



TALLER DE TESTING Y CALIDAD DE SOFTWARE

Semana 2





ESCUELA DE CONSTRUCCIÓN E INGENIERIA

Director: Marcelo Lucero

ELABORACIÓN

Experto disciplinar: Aída Villamar Gallardo.

Diseño instruccional: Carla Silva Alvarado.

VALIDACIÓN

Experto disciplinar: Andrés del Alcázar

Jefa de Diseño Instruccional: Alejandra San Juan Reyes.

EQUIPO DE DESARROLLO

AIEP

AÑO

2021



Tabla de contenidos

Aprendizaje esperado de la semana	4
Introducción	4
1. Etapas del ciclo de vida de producto, considerando concepto de calidad y pruebas asociadas a cada etapa, de acuerdo con requerimientos del mercado. ...	5
1.1.- Características de un producto exitoso	5
1.2.- Ciclo de vida de un producto: etapas y pruebas asociadas.....	12
2. Relación entre calidad de software como producto con su comercialización y rendimiento en el mercado informático.	14
2.1.- Cumplimiento de requerimientos del mercado	14
2.2.- Relaciona verificación y validación con certificación de calidad de software	16
3. Relaciona verificación y validación con certificación de calidad de software.	18
3.1.- Calidad de los productos	18
3.2.- Calidad del software como producto	19
3.3.- Características de calidad del software	20
4. Determina requerimientos de calidad de software, según estándares ISO.	25
4.1.- Calidad de un software según estándares ISO	25
Ideas clave	27
Conclusiones	29
Referencias bibliográficas	30



Aprendizaje esperado de la semana

Analizan aspectos asociados a calidad de software como producto, según requerimientos del mercado.

Introducción

- ¿Qué es calidad en el software?
- ¿Cómo garantizamos un software de calidad?
- ¿Qué relación existe entre la verificación y validación respecto a la calidad del software?

Como podemos apreciar son estas y muchas las preguntas que nos podemos hacer al momento de pensar en calidad dentro de nuestro desarrollo. En esta semana veremos los aspectos tanto en teoría, como son los conceptos involucrados, así como la aplicación, es decir, cuales son las pruebas que se deben realizar, en qué etapa y cómo esto afecta al resultado que vamos a obtener, ya que la calidad no se obtiene por casualidad, es producto de la incorporación del concepto a lo largo de todo el desarrollo de nuestro proyecto.

1. Etapas del ciclo de vida de producto, considerando concepto de calidad y pruebas asociadas a cada etapa, de acuerdo con requerimientos del mercado.

1.1.- Características de un producto exitoso

El software es una categoría amplia que incluye gran variedad de productos informáticos, desde sistemas operativos hasta la parte intangible de juegos electrónicos o los programas que permiten el funcionamiento de un avión.¹



Figura 1: Software que permite el funcionamiento de un cohete

Fuente: (<https://mgx.com.co>, s.f.)

A continuación, algunas características que se esperan en un software considerado exitoso:

Amigabilidad	Evolucionabilidad	Performance	Reusabilidad
Comprensibilidad	Interoperabilidad	Portabilidad	Robustez
Confiabilidad	Mantenibilidad	Productividad	Verificabilidad
Correctitud	Oportunidad	Reparabilidad	Visibilidad

Figura 2: Características que se esperan de un producto de software

Fuente: Elaboración propia

No obstante, encontraremos que solo algunos de los productos de software van a cumplir con estas características, debido a que se da prioridad a una u otra dependiendo del uso que tendrá. Si consideramos el software de una lavadora, veremos que no necesita

¹ <https://www.caracteristicas.co/software/>



priorizar aspectos de seguridad, pues no resguarda ningún tipo de información importante.

¿Cuáles son los pasos?

El software se construye como cualquier producto exitoso, se aplica un proceso idealmente ágil y que se adapte con el fin de obtener un resultado de calidad, es decir, la obtención de un producto de software que debe satisfacer las necesidades de los futuros usuarios. A continuación, veremos los pasos de la aplicación del enfoque de la ingeniería de software.

¿Cuál es el producto final?

En este punto podemos considerar dos visiones diferentes:

- **Ingeniero de software:** para quienes el producto final considera todos los programas, los datos que contienen y otros productos terminados que conforman el software.
- **Usuario:** para quienes el producto final corresponde al resultado de la ejecución, es decir, la información que reciben y que de una u otra manera mejora el mundo en que viven.

Uno de los principios generales de la ingeniería de software corresponde a mantener la visión, lo que es fundamental para lograr el éxito en un proyecto de software, sabemos que si no contamos con esta visión el proyecto se convertirá en un monstruo de dos cabezas, incluso tal vez más. A esto nos referimos con el termino integridad referencial, es importante no comprometer la visión de la arquitectura de un sistema debido a que lo debilita y muchas veces puede generar un colapso incluso de software bien diseñados. Contar con un arquitecto que mantenga la visión y obligue a cumplirla es una garantía para un proyecto de software muy exitoso, y esto se logra con la incorporación en todo el proceso de desarrollo del concepto de calidad por medio de pruebas, verificando y validando que todo se mantenga como se ha planificado.

Otro punto importante para considerar son los mitos que han sido alimentados por más de cincuenta años en la cultura de la programación, ahí encontramos los mitos del profesional, que son



aquellos mitos que aún afirman los trabajadores del software. En sus comienzos, la programación se veía como una forma de arte. No es sencillo que cesen los hábitos y actitudes que se encuentran arraigadas.

Mito: El único producto del trabajo que se entrega en un proyecto exitoso es el programa que funciona.

