3-2-2017

BBSOFT

CALIDAD DE SOFTWARE

Capítulo 1, Capítulo 2

Contenido

[CAPÍTULO 1 CALIDAD EN EL SOFTWARE 1](#_Toc474350470)

[1.1 Introducción 1](#_Toc474350471)

[1.2 Calidad de software 1](#_Toc474350472)

[1.2.1 Evolución Histórica 2](#_Toc474350473)

[1.2.2 Lecciones Aprendidas en Desarrollo de Software 3](#_Toc474350474)

[1.3 Calidad versus velocidad de desarrollo 4](#_Toc474350475)

[1.3.1 Estándares 5](#_Toc474350476)

[1.3.2 Creatividad 7](#_Toc474350477)

[1.3.3 Madurez 8](#_Toc474350478)

[1.4.1 Surgimiento y Evolución 8](#_Toc474350479)

[1.4.2 Modelos 9](#_Toc474350480)

[1.4.3 Certificaciones y Evaluaciones 9](#_Toc474350481)

[CAPÍTULO 2 CAUSAS QUE DETERIORAN LA CALIDADEN EL SOFTWARE 11](#_Toc474350482)

[2.1 Introducción 11](#_Toc474350483)

[2.1.1 Definiciones 11](#_Toc474350484)

[2.2 Causa que deterioran la calidad 17](#_Toc474350485)

[2.3 Aspectos sobre los que trabaja para mejorar la calidad. 19](#_Toc474350486)

[2.4 Formato de tratamiento de los temas 23](#_Toc474350487)

Índice de figuras

[Figura 1 Calidad del software 1](#_Toc474349743)

[Figura 2 Calidad a través de los años 2](#_Toc474349744)

[Figura 3 Nivel de calidad 5](#_Toc474349745)

[Figura 4 Estándares 6](#_Toc474349746)

[Figura 5 Área de desarrollo 22](file:///D:\capitulos%201%20y%202.docx#_Toc474349747)

# CAPÍTULO 1 CALIDAD EN EL SOFTWARE

## 1.1 Introducción

El objetivo de este capítulo es presentar el concepto de calidad asociada al desarrollo de software.

## 1.2 Calidad de software

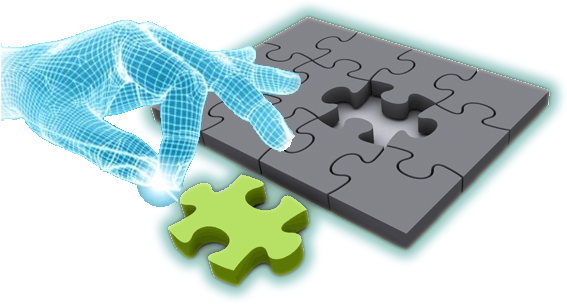


Figura 1 Calidad del software

Es el grado con el que un sistema, componente o proceso cumple los requerimientos especificados y las necesidades o expectativas del cliente.

Cuando hablamos de Calidad del Software, no sólo nos estamos refiriendo al software en sí; nos referimos a la calidad de todos los elementos que se construyen a lo largo del ciclo de vida de un proyecto: ingeniería de requisitos, diseño, arquitectura, documentos de despliegue, desarrollo, plan de pruebas, todo lo relativo al producto.

Tiene calidad si cumple:

1. Funcionalidad (que sirva un propósito),

2. Ejecución (que sea práctico),

3. Confiabilidad (que haga lo que debe),

4. Disponibilidad (que funcione bajo cualquier circunstancia) y

5. Apoyo, a un costo menor o igual al que el usuario está dispuesto a pagar.

### 1.2.1 Evolución Histórica



Figura 2 Calidad a través de los años

Esta historia se fue dando gracias a la evolución y desarrollo de las técnicas para mejorar rigurosamente el software  en las cuales se deben tener muy en cuenta las técnicas de Administración Total de Calidad (TQM) y del Sistema de Administración de Calidad (QMS).

**Niveles de calidad de software**

**A nivel de producto:**

Cuando nos centramos en el proceso de desarrollo de software y hacemos una serie de pruebas en paralelo con cada etapa, para detectar y corregir los posibles defectos que puedan surgir.

**A nivel de proyecto:**

Cuando nos centramos en controlar todas las fases y áreas de gestión de proyecto, implantando metodologías y mejores prácticas que aseguren la correcta gestión de las mismas.

**A nivel de proceso:**

Cuando nos centramos a gestionar todas las áreas de proceso de una organización, mediante la implantación de una metodología.

**Procesos y buenas prácticas:**

La calidad aumenta al aplicar una serie de procesos o metodologías y buenas prácticas. Controlan el proceso para poderlo analizar y mejorar.

**Herramientas:**

Proporcionan apoyo a la gestión de la calidad.

**Personas:**

Son elementos clave en la calidad como creadores y ejecutores.

**Medidas y métricas:**

Son los datos los que permiten evaluar el estado actual y ejecutar acciones para mejorar.

