Normas y estándares de calidad para el desarrollo de SW

Sistemas de Calidad en T.I.

Normas y estándares de calidad para el desarrollo de SW

Sistemas de Calidad en T.I.

Introducción

Los estándares de calidad de software hacen parte de la ingeniería de software, utilización de estándares y metodologías para el diseño, programación, prueba y análisis del software desarrollado, con el objetivo de ofrecer una mayor confiabilidad, mantenibilidad en concordancia con los requisitos exigidos, con esto se eleva la productividad y el control en la calidad de software, parte de la gestión de la calidad se establecen a mejorar su eficacia y eficiencia.

En un escenario en el que los sistemas de software se desarrollan y construyen por terceros proveedores, el contratante del servicio, como primer receptor del mismo, en muchos casos debe confiar en el buen hacer del proveedor seleccionado, especialmente si nos dispone de los medios apropiados para auditar la entrega y en su caso argumentar defectos en el proceso de desarrollo.

En general, una vez validado que el sistema responde a los principales requisitos funcionales especificados, el usuario realizará las pruebas de aceptación, corrigiendo los errores encontrados y tas pasándose al fin del entorno de producción. Sin embargo, en muy pocas ocasiones se validan de manera rigurosa los requisitos funcionales y los no funcionales, o se ejecutan validaciones que aseguren que el sistema es lo suficientemente robusto y estable como para pasar a un entorno productivo con las garantías adecuadas (Fernando Arciniega, 2017).

Contenido

Antecedentes

El control y aseguramiento de la calidad son actividades esenciales para cualquier negocio que genere productos que utilicen otras personas. Antes del siglo XX, el control de calidad era responsabilidad única del artesano que elaboraba el producto. Cuando pasó el tiempo y las técnicas de la producción en masa se hicieron comunes, el control de calidad se convirtió en una actividad ejecutada por personas diferentes de aquellas que elaboraban el producto.

La primera función formal de aseguramiento y control de la calidad se introdujo en los laboratorios Bell en 1916 y se difundió con rapidez al resto del mundo de la manufactura. Durante la década de 1940, sugirieron enfoques más formales del control de calidad. Éstos se basaban en la medición y en el proceso de la mejora continua como elementos clave de la administración de la calidad.

Actualmente, toda compañía tiene mecanismos para asegurar la calidad en sus productos. En realidad, en las últimas décadas, las afirmaciones explícitas del compromiso de una compañía con la calidad se han vuelto un mantra de la mercadotecnia.

La historia del aseguramiento de la calidad en el desarrollo del software corre de manera paralela con la historia de la calidad en la manufactura del hardware. En los primeros días de la computación (décadas de 1950 y 1960), la calidad era responsabilidad única del programador.

Los estándares para asegurar la calidad del software se introdujeron en los contratos para desarrollar software militar en la década de 1970 y se extendieron con rapidez al desarrollo de software en el mundo comercial.

El alcance de la responsabilidad del aseguramiento de la calidad se caracteriza mejor si se parafrasea un comercial de un automóvil popular: "La calidad es el empleo número 1." La implicación para el software es que muchas entidades diferentes tienen responsabilidad en el aseguramiento de la calidad del software: ingenieros de software, gerentes de proyecto, clientes, vendedores y los individuos que trabajan en el grupo de ACS.

Calidad del software

Incluso los desarrolladores de software más experimentados estarán de acuerdo en que obtener software de alta calidad es una meta importante. Pero, ¿cómo se define la calidad del software?

En el sentido más general se define como: **Proceso eficaz de software que se aplica de manera que crea un producto útil que proporciona valor medible a quienes lo producen y a quienes lo utilizan.**

Hay pocas dudas acerca de que la definición anterior podría modificarse o ampliarse en un debate sin fin. La misma sirve a fin de enfatizar tres puntos importantes:

1. Un proceso eficaz de software establece la infraestructura que da apoyo a cualquier esfuerzo de elaboración de un producto de software de alta calidad. Los aspectos de administración del proceso generan las verificaciones y equilibrios que ayudan a evitar que el proyecto caiga en el caos, contribuyente clave de la mala calidad