Realidad: Un programa que funciona sólo es una parte de una configuración de software que incluye muchos elementos. Son varios los productos terminados (modelos, documentos, planes) que proporcionan la base de la ingeniería exitosa y, lo más importante, que guían el apoyo para el software. (Roger S. Pressman, INGENIERÍA DEL SOFTWARE. UN ENFOQUE PRÁCTICO, 2010)

Otra característica conocida en lo que ha software respecta, son los cambios constantes, tiempos de entrega muy acotados, y la necesidad por obtener la satisfacción tanto del cliente como del usuario. Muchas veces el requerimiento administrativo fundamental es el tiempo que se entrega para llegar al mercado.

Si se pierde un nicho de mercado, todo el proyecto de software podría carecer de sentido. Sin embargo, es importante notar que ser el primero en llegar al mercado no es garantía de éxito. En realidad, muchos productos de software muy exitosos han llegado en segundo o hasta en tercer lugar al mercado (aprenden de los errores de sus antecesores). (Roger S. Pressman, INGENIERÍA DEL SOFTWARE. UN ENFOQUE PRÁCTICO, 2010)

Encontramos otro mito, la relación que existe entre personal y esfuerzo, si pensamos en un proyecto de desarrollo de software pequeño, vemos que una persona podría realizar el analizar de los requerimientos, diseñar, generar código y efectuar las pruebas. A medida que el tamaño del proyecto va en aumento, se deben incluir más personas.



¡Rara vez uno puede darse el lujo de abordar un esfuerzo de 10 persona-años con una persona que trabaje durante 10 años! (Roger S. Pressman, INGENIERÍA DEL SOFTWARE. UN ENFOQUE PRÁCTICO, 2010)

Existe otro mito común hoy en día entre gerentes responsables de proyectos de desarrollo de software:

“Si nos atrasamos en el calendario, siempre podemos agregar más programadores y ponernos al corriente en el proyecto más adelante”. (Roger S. Pressman, INGENIERÍA DEL SOFTWARE. UN ENFOQUE PRÁCTICO, 2010)

Lamentablemente incluir personal en etapas posteriores al comienzo del proyecto por lo general implica efectos complejos y el calendario se ve perjudicado aún más, esto debido a que deben interiorizarse acerca del sistema y quienes que les enseñan son los mismos que hacían el trabajo, es decir, ellos enseñan por lo que no trabajan y por lo tanto el proyecto se atrasa aún más, además con esta incorporación aumenta la cantidad de vías de comunicación y la complejidad de estas a lo largo de un proyecto. La comunicación es fundamental en el desarrollo de software exitoso, toda nueva vía de comunicación implica un esfuerzo adicional y, por lo tanto, tiempo adicional.

Otro punto importante para obtener un sistema exitoso consiste en la contribución entre todos, es decir, los clientes y otros participantes, deben colaborar entre sí, sin discusiones por detalles menores y cooperar con los profesionales de la ingeniería de software.

Además de lo anterior, encontramos otro elemento importante, estas son las decisiones orientadas al riesgo que es uno de los atributos fundamentales de un proyecto exitoso de software. Realmente es necesario saber que puede salir mal y definir un plan de contingencia en el caso de que suceda ya que muchas veces se trabaja con un optimismo ciego y cuando un riesgo se convierte en realidad, reina el caos y se acrecienta el nivel de locuras que se realizan, con lo que inevitablemente la calidad se derrumba.

El dilema de la calidad del software se resume mejor con el enunciado de la Ley de Meskimen:



Nunca hay tiempo para hacerlo bien, pero siempre hay tiempo para hacerlo otra vez. Mi consejo es: tomarse el tiempo para hacerlo bien casi nunca es la decisión equivocada. (Roger S. Pressman, INGENIERÍA DEL SOFTWARE. UN ENFOQUE PRÁCTICO, 2010)

El proyecto: En un estudio de 250 grandes proyectos de software desarrollados entre 1998 y 2004, Capers Jones encontró que “alrededor de 25 se consideraron exitosos por haber logrado sus objetivos de calendario, costo y calidad. Aproximadamente 50 tuvieron demoras o excesos por abajo de 35 por ciento, mientras que más o menos 175 experimentaron grandes demoras y excesos, o se dieron por concluidos sin completarse”. Aunque actualmente la tasa de éxito para los proyectos de software puede haber mejorado un poco, la tasa de falla de proyecto sigue siendo mucho más alta de lo que debiera. (Roger S. Pressman, INGENIERÍA DEL SOFTWARE. UN ENFOQUE PRÁCTICO, 2010)

Si queremos lograr éxito en nuestros proyectos estos son los puntos que debemos considerar:

- Evitar un conjunto de señales de advertencia comunes.
- Comprender cuales son los factores de éxito fundamentales que llevan a una buena administración del proyecto.
- Llevar a cabo un enfoque con sentido común para la planificación, monitoreo y control del proyecto.



Si pensamos en el personal involucrado en el desarrollo del proyecto, debemos analizar las siguientes consideraciones:

En un estudio publicado por el IEEE², se preguntó a los vicepresidentes de ingeniería de tres grandes compañías tecnológicas cuál era el elemento más importante para el éxito de un proyecto de software.

Ellos respondieron de la siguiente manera:

VP 1: Supongo que, si tienes que elegir una cosa que sea la más importante en nuestro ambiente, diría que no son las herramientas que usamos, es el personal.

VP 2: El ingrediente más importante que fue exitoso en este proyecto fue tener gente inteligente [...] en mi opinión, muy pocas cosas más importan [...] La cosa más importante que haces para un proyecto es seleccionar al personal [...] El éxito de la organización de desarrollo de software está muy, muy asociada con la habilidad para reclutar buen personal.