### 1.2.2 Lecciones Aprendidas en Desarrollo de Software

Si bien la Ingeniería de software es una disciplina relativamente joven, durante su corta vida ha atravesado diferentes etapas de las cuales aprendió la lección; la calidad es un valor en sí mismo y no un gasto que las empresas deben realizar para que su negocio prospere.

Es muy común encontrarse con proyectos que generan productos de dudosa calidad debido a que los responsables decidieron no realizar tareas de revisión de diseño y código o determinadas pruebas porque dilatan el tiempo de salda al mercado, cuestan dinero y además no habrá diferencia entre uno y otro producto cuando el proyecto termine.

Las lecciones no aprendidas se deben a la falta de madurez de conceptos aprendidos de forma correcta y en escenarios continuamente cambiantes lo que significa que es mejor proveer para las necesidades de largo plazo de las empresas y no focalizarse en las ganancias inmediatas; el objetivo es perdurar en el negocio generando trabajo.

Las consecuencias de percibir la calidad como un gasto del cual se puede prescindir genera el estado de situación en relación con el desarrollador de software.

La causa de falta de sistematización y automatización que contribuye a la baja calidad es el no acuerdo entre los participantes de los grupos de desarrollo acerca de las mejores prácticas a aplicar en los proyectos.

## 1.3 Calidad versus velocidad de desarrollo

Una forma de trabajo empleada a lo largo del tiempo para hacer predecibles los proyectos fue el establecimiento de estándares. También de promovió la reutilización de componentes. El uso repetitivo de estos, una vez probados, aumentaron la confiabilidad de los productos en los cuales se utilizaban.

De este modo se generaron activos y procedimientos para su construcción en disciplinas como el trabajo con requerimientos, diseño y códigos. Estos estándares fueron propuestos como una forma de garantizar la calidad de procesos y productos.

Con el tiempo, este aumento en la predecibilidad de los proyectos los hizo más estables. Pero esta estabilidad no era la dinámica apropiada cuando surgió la internet como plataforma de comunicación, oferta y consumo de servicios. Los tiempos de salida al mercado de los productos se acortaron y la estabilidad mencionada era percibida como un obstáculo; fue como aparecieron las metodologías agiles.

Fue necesario ponderar la creatividad y velocidad de adaptación a los nuevos escenarios por sobre el control y la previsibilidad en los proyectos. Este fenómeno de estabilidad de procesos versus creatividad ya se había percibido décadas antes entre las software Factory de Japón y la industria del software norteamericana. La calidad de excelencia de la producción oriental de logro a expensas de la ausencia de nuevos productos los cuales eran ofrecidos por las empras de EE.UU. que trabajaban en un ambiente mucho menos controlado.

Como Muchas veces a lo largo de la historia de esta industria, se instaló hace algunos años una idea de moda. Esta vez consiste en pensar que cuanto menos estructuradas la empresa sean, mejor estarán posicionadas para enfrentar cambios y ganar nuevos negocios. Sin embargo, como Cosummano muestra en su libro, también están en condiciones mucho más vulnerables para perder todo lo realizado en solo unos meses.

La razón de esto sigue siendo explica por W.Demming y su frase acerca de cuál es el objetivo de las empresas, perdurar en el tiempo y generar empleo.

El desafío de hoy día con respecto a garantizar la calidad en las organizaciones dedicadas al desarrollo de software es organizarse para que sus proyectos sean ordenados y predecibles; aunque deben a su vez tener la capacidad de dejar de lado estos procesos rápidamente para reorganizarse y adaptarse a los cambios. Por eso es muy importante lograr madurez en estos procesos de manera de seguirlos cómodamente para que no se constituyan en una carga que aumente los costos e impida

contar con agilidad para cambiar. Nos dedicaremos a tratar los temas relacionados con los modelos de referencias como CMMi en un capítulo de la última parte del libro

### 1.3.1 Estándares

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.

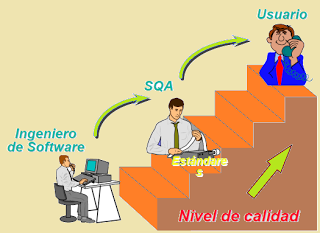
[](http://2.bp.blogspot.com/-aPoJHS28g4I/UjOpQxqkSLI/AAAAAAAAAAc/ZxSeEtmO3e4/s1600/Sin+t%C3%ADtulo1.png)

Figura 3 Nivel de calidad

 Los 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

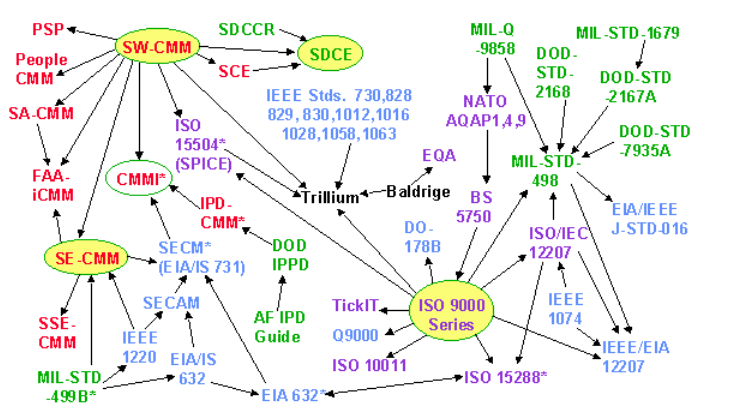


Figura 4 Estándares

Ejemplos

* ISO 9000
* CMMi
* ISO IRT 15504(SPICE)

