Observaciones
ISO 9001	2,500 - 5,000	6 – 12	1	Se compone de módulos para implementación y auditoría.
ISTQB Certified Tester Foundation Level (CTFL)	300	2 – 3	3	Consta de un examen directo basado en fundamentos de pruebas.
Certified Software Quality Analyst (CSQA)	450 – 500	3 – 4	4	Se compone de módulos de evaluación de calidad.
CDMP (Certified Data Management Professional)	311 por examen	Variable (depende de exámenes)	5	Basado en exámenes de diferentes áreas de gestión de datos.
Certified Scrum Master (CSM)	400 - 1,200	1 – 2	2	La preparación está enfocada en un solo módulo.
Six Sigma Green Belt	400 - 1,200	3 – 6	3	Existen varios módulos de mejora de procesos y control de calidad.

Nota. El tiempo establecido considera la modalidad en línea según el ritmo de los participantes.
Fuente: autoría propia.

Referencias

Alvarado, F. (2019). Normas ISO 9000: *conoce el sistema de gestión de calidad.* Conexión Esan. https://bit.ly/41mRVKH

Chrissis, M. B., Konrad, M., & Shrum, S. (2011). *CMMI for development: guidelines for process integration and product improvement.* Pearson Education.

Faisal (2023). *Core Metrics, Score and Rating.* [Conjunto de datos de GitHub] https://github.com/whitesmith/rubycritic/blob/master/docs/core-metrics.md

Forsgren, N., Humble, J., & Kim, G. (2018). *Accelerate: The science of lean software and devops: Building and scaling high performing technology organizations.* IT Revolution.

Global Association for Quality Management [GAQM]. Logotipo certificación iso 9001 [imagen]. https://gaqm.org/certifications/iso_certifications/iso-9001-2015-cla

Global Association for Quality Management [GAQM]. Logotipo certificación Agile testing [imagen]. https://gaqm.org/certifications/agile_testing/catf

Global Association for Quality Management [GAQM]. *Logotipo certificación lean six sigma* [imagen]. https://gaqm.org/certifications/lean_six_sigma/clssgb

Global Data Management Community. *Logotipo certificación CDMP* [imagen]. https://cdmp.info/

International Organization for Standardization (ISO). (2015). ISO 9001:2015 (es) *Sistemas de gestión de la calidad - Requisitos ISO.* https://www.iso.org/obp/ui/es/#iso:std:iso:9001:ed-5:v1:es

International Software Testing Qualifications Board [ISTQB]. *Logotipo certificación ISTQB* [imagen]. https://www.istqb.org/

Lasso Pozo, E. (2021). *Efecto de la aplicación de la Norma ISO 9001-2015 "Mejora" (Sistema de Gestión de Calidad) dentro de las empresas situadas en el DM. Quito, sector norte de la ciudad, dentro del periodo 2020.* [Licenciatura, Universidad Politécnica Salesiana]. Dspace. https://dspace.ups.edu.ec/bitstream/123456789/20980/1/TTQ420.pdf

López Gil, A. (2018). *Estudio comparativo de metodologías tradicionales y ágiles para proyectos de Desarrollo de Software.*

López Lemos, P. (2014). *Novedades ISO 9001: 2015.* FC Editorial. https://elibro.net/es/ereader/istte/114074?page=11

Martínez, S. J., García, J. L., & Guerrero, J. L. (2018). Sistema de gestión de calidad y certificación ISO 9001: 2008-Limitantes y desafíos para las Pymes. *Revista espacios*, 39(09).

Mejía Trejo, J. (2024). P*rincipios de aseguramiento de calidad para el diseño de software: innovación de procesos en las tecnologías de información.* Academia Mexicana de Investigación y Docencia en Innovación (AMIDI). https://elibro.net/es/lc/istte/titulos/250101

Poppendieck, M., & Poppendieck, T. (2003). *Lean software development: An agile toolkit: An agile toolkit.* Addison-Wesley.

Sánchez Pérez, E. P. (2019). *Evaluación del ERP university de la universidad ULADECH católica usando ISO/IEC 15504-4–Chimbote.*

Schroeder, R. G., Linderman, K., Liedtke, C., & Choo, A. S. (2008). Six Sigma: Definition and underlying theory. *Journal of operations Management*, 26(4), 536-554.

Van Veenendaal, E., & Wells, B. (2012). Test maturity model integration (TMMi). *TMMI Foundation (www. tmmifoundation. org)*, Uitgeverij Tutein Nolthenius.

Zimbrón, G. (2017). *Factores de calidad según Boehm* [imagen]. Zimbronapps. https://zimbronapps.com/wp-content/uploads/2017/11/Factores-de-calidad-seg%C3%BAn-Boehm.jpg

ISBN: 978-9942-676-89-4

Matriz La Magdalena: Calle Jambelí Oe3-158 y La Unión.
Campus Eloy Alfaro: José Barreiro y Av. Eloy Alfaro N52-85, Sector Solca.
Campus Calderón: Calle Los Cipreses N6-99 y Giovanni Calles.
Campus Pifo: Ignacio Fernández Salvador Oe2-439 y Pasaje Baldeón.
Sede Santa Elena: La Libertad, barrio 25 de Septiembre, Av. 25 y calle 28.
Sede Guayaquil: Calle 6 de Marzo y Rosendo Avilés, Barrio del Centenario.
www.tecnoecuatoriano.edu.ec