VP 3: La única regla que tengo en la administración es asegurarme de que tengo buen personal, gente realmente buena, y que hago crecer gente buena y que proporcionó un ambiente en el que la gente buena puede producir. (Roger S. Pressman, INGENIERÍA DEL SOFTWARE. UN ENFOQUE PRÁCTICO, 2010)

Si consideramos los testimonios anteriores podemos ver que son testimonios bastante concluyentes con respecto a la importancia del personal en el proceso de ingeniería de software, los vicepresidentes ejecutivos de ingeniería hasta los profesionales de nivel inferior generalmente dan por hecho que es el personal. Los administradores indican al igual que en los ejemplos anteriores que las personas son lo importante, sin embargo, muchas veces sus acciones contradicen sus palabras.

² Institute of Electrical and Electronics Engineers - Instituto de Ingenieros Eléctricos y Electrónicos



Si pensamos en las personas que participan en el proceso de software y la forma en la que se organizan para realizar ingeniería de software efectiva encontramos las siguientes afirmaciones:

Otro punto que considerar son los Líderes de equipo, como Edgemon afirma: “Por desgracia, y por muy frecuente que parezca, los individuos simplemente se topan con el papel de gerente de proyecto y se convierten en gerentes accidentales de proyecto”. (Edgemon, J., “Right Stuff: How to Recognize It When Selecting a Project Manager”, vol. 2, núm. 5,, mayo 1995)

En el libro *On Becoming a Technical Leader*, Dorset House (1986), acerca del liderazgo técnico, escrito por Jerry Weinberg sugiere un modelo MOI de liderazgo:

Weinberg sugiere que los líderes de proyecto exitosos aplican un estilo administrativo de resolución de problemas. Es decir, un gerente de proyecto de software debe concentrarse en comprender el problema que se va a resolver, en administrar el flujo de ideas y, al mismo tiempo, en dejar que todos en el equipo sepan (por medio de palabras y, mucho más importante, con acciones) que la calidad cuenta y que no se comprometerá. (Roger S. Pressman, *INGENIERÍA DEL SOFTWARE. UN ENFOQUE PRÁCTICO*, 2010)

1.2.- Ciclo de vida de un producto: etapas y pruebas asociadas

Para la estrategia de pruebas, debemos considerar casos de prueba los que ayudaran a detectar errores, normalmente los casos de prueba se basan en los requerimientos.

Consideraremos un ciclo de vida tradicional, como se muestra en la figura siguiente:

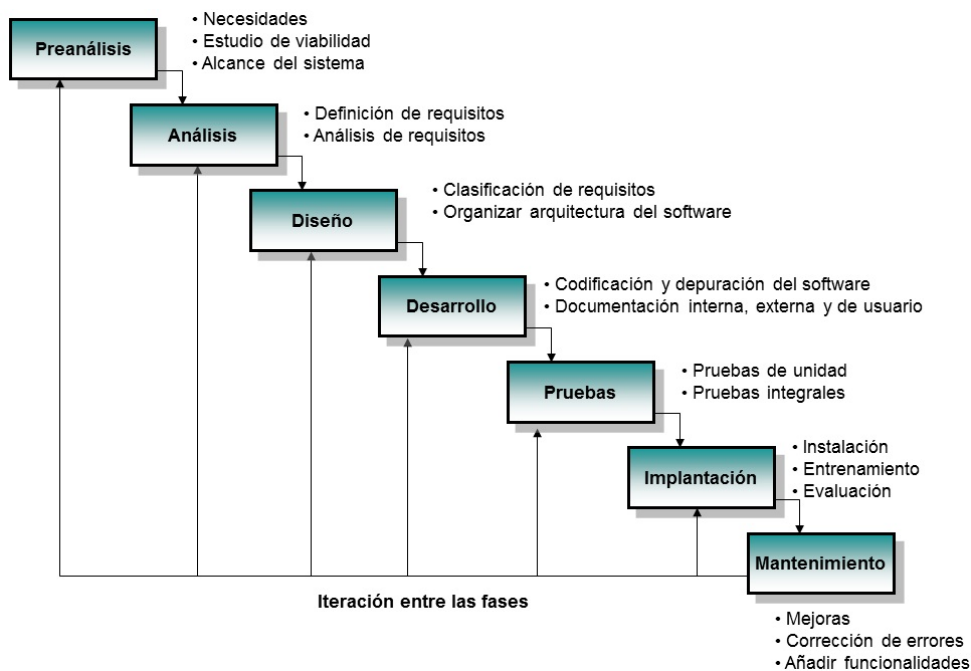


Figura 2: Modelo clásico de desarrollo
Fuente: (<http://2.bp.blogspot.com/>, s.f.)

Si consideramos la tres primeras etapas, debemos utilizar técnicas estáticas que no necesitan ejecutar la aplicación, por el contrario, lo que se realiza es un análisis estático, primero se **evalúan todos los documentos que han emanado**, por ejemplo, definición de requerimientos, especificación de requerimientos, diseños realizados e



incluir el código fuente. Lo que se busca es detectar de manera temprana errores tanto en el diseño como en el código.

Como ejemplo de los errores mencionados podríamos encontrar:

- Error en la lógica de programación.
- Un requerimiento que se encuentre mal especificado.

¿Qué pruebas podemos utilizar?

- Efectuar inspecciones visuales.
- Pruebas de escritorio.
- Caja blanca.

Análisis Estático basado en Herramientas. Defectos típicamente detectados:

- Variables mal definidas o utilizadas.
- Errores en interfaces.
- Código inaccesible.
- Lógica errónea.
- Estándares incumplidos.
- Seguridad vulnerable.
- Errores de sintaxis.