**Capability Maturity Model Integration (CMMI)**

Para las compañías un producto o servicio es de calidad cuando satisface las necesidades y expectativas del cliente otorgando a éste seguridad sobre su uso, fiabilidad de sus funciones esperadas y confianza en un producto o servicio sin fallos y duradero según tiempos establecidos y acordados. Debido a la amplitud de temas que engloba el concepto de calidad se ha definido el concepto de Calidad Total, el cual se define como un sistema de gestión organizacional enfocado en la mejora continua del producto o servicio en todo su ciclo de vida, involucrando marketing, compras, diseño, fabricación y entrega.

La Calidad Total contempla dos fases:

1. Control de calidad, basado en técnicas de inspección aplicadas a producción.

2. Aseguramiento de la calidad, que persigue garantizar un nivel continúo de la calidad del Producto o Servicio proporcionado.

Los principios básicos de la Calidad Total son nueve:

1. Consecución de la plena satisfacción de las necesidades y expectativas del cliente (interno y externo).

2. Desarrollo de un proceso de mejora continua en todos los procesos.

3. Total compromiso de la Dirección.

4. Liderazgo activo de todo el equipo directivo.

5. Participación de todos los miembros de la organización

6. Fomento del trabajo en equipo hacia una gestión de Calidad Total.

7. El proveedor debe estar involucrado en el sistema de Calidad Total de la empresa.

8. Identificación y Gestión de los Procesos Claves de la organización.

9. Toma de decisiones de gestión basada en datos y hechos objetivos.

En este contexto, los modelos de calidad son sistemas basados en estudios experimentales de mejores prácticas que ayudan a una organización a implantar un Sistema de aseguramiento de la calidad. Los modelos de calidad se dividen en modelos de referencia, que indican cuáles son las prácticas pero no cómo se consiguen, y los modelos de implantación que se enfocan en cómo se. Consiguen aquella práctica.

Capability Maturity Model Integration, de ahora en adelante CMMI, es un modelo de referencia que se diferencia de otros modelos por el hecho de estar basado en prácticas ajustables a cualquier dominio de producción y poseer un enfoque global e integrado de la organización, con el propósito de alcanzar los objetivos del negocio. De esa forma CMMI permite a empresas complejas compuestas por varias áreas de negocio instaurar de una forma más sencilla un sistema de aseguramiento de la calidad.

### 1.3.2 Creatividad

Con los procedimientos y activos estándares agrupados en nuevas metodologías y modelos de referencia se comenzó a transitar por la etapa ya descripta que se dio a partir de la instalación de Internet como plataforma de comunicación y oferta y consumo de servicios. Por ese motivo los tiempos de salida al mercado y de estabilidad de los requerimientos se acortaron de una manera impensada. Por esta razón se comenzó a cuestionar los estándares, ya que estos proponían una forma de trabajo cuya estabilidad contrastaba con la dinámica de la nueva etapa. Así fue que nacieron la metodologías ágiles (Mary and Tom Poppendiek), a partir de que la comunidad comenzó a ponderar la creatividad y velocidad de adaptación a los nuevos escenarios por sobre el control y la predicibilidad en los proyectos. Este fenómeno de estabilidad de procesos versus creatividad ya se había percibido décadas atrás entre las software factory de Japón y la industria del software norteamericana, cuando se analizó la calidad de excelencia de los productos orientales a expensas de la ausencia de nuevos productos, los cuales eran ofrecidos por las empresas de EE.UU. que trabajaban en un ambiente mucho menos controlado (Michael Cusumano).

### 1.3.3 Madurez

Como muchas veces a lo largo de la historia de esta industria, hace algunos años se instaló una idea de moda. Esta vez consiste en pensar que cuanto menos estructuradas sean las empresas, mejor estarán posicionadas para enfrentar cambios y ganar nuevos negocios. Sin embargo, como Cosummano muestra en su libro, también están en condiciones mucho más vulnerables para perder todo lo realizado en solo unos meses. La razón de esto la explica W. Demming en su frase acerca de cuál es el objetivo de las empresas. Actualmente, el desafío con respecto a garantizar la calidad en las organizaciones dedicadas al desarrollo de software es organizarse para que sus proyectos sean ordenados y predecibles; aunque deben a su vez tener la capacidad de dejar de lado estos procesos rápidamente para reorganizarse adaptándose a los cambios. Para esto es muy importante lograr madurez en estos procesos de manera de seguirlos cómodamente para que no se constituyan en una carga que aumente los costos e impida contar con agilidad para cambiar. Nos dedicaremos a tratar este tema en relación con los modelos de referencia como CMMi en el capítulo siete.