2. Un producto útil entrega contenido, funciones y características que el usuario final desea; sin embargo, de igual importancia es que entrega estos activos en forma confiable y libre de errores. Un producto útil siempre satisface los requerimientos establecidos en forma explícita por los participantes. Además, satisface el conjunto de requerimiento (por ejemplo, la facilidad de uso) con los que se espera que cuente el software de alta calidad.
3. Al agregar valor para el productor y para el usuario de un producto, el software de alta calidad proporciona beneficios a la organización que lo produce y a la comunidad de usuarios finales. La organización que elabora el software obtiene valor agregado porque el software de alta calidad requiere un menor esfuerzo de mantenimiento, menos errores que corregir y poca asistencia al cliente.
   Esto permite que los ingenieros de software dediquen más tiempo a crear nuevas aplicaciones y menos a repetir trabajos mal hechos. El resultado final es:
   a. mayores utilidades por el producto de software
   b. más rentabilidad cuando una aplicación apoya un proceso de negocios
   c. mejor disponibilidad de información, que es crucial para el negocio.

> *"La diferencia entre algo que puede salir mal y algo que posiblemente no salga mal es que cuando esto último sale mal, por lo general es imposible corregirlo o repararlo."*
>
> **Douglas Adams**

Dimensiones de la calidad de Garvin

David Garvin sugiere que la calidad debe tomarse en cuenta, adoptando un punto de vista multidimensional que comience con la evaluación de la conformidad y termine con una visión trascendental (estética).

Aunque las ocho dimensiones de Garvin de la calidad no fueron desarrolladas específicamente para el software, se aplican a la calidad de éste:

- **Calidad del desempeño.** ¿El software entrega todo el contenido, las funciones y las características especificadas como parte del modelo de requerimientos, de manera que da valor al usuario final?
- **Calidad de las características.** ¿El software tiene características que sorprenden y agradan la primera vez que lo emplean los usuarios finales?
- **Confiabilidad.** ¿El software proporciona todas las características y capacidades sin fallar? ¿Está disponible cuando se necesita? ¿Entrega funcionalidad libre de errores?
- **Conformidad.** ¿El software concuerda con los estándares locales y externos que son relevantes para la aplicación? ¿Concuerda con el diseño de facto y las convenciones de código? Por ejemplo, ¿la interfaz de usuario está de acuerdo con las reglas aceptadas del diseño para la selección de menú o para la entrada de datos?

- **Durabilidad.** ¿El software puede recibir mantenimiento (cambiar) o corregirse (depurarse) sin la generación inadvertida de eventos colaterales? ¿Los cambios ocasionarán que la tasa de errores o la confiabilidad disminuyan con el tiempo?
- **Servicio.** ¿Existe la posibilidad de que el software reciba mantenimiento (cambios) o correcciones (depuración) en un periodo de tiempo aceptablemente breve? ¿El equipo de apoyo puede adquirir toda la información necesaria para hacer cambios o corregir defectos?
- **Estética.** No hay duda de que todos tenemos una visión diferente y muy subjetiva de lo que es estético. Aun así, la mayoría de nosotros estaría de acuerdo en que una entidad estética posee cierta elegancia, un flujo único y una "presencia" obvia que es difícil de cuantificar y que, no obstante, resulta evidente. El software estético tiene estas características.
- **Percepción**. En ciertas situaciones, existen prejuicios que influirán en la percepción de la calidad por parte del usuario. Por ejemplo, si se introduce un producto de software elaborado por un proveedor que en el pasado ha demostrado mala calidad, se estará receloso y la percepción de la calidad del producto tendrá influencia negativa. De manera similar, si un vendedor tiene una reputación excelente se percibirá buena calidad, aun si ésta en realidad no existe.

Factores de la calidad de McCall

Richards y Walters proponen una clasificación útil de los factores que afectan la calidad del software, se centran en tres aspectos importantes del producto de software: sus características operativas, su capacidad de ser modificado y su adaptabilidad a nuevos ambientes.