Fuente: (<http://dinamizaconsulting.com/istqb/>, s.f.)

Para la **etapa de desarrollo**, en la que ya contamos con parte de nuestro código que se va ejecutando podemos considerar técnicas dinámicas, por ejemplo, las siguientes pruebas:

- Pruebas unitarias.
- Caja negra.
- Regresión.

Y en la **etapa de pruebas**, las siguientes:

- Integración.
- Funcionales.
- No funcionales.

-
- 
- Prueba Alpha.
 - Prueba Beta.
 - Pruebas de sistema.
 - Pruebas de Aceptación.

Estas pruebas permiten verificar que los componentes de la aplicación cumplen con las características del modelo y con la funcionalidad del sistema. La aplicación debe responder a lo deseado. También se corrigen fallos y posibles vulnerabilidades. Terminados estos análisis, se ejecuta el proceso de implantación.

Fuente: (<https://www.applicatta.cl/>, s.f.)

Para el **proceso de implantación** se realizan las pruebas de todos los elementos con una versión definitiva del software, se consideran pruebas de integración con otros módulos o sistemas con los que interactúe.

Finalmente, para la etapa de mantenimiento, se realizan ajustes en base a la detección de posibles errores, debiéndose efectuar pruebas de regresión, para asegurarse de que se ha corregido el error y que no se ha afectado con estos cambios otra parte del sistema.

En este caso se ha considerado un modelo de desarrollo clásico, como sabemos existen otros modelos de desarrollo que cuentan con distintas etapas, por lo que dependiendo del modelo que estemos utilizando debemos escoger las pruebas.

2. Relación entre calidad de software como producto con su comercialización y rendimiento en el mercado informático.

2.1.- Cumplimiento de requerimientos del mercado

¿Cómo garantizamos un software de calidad?

Para garantizar la calidad de software es importante implementar algún modelo o estándar de calidad que permita la gestión de

atributos en el proceso de construcción de software, teniendo en cuenta que la concordancia de los requisitos y su construcción son la base de las medidas de calidad establecidas. Fuente: (<https://www.redalyc.org/>, s.f.)

El término calidad de software hace mención del nivel de desempeño de las principales características que debe cumplir un software en su ciclo de vida, ellas de cierto modo aseguran que el cliente disponga de un sistema confiable, lo que incrementa su satisfacción ante la funcionalidad y eficiencia del sistema desarrollado.

Los modelos de calidad de software normalmente se encuentran estructurados como se puede apreciar en la Figura 3, se pueden observar los diversos factores de calidad que a su vez se conforman de criterios que son evaluados por métricas, con la finalidad de abordar la evaluación desde una visión desde lo más general a lo más particular, y lograr así la disminución de la subjetividad en la asignación de un valor, ya sea cuantitativo o cualitativo.

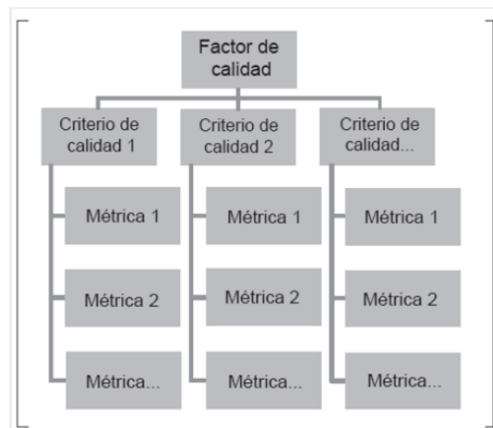


Figura 3: Estructura de la calidad de software.
Fuente: (<https://www.redalyc.org/>, s.f.)

De esta misma forma, los modelos de calidad de software son clasificados según el enfoque de evaluación, a nivel de proceso, producto o calidad en uso.

Si nos enfocamos en la **calidad en uso**: Es importante destacar que, aunque en distintos escenarios se usan los términos usabilidad y calidad en uso, con la misma finalidad y de forma intercambiable son diferentes, fundamentalmente debido a que el concepto de calidad



en uso es más amplio y comprende más elementos que la usabilidad (Covella, 2005) y esta última es una de las características de calidad de un producto software. **La calidad en uso** es definida como "conjunto de atributos relacionados con la aceptación por parte del usuario final y seguridad", y está basada en la eficacia, productividad, seguridad y satisfacción, según ISO/IEC 9126.

FURPS: El modelo FURPS ha sido utilizado para el diseño y validación de interfaces para usuarios finales, evaluando su funcionalidad, usabilidad, confiabilidad, desempeño y soporte, para tener como salida final un producto que cumpla las reglas del negocio (Eeles, 2005), es así como se ha utilizado como un clasificador de requisitos, ayudando a la asignación correcta de requisitos, implementación, y diseño de interfaces. En la Figura 4 se presentan algunas empresas que han acogido e implementado el modelo FURPS

Empresa	Área	País
Universidad De Santander	Académico y diseño de interfaces	Colombia
Hewlett Packard	Creación y ensamblaje de equipos de computo	Estados unidos
Universidad de Santander	Académico	Colombia
IBM Rational Software Company	Desarrollo de software	Estados unidos
IBM	Desarrollo de software	Estados unidos

Figura 4 Implementación del modelo FURPS
Fuente: (<https://www.redalyc.org/>, s.f.)

2.2.- Relaciona verificación y validación con certificación de calidad de software

¿Qué relación existe entre la verificación y validación respecto a la calidad del software?