## 1.4.1 Surgimiento y Evolución

Con el objetivo de establecer formas estándares en las prácticas y activos de desarrollo aparecieron a lo largo de los años distintas normas y modelos de referencia. Los llamamos así porque constituyen la referencia al momento de implementar una mejora de procesos en una organización con el objetivo de aumentar la calidad de procesos y productos generados. Un fin para el cual se han utilizado estos modelos ha sido la calificación, clasificaciones y comparación de empresas por parte de los compradores de productos de software. El ejemplo más notorio, ya citado, es el del DoD y el modelo CMM, y después CMMi. Básicamente, con ellos se buscó establecer estándares de organizaciones. La utilización inteligente de estos modelos contribuyo a mejorar los productos de software, mientras que el mal uso generó frustraciones y despilfarro de recursos. En la década de 1990 el modelo CMM (Capability Maturity Model) se convirtió en el estándar de hecho a nivel global. Más tarde, en la década de 2000, fue remplazado por su versión mejorada CMMi (Capability Maturity Model Integration).

En la industria del software se pueden evidenciar necesidades de satisfacción del cliente de productos o servicios de software, de reducción de recursos invertidos en proyectos de software y de la efectiva asignación de recursos humanos. Si hablamos de la calidad del software, una de las primeras definiciones aseguraba que “la calidad de un programa o sistema se evaluaba de acuerdo al número de defectos por cada mil líneas de código. (KLOC: Kilo Lines Of Code)”.

## 1.4.2 Modelos

Existen una variedad de modelos para la gestión de la calidad del software.

Algunos de los modelos más utilizados son:

* **CMMI (Modelo de Madurez de Capacidades Integrado)**
* TSP (Team Software Process)
* ISO/IEC 15504 (Evaluación del Proceso)
* ISO/IEC 12207 (Procesos del Ciclo de Vida del Software)
* PSP (Proceso Personal de Software)
* ITIL (Librería de Infraestructura de Tecnologías de Información)
* MOPROSOFT (Modelo de Procesos Para la Industria del Software)

(Guillermo, 2011)

## 1.4.3 Certificaciones y Evaluaciones

**CMMI (Modelo de Madurez de Capacidades Integrado)**

Se construye a través de prácticas y objetivos que sirven para evaluar el desempeño y madurez de la organización y decidir qué es lo que se desea mejorar.

Las organizaciones que aplican CMMI pueden predecir sus costos, cronogramas y calidad, obteniendo resultados de negocio que los diferencian de sus competidores.

**Modelos**

CMMI cuenta con los siguientes modelos:

* CMMI para el Desarrollo (CMMI-DEV)

Está dirigido a procesos de desarrollo de productos (software, hardware, sistemas). La última versión es la v1.3 liberada en Noviembre del 2010.

* CMMI para Adquisiciones (CMMI-ACQ)

Está dirigido a gestión de cadenas de suministro, y a procesos de adquisición y subcontrataciones. La última versión es la v1.3 liberada en Noviembre del 2010.

* CMMI para Servicios (CMMI-SVC)

Está dirigido para servir de guía en la prestación de servicios tanto dentro de una organización como para clientes externos. La última versión es la v1.3 liberada en Noviembre.

**Niveles de Madurez CMMI:**

* Inicial o Nivel 1 CMMI. Este es el nivel en donde están todas las empresas que no tienen procesos.
* Nivel 2 CMMI. Quiere decir que el éxito de los resultados obtenidos se puede repetir.
* Nivel 3 CMMI. Alcanzar este nivel significa que la forma de desarrollar proyectos (gestión e ingeniería) está definida.
* Nivel 4 CMMI. Los proyectos usan objetivos medibles para alcanzar las necesidades de los clientes y la organización.
* Nivel 5 CMMI. Los procesos de los proyectos y de la organización están orientados a la mejora de las actividades

**Beneficios de CMMI**

CMMI constituye una guía para mejorar la efectividad y eficiencia en las distintas disciplinas de proceso de la organización y provee una visión de mejora integrada y transversal dentro de la organización. La finalidad es mejorar el desempeño por medio de la Reducción de costos, las Entregas a Tiempo y Mejoras en la Productividad, junto con un Aumento en la Calidad y la Satisfacción del Cliente.

(America Veintiuno, 2017)

# CAPÍTULO 2 CAUSAS QUE DETERIORAN LA CALIDADEN EL SOFTWARE

## 2.1 Introducción

### 2.1.1 Definiciones

* **Corrección.**

El grado en que el programa cumple con su especificación y satisfacer los objetivos que propuso el cliente.

* **Confiabilidad.**

El grado en que se esperaría que un programa desempeña su función con la precisión requerida.

* **Eficiencia.**

La cantidad de código y de recursos de cómputo necesarios para que un programa realice su función.

* **Integridad.**

El grado de control sobre el acceso al software o los datos por parte de las personas no autorizadas.

* **Facilidad de uso.**

El esfuerzo necesario para aprender, operar y preparar los datos de entrada de un programa interpreta la salida.

* **Facilidad de mantenimiento.**

El esfuerzo necesario para localizar y corregir un error en un programa.

* **Flexibilidad.**

El esfuerzo que demanda probar un programa con el fin de asegurar que realiza su función.