- **Corrección**. Grado en el que un programa satisface sus especificaciones y en el que cumple con los objetivos de la misión del cliente.
- **Confiabilidad**. Grado en el que se espera que un programa cumpla con su función y con la precisión requerida.
- **Eficiencia**. Cantidad de recursos de cómputo y de código requeridos por un programa para llevar a cabo su función.
- **Integridad**. Grado en el que es posible controlar el acceso de personas no autorizadas al software o a los datos.
- **Usabilidad**. Esfuerzo que se requiere para aprender, operar, preparar las entradas e interpretar las salidas de un programa.
- **Facilidad de recibir mantenimiento**. Esfuerzo requerido para detectar y corregir un error en un programa.
- **Flexibilidad**. Esfuerzo necesario para modificar un programa que ya opera.
- **Susceptibilidad de someterse a pruebas.** Esfuerzo que se requiere para probar un programa a fin de garantizar que realiza la función que se pretende.
- **Portabilidad**. Esfuerzo que se necesita para transferir el programa de un ambiente de sistema de hardware o software a otro.

- **Reusabilidad**. Grado en el que un programa (o partes de uno) pueden volverse a utilizar en otras aplicaciones.
- **Interoperabilidad**. Esfuerzo requerido para acoplar un sistema con otro.

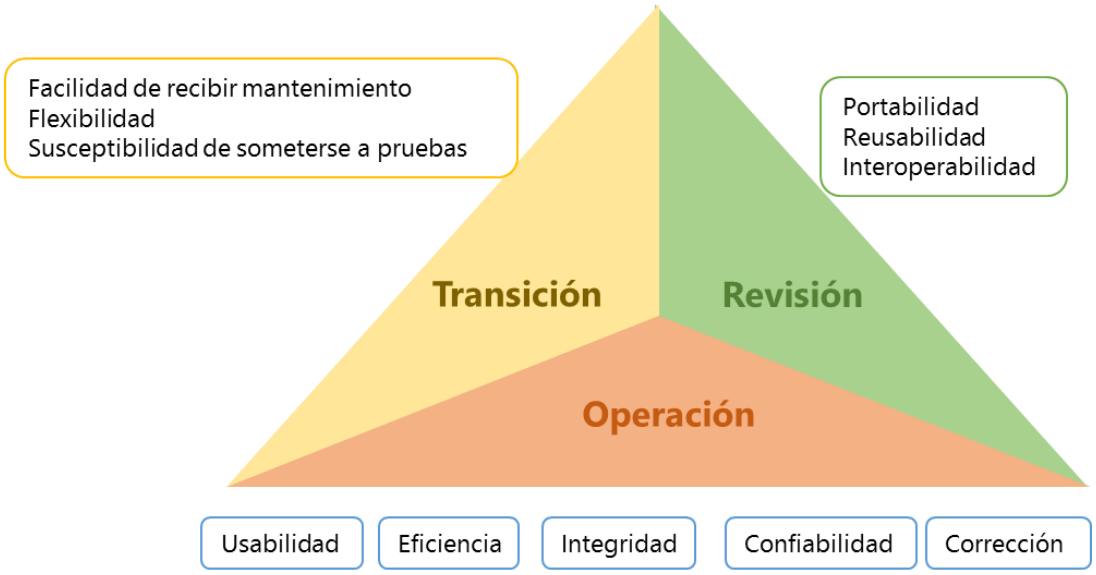

***Figura 1**. Factores de la calidad de McCall.*

Normas y estándares de calidad

Las normas y estándares son importantes para el desarrollo de software y para la realización de proyectos de TI, ya que en cada una de ellas nos darán a conocer las reglas establecidas para poder realizarlas.

Hoy en día la calidad es muy importante para poder satisfacer a los clientes en cada proyecto de TI y desarrollo de Software, también rigen el torno a este mundo para el desarrollo correcto de las aplicaciones de calidad y cumplimiento con las normas y parámetros.

Las **Normas** son todas aquellas reglas que deben ser respetadas, y que permiten el ajuste en ciertas conductas. Pero en cuestión de desarrollo de software y en proyectos de TI, se enfocan más en los procesos por los que tienen que pasar y los estándares que especifican la calidad con la que debe contar.

Mientras tanto los **Estándares**, son el conjunto de reglas que deben de cumplir los procedimientos y ciertas investigaciones que puedan ser compatibles con lo especificado. Además ofrecen muchos beneficios como la reducción de diferencia entre los productos y generan una estabilidad, madurez y calidad en beneficio del consumidor.