La verificación esta enfocada al proceso de evaluar el sistema o partes de él, es decir, sus componentes, permite establecer si los productos de una etapa específica del desarrollo cumplen las condiciones definidas al inicio de la fase. **La validación** es la evaluación del software al



terminar el proceso de desarrollo de software para establecer su conformidad con los requisitos IEEE.

Un proceso asociado a la verificación y validación es la depuración, que localiza y corrige los errores descubiertos durante la V&V.

- Es un proceso complejo ya que no siempre los errores son detectados cerca del lugar en que se generaron.
- Se usan herramientas de depuración, que simplifican el proceso.

Posteriormente a la reparación del error, es necesario volver a probar el sistema lo que se conoce como pruebas de regresión, ya que cuando se realiza la solución al primer fallo puede generar nuevos fallos.



Figura 5: Verificación y validación

Fuente: (<https://www.ctr.unican.es/>, s.f.)

Si consideramos la calidad en el desarrollo de software, podemos apreciar que al incluir **la depuración** que corresponde al proceso de eliminación de los errores que se descubren durante las fases de prueba, con ello estamos minimizando la ocurrencia de errores, cabe destacar que este es un proceso independiente que no tiene por qué estar integrado:

- Verificación & validación: Establece existencia de defectos en el programa.
- Depuración: Proceso que localiza el origen y corrige los defectos.

Hemos visto como podemos mejorar la calidad en nuestro desarrollo, pero además del proceso en sí, debemos considerar las certificaciones de calidad por medio de las cuales es posible asegurar la eficacia y eficiencia de los procesos. La mayor parte de las certificaciones generan un impacto al interior de las organizaciones, por una parte, les



exigen pensar en las mejores maneras de lograr objetivos previamente a certificar y, por otro lado, a proceder de manera predecible posteriormente, basando las decisiones sobre información certera. Las certificaciones de calidad añaden valor al producto, previsibilidad al trabajo, y confianza a los clientes, esto permite aumentar el nivel de competitividad de la empresa. Se considera como un aspecto valioso de la empresa que de a conocer su calidad y muestra cómo trabaja ante organismos de certificación externos o internacionales.

3. Relaciona verificación y validación con certificación de calidad de software.

3.1.- Calidad de los productos

“La calidad de un producto y/o servicio y el cliente o consumidor. Es un conjunto de características o propiedades inherentes, que tiene un producto o servicio las cuales satisfacen las necesidades del cliente, las mismas que se ven reflejadas en una sensación de bienestar de complacencia”.

Fuente: (<http://centrocastelmonte.com/>, s.f.)

Se deben estandarizar las características y mantenerlas en el tiempo, es decir, cada vez que se compra el producto estén presentes, también deben poseer un costo óptimo.

Para realizar exitosamente la gestión de la calidad hay que considerar al menos cuatro factores:

- a) Debe haber un compromiso entre la dirección y su personal ejecutivo.
- b) Disponer de personal que se encuentre involucrado y además informado.
- c) Realizar capacitaciones al personal.
- d) Mantener contacto con los clientes.



3.2.- Calidad del software como producto

Calidad se entiende como el grado en el que el producto software integra un conjunto de características, determinadas por la industria, de forma que se asegura su eficiencia de uso, en relación con los requerimientos de los clientes.

Es decir, calidad del software está asociado con el nivel en el que un cliente percibe que el software cumple con sus expectativas.

Se han formulado muchas definiciones sobre el concepto de calidad en el software. Para ello debemos responder la pregunta ¿qué es calidad en el software? Seguramente la primera respuesta en qué pensaría la mayoría de las personas es:

La calidad en el software está en relación directa con el cumplimiento de los requerimientos formulados por el usuario, de tal forma que si un programa no cumple con alguno de estos requerimientos es un software de baja calidad.

Aunque el criterio de cumplimiento de los requerimientos es un factor importante, no es el único, ya que existen condiciones implícitas que el software debe cumplir como son eficiencia, seguridad, integridad, consistencia, etc.; por lo tanto, no se puede afirmar que un software es de alta calidad cuando cumple con los requerimientos del usuario, pero:

- No es eficiente al utilizar los recursos de la máquina (programas muy lentos).
- O no es confiable; los resultados que entrega varían, no son siempre iguales al procesar los mismos datos.
- O no es fácil de utilizar.
- O no es seguro.
- O no es fácil hacerle mantenimiento.

La calidad en el software es una mezcla compleja de ciertos factores que varían de acuerdo con el usuario y con los tipos de aplicación.



Podemos resumir el concepto de la calidad en el software por medio de los siguientes puntos:

- 1) Los requerimientos del usuario sobre un programa son los fundamentos desde los que se mide la calidad.
- 2) La falta de concordancia con estos requerimientos es una falta de calidad.
- 3) Los estándares especificados definen un conjunto de criterios de desarrollo que guían la forma como se aplica la ingeniería de software; si no se siguen estos estándares, probablemente se obtendrá software de baja calidad.
- 4) Existe un conjunto de requerimientos implícitos que a menudo no se mencionan (eficiencia, facilidad de uso, facilidad de mantenimiento). Si el software falla en alcanzar los requerimientos implícitos, la calidad en el software queda en entredicho.

Fuente: (Bahamon, 2010)

El concepto de calidad frente al desarrollo de software es diferente a lo que podemos considerar como calidad de la manufactura de un producto, la percepción además es distinta, si pensamos en un producto que ha sido elaborado al comprarlo podemos saber inmediatamente si cumple con nuestras expectativas, por ejemplo, si nos compramos un abrigo y nos quita el frío entonces cumplió las expectativas, si tenemos sed por medio de una bebida nos refresca, pero en el caso de un software es distinto, como el cliente o usuario lo percibe, será a través de los resultados que entregue, reflejado en menos tiempo de trabajo, cálculos precisos, etc.... y esto es importante porque hará la diferencia en que lo vea como un gasto o una inversión.