* **Portabilidad.**

El esfuerzo necesario para transferir el programa de un entorno de hardware o software a otro.

* **Facilidad de reutilización.**

El grado en que un programa o partes de él pueden reutilizarse en otras aplicaciones (en relación con el empaquetamiento y el alcance de las funciones que realiza el programa).

* **Interoperabilidad.**

El esfuerzo necesario para acoplar un sistema con otro.

Es difícil y en algunos casos imposibles, desarrollar medidas directas[1](http://www.eumed.net/tesis-doctorales/2014/jlcv/calidad-software.htm" \l "_ftn1" \o ") de estos factores de la calidad. En realidad, muchas de las métricas que definen McCall et al. Sólo se miden de forma subjetiva. Ya que es común que las métricas adquieran la forma de una lista de comprobación que se emplea para “asignar una graduación” a atributos específicos del software. Vega et al. (2008), proponen un modelo con métricas distintas al propuesto por McCall y que ha sido utilizado y comprobado en distintos proyectos de desarrollo de software. Los factores que conforman al modelo y su descripción, se presentan a continuación.

* **Corrección.**

El grado en que un producto de software satisface sus especificaciones y consigue los objetivos de la misión encomendada por el usuario.

* **Confiabilidad.**

El grado en que se puede esperar que un producto de software lleve a cabo sus funciones esperadas con la precisión requerida.

* **Eficiencia**.

La cantidad de recursos computacionales y de código requeridos por un producto de software para llevar a cabo las funciones encomendadas.

* **Integridad**.

El grado en que puede controlarse (facilitar y restringir) el uso y acceso al software y a los datos, tanto al personal autorizado como al no autorizado.

* **Facilidad de uso**.

El esfuerzo requerido para aprender, trabajar, preparar la entrada e interpretar la salida de un producto de software.

**Facilidad de mantenimiento**.

El esfuerzo necesario para localizar y corregir los errores en un producto de software.

* **Flexibilidad**.

El esfuerzo requerido para modificar un producto de software una vez que se encuentra ya liberado o en producción, esto es, una vez que el usuario esté haciendo uso de él.

* **Facilidad de prueba**.

El esfuerzo requerido para probar un producto de software, de tal forma que se asegure que realiza las funciones especificadas por el usuario.

* **Portabilidad**.

El esfuerzo requerido para transferir un producto de software de una plataforma (entorno de hardware y software) a otra.

* **Reusabilidad**.

El grado en que un producto de software (o alguna de sus partes) pueda volver a ser utilizado en otras aplicaciones, aun cuando la funcionalidad de la misma cambie.

* **Facilidad de interoperación**.

El esfuerzo requerido para lograr que un producto de software trabaje con otro, compartiendo recursos.

#### 2.1.1.1 Calidad de producto

La calidad del producto estudia la adecuación del producto a las especificaciones.

La calidad del proceso estudia la capacidad del sistema de producción de obtener productos que cumplan las especificaciones.

**Calidad del Producto y calidad de proceso:**

Por calidad del producto se entiende por la propiedad o conjunto de propiedades inherentes a algo, que permite juzgar su valor.

Grado en el que un conjunto de características inherentes cumplen con los requisitos. ISO 9000.

Calidad es cumplimiento de requisitos.

Calidad es satisfacción del cliente

**Cuando no hay calidad:**

Cuando los programas no hacen exactamente lo que se espera.

**Proyectos que no se terminaron nunca.**

Sistemas informáticos que no se utilizan por la dificultad de su manejo.

**Software poco seguro.**

**Aspectos de la Calidad del Software:**

**Interna:** medible a partir de las características intrínsecas del propio producto software (como el código fuente).

**Externa:** medible a partir del comportamiento del producto software (como durante una prueba).

**En uso:** medible durante la utilización efectiva por parte del usuario (en un entorno de pre o producción).

**Calidad del producto software**

Es el grado en el que producto software incorpora un conjunto de características, de manera que garantiza el cumplimiento de los requisitos y necesidades del cliente.

El objetivo no es necesariamente alcanzar una calidad perfecta, sino la necesaria y suficiente para cada contexto de uso a la hora de la entrega y del uso por parte de los usuarios.

Es primordial comprender la existencia de distintas visiones de la calidad (productor, usuario, valor, etc.).

Dada la complejidad de la calidad, es necesario utilizar un modelo que especifique las características de la calidad.

**Principales Normas y Estándares de la calidad del producto**

**Funcionalidad**

**Adecuación:** Capacidad del producto software para proporcionar un conjunto apropiado de funciones para tareas y objetivos de usuarios específicos.

**Exactitud:** Capacidad del producto software para proporcionar los resultados o efectos correctos o acordados, con el grado necesario de precisión.

**Interoperabilidad:** Capacidad del producto software para interactuar con uno o más sistemas especificados.

**Seguridad de acceso:** Capacidad del producto software para proteger información y datos de manera que las personas o sistemas no autorizados no puedan leerlos o modificarlos, al tiempo que no se deniega el acceso a las personas o sistemas autorizados.

**Cumplimento Funcional:** Capacidad del producto software para adherirse a normas, convenciones o regulaciones en leyes y prescripciones similares relacionadas con funcionalidad.