A continuación, en el presente documento se darán a conocer cada una de las normas y estándares implementados para el desarrollo de software y para proyectos de TI.

ISO. International Organization for Standardization.

Son normas que tienen como objetivo garantizar al cliente que los productos o servicios adquiridos siempre tendrán las mismas propiedades y características. El propósito de ISO es promover el desarrollo de la estandarización y de las actividades relacionadas del mundo para facilitar el intercambio internacional de mercancías y de servicios, y para desarrollar la cooperación en actividad intelectual, científica, tecnológica y económica.

CMMI. Capability Maturity Model Integration.

Es un modelo de mejoras de procesos de construcción que provee los elementos necesarios para determinar su efectividad. CMMI es el estándar más reconocido para la mejora de procesos para desarrollo de proyectos, gestión de proveedores y gestión del servicio.

PSP. Personal Software Process.

Es un conjunto de prácticas disciplinadas para la gestión del tiempo y mejorar de la productividad personal de los programadores o ingenieros de software. Su principal función, consiste en el registro de la información de todo proceso de desarrollo en formatos, esto para generar estadísticas que se podrán utilizar en futuros desarrollos, mejorando así el desempeño laborar del programador.

TSP. Team Software Process.

Es una metodología para dirigir el trabajo de mejora y desarrollo de software, además establece un entorno donde el trabajo de equipo sea efectivo, normal y natural.

IEEE (Instituto de Ingenieros Electrónicos y Eléctricos).

Está diseñada para servir a los profesionales involucrados en todos los aspectos de los campos eléctricos, electrónicos de computación y áreas afines de la ciencia y la tecnología. Las normas que son establecidas IEEE según el Software Engineering & Tessing son totalmente voluntarias para el desarrollo del software.

MoProsoft.

Es el modelo de Procesos para la industria del software, es un modelo para la mejora y la evaluación de los procesos de desarrollo y mantenimiento de sistemas y productos de software. MoProsoft fue desarrollado por la Asociación Mexicana para la Calidad en Ingeniería de Software y a solicitud de la Secretaria de Economía, esto con la finalidad de obtener una norma mexicana que resulta apropiada

para las características de la mayoría de las empresas mexicanas de desarrollo y mantenimiento de software.

Conclusiones

Gracias a las normas y estándares aplicados a proyectos TI y de calidad para el desarrollo de software hoy en día se nos puede facilitar la realización de los proyectos ya que con las normas podemos seguir ciertos pasos para que los proyectos sean más eficientes y más fáciles de realizarlos paso a paso y los estándares nos especifican que el desarrollo de un proyecto debe ser de calidad, el cual debe satisfacer las necesidades del cliente o de la empresa a la que se le esté desarrollando dicho software.

También gracias a los importantes estándares como el proceso de software personal es de gran ayuda para los ingenieros involucrados en el proyecto ya que les permite mejorar la forma en que trabajan y controlar los tiempos mediante formatos de tiempo para cada una de las actividades y que el software desarrollado sea de calidad.

Por otra parte, el CMMI nos ayuda a mejorar los procesos de construcción de software y de proyectos de TI, el estándar IEEE nos brinda una serie de documentación el desarrollo de software y proyectos de TI Y el TSP se enfoca más en la mejora de trabajo en equipo para los procesos de software.

Por último la aplicación de una norma o estándar los podemos aplicar en nuestros proyectos de acuerdo a la necesidades de dicho proyecto.

Referencias

Fernando Arciniega. (25 de Octubre de 2017). Fernando Arciniega. Obtenido de Fernando Arciniega: http://fernandoarciniega.com/normas-y-estandares-de-calidad-para-el-desarrollo-de-software/

McCall, J., P. Richards y G. Walters, "Factors in Software Quality", tres volúmenes, NTIS AD-A049014, 015, 055, noviembre 1977.

Garvin D., "Competing on the Eight Dimensions of Quality", Harvard Business Review, n oviembre 1987, pp. 101-109. Resumen disponible en www.acm.org/crossroads/ xrds6-4/software.html.

Pressman, R. S., & Troya, J. M. (1988). Ingeniería del software