3.3.- Características de calidad del software

La calidad es una cualidad y propiedad inherente de las cosas, que permite que estas sean comparadas con otras de su misma especie. La definición de calidad nunca puede ser precisa, ya que se trata de una apreciación subjetiva.



Las tecnologías de información, por su parte, hablan de la calidad de datos al momento de comprobar que los datos capturados, procesados, almacenados y entregados son un fiel reflejo de la realidad.

- Funcionalidad: el usuario puede utilizar el software y cumple sus objetivos.
- Confiabilidad: sus datos son íntegros.
- Usabilidad: fácil de usar y fácil de aprender a usar.
- Portabilidad: ejecutable en diferentes plataformas.
- Compatibilidad: capacidad que tienen dos sistemas de trabajar uno con otro.
- Mantenibilidad: capacidad de ser modificado.
- Eficiente: hace lo que debe hacer, bien, a tiempo y no malgasta recursos.
- Robustez: sistema fuerte, sin debilidades ni vacíos de seguridad.
- Disponibilidad: sistema y datos se pueden acceder en cualquier momento.

Métricas de control de la calidad en el software

Se define como métrica el valor asociado con la respuesta a una pregunta formulada en una revisión para evaluar o establecer un atributo o un requerimiento de un criterio o subcriterio relacionado con un factor.

Aspecto	Factor
Operación del producto	Cumplimiento Exactitud Eficiencia Integridad Facilidad de uso
Revisión del producto	Facilidad de mantenimiento Facilidad de prueba Flexibilidad
Transición del producto	Portabilidad Reusabilidad Interoperatividad

Figura 6: clasificación sugerida por J.A. McCa" en su libro Factors in Software Quality,

Fuente: (Bahamon, 2010)



Factor calidad	Definición
Cumplimiento	El grado en que un programa satisface sus especificaciones y consigue los objetivos de la misión encomendada por el cliente.
Fiabilidad	El grado en que se puede esperar que un programa lleve a cabo sus funciones esperadas con la precisión requerida.
Eficiencia	La cantidad de recursos de hardware y de código requerido por un programa para realizar su función.
Integridad	El grado en que puede controlarse el acceso al software o a los datos por personas no autorizadas.
Facilidad de uso	El esfuerzo requerido para aprender, trabajar, preparar la entrada e interpretar la salida de un programa.
Facilidad de mantenimiento	El esfuerzo requerido para localizar y arreglar un error en un programa.
Facilidad de prueba	El esfuerzo requerido para probar un programa de forma que se asegure que realiza la función requerida.
Portabilidad	El esfuerzo requerido para transferir el programa desde una configuración de hardware o sistema operativo a otro.
Reusabilidad	El grado en que un programa (o partes de él) se pueden reutilizar en otras aplicaciones.
Facilidad de interoperación	El esfuerzo requerido para acoplar un sistema a otro.

Figura 7:

cada factor

Fuente: (Bahamon, 2010)

Descripción de



Facilidad de auditoría	Facilidad con que se puede comprobar la conformidad con los estándares.
Exactitud	La precisión en los cálculos y el control.
Normalización de las comunicaciones.	El grado en que se usan el ancho de banda, los protocolos y las interfases estándar.
Complejidad	El grado en que se ha conseguido la total implementación de las funciones requeridas.
Concisión	Lo compacto que es el programa en términos de líneas de código.
Consistencia	El uso de un diseño uniforme y de técnicas de documentación.
Estandarización datos	El uso de estructuras de datos y de tipos datos estándar.
Tolerancia de errores	El daño que se produce cuando el programa encuentra un error.
Eficiencia en la ejecución	El rendimiento en tiempo de ejecución de un programa.
Facilidad de expansión	El grado en que se puede ampliar el diseño arquitectónico de datos.
Generalidad	La amplitud de aplicación potencial de los componentes del programa.
Independencia del hardware	El grado en que el software es independiente del hardware que usa.
Instrumentación	El grado en que el programa muestra su propio funcionamiento e identifica errores que aparecen.
Modularidad	Independencia funcional de los componentes del programa.
Facilidad de operación	Grado de facilidad de operación.
Seguridad	La disponibilidad de mecanismos que controlen o protejan los programas o los datos.
Autodocumentación	El grado en que el código fuente proporciona documentación significativa.
Simplicidad	El grado en que un programa puede ser entendido sin dificultad.
Facilidad de trazo	La posibilidad de seguir la pista de la representación del diseño de los componentes reales del programa hacia atrás.
Formación	El grado en que el software ayuda a permitir que nuevos usuarios apliquen el sistema.

Figura 8: Lista de Fuente:

criterios
(Bahamon, 2010)



Factor de calidad \ Métrica de calidad del software	Corrección	Fiabilidad	Eficacia	Integridad	Facilidad de mantenimiento	Flexibilidad	Facilidad de prueba	Portabilidad	Reusabilidad	Interoperabilidad	Facilidad de uso
Facilidad de auditoría				X			X				
Exactitud		X									
Normalización de las comunicaciones										X	
Complejidad	X										
Complejidad		X				X	X				
Concisión			X		X	X					
Consistencia	X	X			X	X					
Estandarización en los datos										X	
Tolerancia de errores		X									
Eficiencia en la ejecución			X								
Facilidad de expansión						X					
Generalidad						X		X	X	X	
Indep. del hardware								X	X		
Instrumentación				X	X		X				
Modularidad		X			X	X	X	X	X	X	
Facilidad de operación			X								X
Seguridad				x							
Auto-documentación					X	X	X	X	X		
Simplicidad		X			X	X	X				
Indep. del sistema								X	X		
Facilidad de traza	X										
Formación											X

Figura 9:

entre los criterios y los factores
Fuente: (Bahamon, 2010)

Relación



Figura 10:

negativas entre factores

Fuente: (Bahamon, 2010)

Relaciones

4. Determina requerimientos de calidad de software, según estándares ISO.

4.1.- Calidad de un software según estándares ISO

Como hemos visto la calidad en el desarrollo de software en muchos aspectos no se asemeja a la calidad en el proceso de manufactura de un producto.