**Fiabilidad**

**Madurez:** Capacidad del producto software para evitar fallar como resultado de fallos en el software.

**Tolerancia a fallos:** Capacidad del software para mantener un nivel especificado de prestaciones en caso de fallos software o de infringir sus interfaces especificados.

**Capacidad de recuperación**: Capacidad del producto software para reestablecer un nivel de prestaciones especificado y de recuperar los datos directamente afectados en caso de fallo.

**Cumplimiento de la fiabilidad**: Capacidad del producto software para adherirse a normas, convenciones o regulaciones relacionadas con al fiabilidad.

**Usabilidad**

**Capacidad para ser entendido**: Capacidad del producto software que permite al usuario entender si el software es adecuado y cómo puede ser usado para unas tareas o condiciones de uso particulares.

**Capacidad para ser aprendido**: Capacidad del producto software que permite al usuario aprender sobre su aplicación.

**Capacidad para ser operado:** Capacidad del producto software que permite al usuario operarlo y controlarlo.

**Capacidad de atracción**: Capacidad del producto software para ser atractivo al usuario.

**Cumplimiento de la usabilidad:** Capacidad del producto software para adherirse a normas, convenciones, guías de estilo o regulaciones relacionadas con la usabilidad.

**Eficiencia**

**Comportamiento temporal:** Capacidad del producto software para proporcionar tiempos de respuesta, tiempos de proceso y potencia apropiados, bajo condiciones determinadas.

**Utilización de recursos:** Capacidad del producto software para usar las cantidades y tipos de recursos adecuados cuando el software lleva a cabo su función bajo condiciones determinadas.

**Cumplimiento de la eficiencia:** Capacidad del producto software para adherirse a normas o convenciones relacionadas con la eficiencia.

**Mantenibilidad**

**Capacidad para ser analizado:** Es la capacidad del producto software para serle diagnosticadas deficiencias o causas de los fallos en el software, o para identificar las partes que han de ser modificadas.

**Capacidad para ser cambiado**: Capacidad del producto software que permite que una determinada modificación sea implementada.

**Estabilidad:** Capacidad del producto software para evitar efectos inesperados debidos a modificaciones del software.

**Capacidad para ser probado**: Capacidad del producto software que permite que el software modificado sea validado.

**Cumplimiento de la mantenibilidad**: Capacidad del producto software para adherirse a normas o convenciones relacionadas con la mantenibilidad.

**Portabilidad**

**Adaptabilidad:** Capacidad del producto software para ser adaptado a diferentes entornos especificados, sin aplicar acciones o mecanismos distintos de aquellos proporcionados para este propósito por el propio software considerado.

**Instabilidad:** Capacidad del producto software para ser instalado en un entorno especificado.

**Coexistencia:** Capacidad del producto software para coexistir con otro software independiente, en un entorno común, compartiendo recursos comunes.

**Capacidad para reemplazar**: Capacidad del producto software para ser usado en lugar de otro producto software, para el mismo propósito, en el mismo entorno.

**Cumplimiento de la portabilidad**: Capacidad del producto software para adherirse a normas o convenciones relacionadas con la portabilidad.

Actualmente se encuentra en estado 90.60 formado por 6 partes donde establece los procesos para realizar la evaluación de la calidad del producto software directamente relacionado con la ISO 9126.

Parte 1: Visión general

Parte 2: Gestión y planificación

Parte 3: Proceso para los desarrolladores

Parte 4: proceso para los adquisidores

Parte 5: Proceso para los evaluadores

Parte 6: Documentación de los módulos de evaluación

## 2.2 Causa que deterioran la calidad

***Frase de Joseph Juran:***

*“Calidad es la idoneidad de uso. Es decir, las características del producto que satisfacen las necesidades del cliente y por tanto producen satisfacción de producto. La Calidad es la inexistencia de deficiencias”.*

Ahora sí, comencemos con la definición de la palabra calidad. Existen varias: algunas más formales que otras, algunas más enfocadas al producto, otras al cliente y, otras, al servicio. Sin embargo, todas incluyen el mismo propósito de garantía.

En la actualidad, existen diversos factores que pueden afectar los procesos de calidad. Estos provocan desde procesos tardíos hasta el fracaso de los mismos.

Con el fin de aumentar la calidad en un producto de software o proceso, es recomendable establecer una metodología que cubra nuestras necesidades. Esto nos llevará a definir buenas prácticas y nos permitirá llevar un control de lo que se está realizando en cada una de las etapas

del producto. También, se podrá realizar una evaluación, la cual nos ayudará a saber en qué punto se falló para poder mejorarlo.

Algunas de las actividades que se llevan a cabo en la gestión de la calidad son:

**ASEGURAMIENTO, PLANIFICACIÓN Y CONTROL DE LA CALIDAD.**

Entre los factores más importantes que intervienen en la calidad encontramos: procesos y prácticas, herramientas, personal y métricas.

A continuación, se mencionan algunos de factores que intervienen en el mal uso de las prácticas para el aseguramiento de calidad.

**•NO CONTAR CON GESTIÓN DE PRUEBAS**

Al no contar con un plan de pruebas, no tener casos de pruebas o datos y sólo mirar la funcionalidad del sistema, se obtendrán como resultado altos índices de errores y, como consecuencia de la mala gestión, el producto se liberará demasiado tarde y/o incompleto, ocasionando un posible disgusto del cliente. Por ende, esto traerá aparejados costos demasiados elevados.

**•LA COMPLEJIDAD Y EL TIEMPO PLANIFICANDO ESTÁN SUBESTIMADOS**

La complejidad de un proyecto, en ocasiones, es la principal causa en la demora del inicio de un desarrollo. Otros factores son la falta de un plan de pruebas, o bien, no tener la estimación adecuada, lo cual impacta en el tiempo para el diseño de pruebas y la ejecución de las mismas.

Es necesario que la gestión de pruebas comience con el proyecto. El diseño de caso de pruebas y las pruebas pueden comenzar con los requisitos.

**•DEFICIENCIA EN LA GESTIÓN DE DATOS DE PRUEBA**

Al definir y desarrollar los casos de pruebas, es importante tener en mente los parámetros a utilizar para su ejecución. En muchas ocasiones, sólo se definen los casos olvidando los datos de pruebas.

Es fundamental definir y gestionar los datos de prueba que serán necesarios junto con los casos de prueba. Para ello, cabe recordar que se debe contar con datos de prueba primarios y secundarios, lo cual ayudará bastante para las pruebas de regresión.

**•AUTOMATIZACIÓN DE PRUEBAS**

Muchas veces sólo se piensa en las pruebas GUI (Interfaz Gráfica de Usuario), sin pensar qué hay del código, interfaces, rendimiento, etc. Una buena práctica, es comenzar temprano con el control de código, usando una métrica de cobertura. Además, es importante comenzar con

las pruebas de la GUI, sólo si la GUI está estable (recuerde que es necesario contar datos de prueba correctos para su funcionamiento).

**•CONOCIMIENTO DEL NEGOCIO**

**¡Pieza importantísima!** En muchas ocasiones se utilizan recursos para diseñar tareas del proyecto, sin tener el suficiente dominio o conocimiento del negocio. Es indispensable que se cuente con personal capaz de realizar cada una de las tareas referentes a su área, de lo contrario, es importante capacitarlos antes de comenzar con las actividades del proyecto. Esto ayudará a agilizar y terminar en tiempo y forma cada una de las etapas del proyecto.

**•TÉCNICAS DE PRUEBAS**

Los gestores y los recursos no están familiarizados con las técnicas de pruebas. Las técnicas sistemáticas de diseño de casos de prueba reducen bastante el número de casos de prueba, ahorrando tiempo y costos.

**•NO SE USAN HERRAMIENTAS DE PRUEBAS, O SE USAN LAS INADECUADAS**

Las herramientas de pruebas ayudan a mejorar los procesos para la calidad de software y existen diferentes opciones que se pueden utilizar en el proceso de desarrollo. Cabe mencionar que, aparte de los beneficios que nos ofrecen, también existen algunos riegos a los que se está expuesto al no tener el conocimiento con exactitud de cada herramienta:

## 2.3 Aspectos sobre los que trabaja para mejorar la calidad.

**Crear constancia en la mejora de los productos:** ser un agente competitivo en el mercado y mantenerse activo.

**Adoptar una filosofía corporativa** que involucre a cada uno de los empleados de la organización. Mayor implicación es sinónimo de mayor eficacia.

**Incluir la calidad del producto desde el inicio** y no sólo en sus fases posteriores.

**Minimizar los costes totales a largo plazo.** En lo posible, tener un solo proveedor para cada caso y fomentar con ellos relaciones de confianza y lealtad.

**Todas las actividades de la producción pueden ser mejoradas.**

**Adoptar las actividades formativas en gestión de calidad y excelencia** como una parte fundamental del área de Recursos Humanos.

**Identificar líderes con capacidades** y talentos adecuados en cada caso. La idea es un mayor grado de orientación y supervisión en los procesos.

**Eliminar el miedo en la estructura corporativa**.

**Difuminar, mientras sea posible**, las barreras jerárquicas de la empresa y construir en su lugar un sistema de cooperación y beneficio mutuo.

**Remover los sistemas de actuación** y otros elementos que generen competencia y rivalidad entre los miembros de los equipos de trabajo.

**Incorporar un sistema de evaluación y autocrítica en la empresa.**

**Impulsar el trabajo conjunto y la retroalimentación**, el diálogo y las soluciones consensuadas. El nivel de compromiso de los empleados aumentará.

**La actitud preventiva**

En la Calidad del Software, al igual que en otras disciplinas, prevenir resulta ser la opción más efectiva y de menor coste. Es un hecho que el coste de la corrección de errores aumenta, en ciertos casos casi exponencialmente, a medida que se avanza en el ciclo de vida de desarrollo.

Por ejemplo: detectar y corregir un error en requisitos gracias a la actividad de revisión correspondiente es poco costoso: consistiría en modificar un documento y las posibles dependencias con otros. Corregir el mismo error cuando el sistema ya está en producción, supone realizar la petición de cambio correspondiente, modificar SW, documentar, probar, progresar SW entre entornos, etc.