Hoy en día en las organizaciones la utilización de software ha tomado un rol estratégico, los procesos mas importantes se basan en ellos, vemos que la supervivencia de estas organizaciones depende del correcto funcionamiento de estos sistemas.

El objetivo general de la creación del estándar ISO/IEC 3 25000 SQuaRE (System and Software Quality Requirements and Evaluation) es organizar, enriquecer y unificar las series que cubren dos procesos principales: especificación de requisitos de calidad del software y

³ *ISO (International Organization for Standardization) e IEC (International Electrotechnical Commission)



evaluación de la calidad del software, soportada por el proceso de medición de calidad del software.

Fuente: (<https://sites.google.com/>, s.f.)

ISO/IEC 25000, conocida como SQuaRE (System and Software Quality Requirements and Evaluation), 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 familia ISO/IEC 25000 es el resultado de la evolución de otras normas anteriores, especialmente de las normas ISO/IEC 9126, que describe las particularidades de un modelo de calidad del producto software, e ISO/IEC 14598, que abordaba el proceso de evaluación del producto software. Fuente: (<https://iso25000.com/>, s.f.)

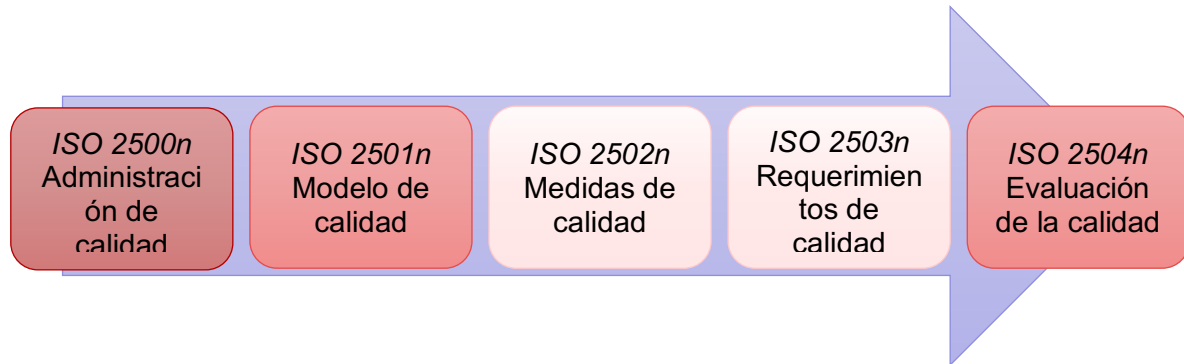
Los beneficios de utilizar SQuare son:

- 1) El modelo representa la calidad esperada del producto de software.
- 2) Establecimiento de la separación de las necesidades o expectativas en calidad en uso, calidad externa y calidad interna.
- 3) Permite una mayor eficacia en la definición del software.
- 4) Plantea la evaluación de productos intermedios.
- 5) Propone una calidad final a través de las evaluaciones intermedias.
- 6) Permite efectuar un rastreo entre las expectativas, requisitos y medidas de evaluación.
- 7) Mejora la calidad del producto.

Fuente: (<https://www.ecured.cu/>, s.f.)

Ideas clave

Agrupación de 5 tópicos, conformado por 14 documentos, pertenecientes a SQuaRE en su estándar de requerimientos de calidad.

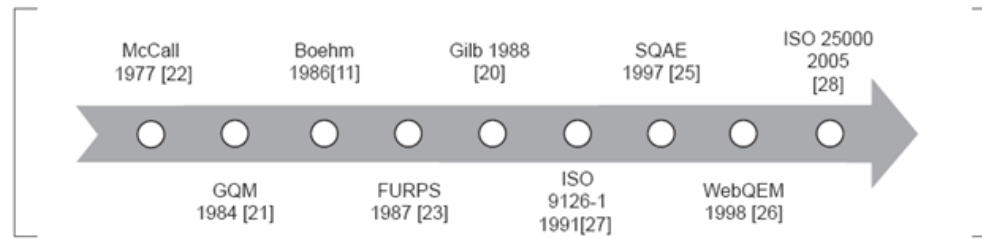


El modelo de calidad del producto de la ISO 25000 clasifica la calidad del producto en ocho características: funcionalidad, rendimiento, compatibilidad, usabilidad, fiabilidad, seguridad, mantenibilidad y portabilidad.



Fuente: (<https://sites.google.com/>, s.f.)

Línea de tiempo de algunos modelos de calidad de software a nivel de producto.



Fuente: (<https://www.redalyc.org/>, s.f.)