**Los procesos**

Es básico definir los procesos que lleven a la industrialización de la construcción SW dentro de la Organización. Esto permite la realización de los distintos proyectos bajo pautas similares y reducir la dependencia en personas concretas. Es decir, se elimina el efecto héroe como única garantía para el éxito de los proyectos.

La definición, implantación, medición y mejora continua de los procesos no es una labor sencilla y requiere su tiempo. Hay que desconfiar de propuestas de cambios en procesos en tiempos cortos. Afortunadamente existen modelos (ej: CMMI y SPICE) que permiten no partir de cero para dotar de procesos de ingeniería, soporte y gestión a las Organizaciones.

**Los productos**

Los modelos basados en procesos parten de la hipótesis de que un proceso bien definido lleva a un producto de calidad, que al fin y al cabo es el objetivo primordial.

Formalizar el proceso de revisión de los distintos entregables (productos), tanto documentación como código, permite garantizar que se ha generado lo que se requiere y que además no existen ambigüedades e inconsistencias, que introducen el riesgo de errores por asunciones no correctas.

**La tecnología**

Afortunadamente, y gracias en parte al tirón del Opensource, existen soluciones para automatizar gran parte de las actividades y tareas que integran los distintos procesos relacionados con el desarrollo, con una relación de costes razonable.

Estas soluciones sin lugar a dudas mejoran la eficiencia y efectividad de las Organización, pero hay que considerar ciertos aspectos para no fracasar en su implantación:

La selección de herramientas hay que considerarlo como un proyecto en sí y darle la importancia que tiene: establecer los criterios de selección, evaluar, hacer pilotos con las soluciones que mejor se adaptan a mis necesidades, etc.

Nunca adaptar los procesos a la herramienta seleccionada, sino al revés. Primero se definen los procesos y luego se seleccionan las herramientas que lo soportan.

Cuidar la integración entre distintas soluciones: por ejemplo, una herramienta de gestión de configuración es importante que esté integrada con una de gestión de requisitos.

**La capacitación**

Capacitar a los recursos y dotarlos de los conocimientos necesarios, apoyándose en Certificaciones ya contrastadas, permiten además de mejoras en el desempeño de las distintas actividades, aumentar la motivación y fidelización de los RRHH especializados.

**Independencia**

El rol del asegurador de la Calidad, como velador de que se están aplicando los procesos establecidos y los productos generados se ajustan a los requisitos, además de la especialización que necesita, requiere de una independencia.

Esto se puede conseguir tanto creando grupos internos con independencia organizativa de otras áreas involucradas en la

construcción de aplicaciones, especialmente de las áreas de desarrollo; como mediante subcontratación a empresas especializadas e independientes.

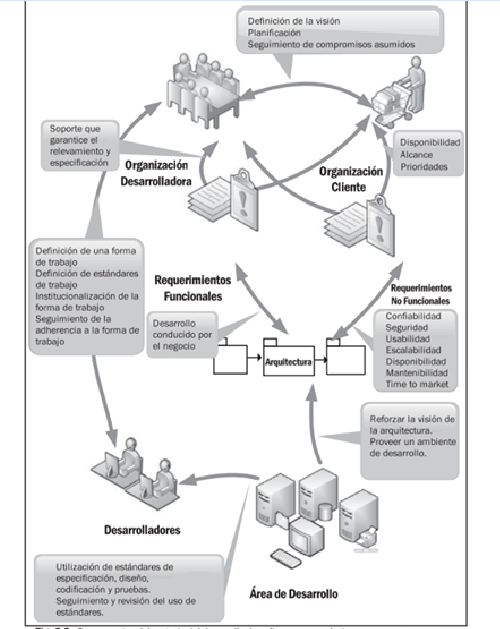
**Medir**

Sin medir no se puede conocer el estado cuantitativo y por consiguiente tampoco se puede tener referencias para la mejora.

Para medir hay que definir dos aspectos:

Modelo de métricas, que estructura la información desde la capa de indicadores que aportan información (de los procesos, productos, financieros, etc.) hasta las métricas básicas que los componen.

En definitiva, la aplicación de procesos de Verificación y Validación del software conlleva un aumento en la Calidad de los Productos, lo que implica una mayor productividad y un rápido retorno de la inversión.



## 2.4 Formato de tratamiento de los temas

Para mejorar las prácticas de una organización desarrolladora de Software se debe trabajar sobre los errores comunes, las tareas básicas de gestión y las tareas básicas técnicas. Afirma que la forma de lograr mejoras es comenzando por las bases; es decir, mejorar las actividades básicas diarias, las realizadas por los diferentes roles en su participación en los proyectos. En forma coincidente proponemos capacitar a los miembros de los grupos de desarrollo en la realización de sus actividades. Asumimos que esta capacitación se lleva a cabo y los involucrados en los proyectos ya están aplicando lo aprendido. Cada propuesta que hagamos estará orientada a la mejora en el sentido de evitar las causas o minimizar los efectos del deterioro mencionado.