10 pasos que garantizan un proyecto de software exitoso



- ✓ 1.- **Demostración del software de partida:** Siempre que afrontamos un nuevo proyecto de software, tomamos como base de partida un software del que aprovechamos módulos que tengan la misma o parecida funcionalidad que los requeridos por nuestro cliente. La mayor parte de las veces ofrecemos la solución completa, a falta de personalizar.
- ✓ 2.- **Análisis de requerimientos:** Realizamos un análisis detallado de los requerimientos de cada departamento.
- ✓ 3.- **Elaboración de propuesta:** Elaboramos una propuesta por escrito de las cosas a desarrollar, el tiempo estimado de desarrollo y el precio final del proyecto.
- ✓ 4.- **Presentación de propuesta:** Presentamos nuestra propuesta a la gerencia y a los distintos departamentos para su aprobación y firma.
- ✓ 5.- **Establecimiento de calendario y Firma del proyecto:** Firma del proyecto con detalle de los hitos, fecha de comienzo, fecha de puesta en explotación y alcance económico del mismo.
- ✓ 6.- **Asignación de recursos al proyecto:** A cada proyecto se le asigna un jefe de proyecto, un consultor, y uno o varios programadores, dependiendo de la dificultad o los tiempos de implantación acordados.
- ✓ 7.- **Instalación de Software de partida.** Formación. Inicio de los desarrollos a medida.
- ✓ 8.- **Instalación, prueba y revisión de los desarrollos a medida.** Formación: Redacción de documentación. Modificaciones. Aprobación de la funcionalidad de los desarrollos a medida.
- ✓ 9.- **Puesta en marcha en los tiempos acordados:** Los dos primeros días de puesta en marcha , el jefe de proyecto asignado estará presente para resolver dudas de última hora y "mitigar miedos a meter la pata".
- ✓ 10.- **Vigilancia:** Durante los dos primeros meses de puesta en marcha del proyecto se establece un sistema de vigilancia para verificar que todo funciona según lo acordado.

Fuente: (<https://www.unybase.com/>, s.f.)

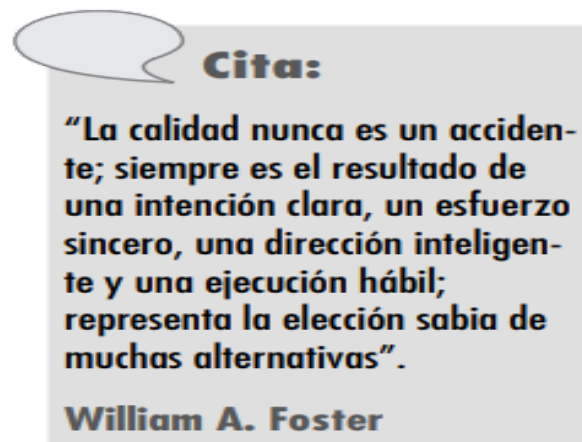


Conclusiones

Se observa que el proceso de pruebas impacta en los riesgos del producto software; por ende, para las empresas de software es vital formular y adecuar un buen proceso de pruebas de calidad de software. (Mera-Paz, 2016)

El término calidad de software hace referencia al nivel de desempeño de las principales características con las que debe cumplir un sistema computacional durante su ciclo de vida, estas características de alguna forma dan garantías de que el cliente va a disponer de un sistema confiable, lo que incrementa su satisfacción ante la funcionalidad y eficiencia del sistema desarrollado.

Hemos visto que, gracias a las pruebas, es posible medir la calidad de un software en términos de los defectos que se han detectado, esto en base a los requisitos y características funcionales y no funcionales, como son la fiabilidad, usabilidad, eficiencia, mantenibilidad y portabilidad.



Fuente: (Roger S. INGENIERÍA DEL SOFTWARE. UN ENFOQUE PRÁCTICO, 2010)

Pressman,

Referencias bibliográficas

Bahamon, L. J. (2010). *CONTROL DE CALIDAD EN EL SOFTWARE*. Publicaciones Icesi.

Blanco-Llano, Javier, & Rodríguez-Hernández, Aida (2011). *REVISIÓN, VERIFICACIÓN Y VALIDACIÓN EN UN PROCESO DE DESARROLLO DE SOFTWARE*. *Ingeniería Industrial*, XXXII (1), 28-36. [fecha de Consulta 24 de Julio de 2021]. ISSN: 0258-5960. Disponible en: <https://www.redalyc.org/articulo.oa?id=360433575005>

Covella, G. J. (2005). *Medición y evaluación de calidad en uso de aplicaciones web*. Tesis Doctoral. Facultad de Informática. Universidad Nacional de La Plata.

Edgemon, J., "Right Stuff: How to Recognize It When Selecting a Project Manager", vol. 2, núm. 5, (mayo 1995). En J. Edgemon, *Right Stuff: How to Recognize It When Selecting a Project Manager* (págs. 37-42). Application Development Trends.

Eeles, P. (2005). *Capturing architectural requirements*. IBM Rational developer works.

<http://dinamizaconsulting.com/istqb/>. (s.f.). Obtenido de <http://dinamizaconsulting.com/istqb/T4-Tecnicas%20estaticas.pdf>

<https://www.ctr.unican.es/>. (s.f.). Obtenido de https://www.ctr.unican.es/asignaturas/Ingenieria_Software_4_F/Doc/M7_09_VerificacionValidacion-2011.pdf

<https://www.ecured.cu/>. (s.f.). Obtenido de https://www.ecured.cu/ISO/IEC_25000

Mera-Paz, J. A. (2016). *Análisis del proceso de pruebas de calidad de software*. Popayán, Colombia: Ingeniería Solidaria, vol. 12, no. 20, pp. 163-176, oct. 2016.

Roger S. Pressman, P. (2010). *INGENIERÍA DEL SOFTWARE. UN ENFOQUE PRÁCTICO*. McGRAW-HILL INTERAMERICANA EDITORES, S.A. DE C.V.