ISO/IEC 25010

- El modelo de calidad representa la piedra angular en torno a la cual se establece el sistema para la ***evaluación de la calidad del producto.***
- *En este modelo se determinan las características de calidad que se van a tener en cuenta a la hora de evaluar las propiedades de un producto software determinado.*
- La calidad del producto de software se puede interpretar como el grado en que dicho producto satisface los requisitos de sus usuarios aportando de esta manera un valor.
- El modelo de calidad del producto definido por la ISO/IEC 25010 se encuentra compuesto por las ocho características de calidad que se muestran en la siguiente figura:

Adecuación Funcional: ***Representa la capacidad del producto software para proporcionar funciones que satisfacen las necesidades declaradas e implícitas****,* cuando el producto se usa en las condiciones especificadas. Esta característica se subdivide a su vez en las siguientes subcaracterísticas:

- **Completitud funcional.** Grado en el cual el conjunto de funcionalidades cubre todas las tareas y los objetivos del usuario especificados.
- **Corrección funcional.** Capacidad del producto o sistema para proveer resultados correctos con el nivel de precisión requerido.
- **Pertinencia funcional.** Capacidad del producto de software para proporcionar un conjunto apropiado de funciones para tareas y objetivos de usuario especificados.

Eficiencia de desempeño: *Representa el desempeño relativo a la cantidad de recursos utilizados bajo determinadas condiciones.* Esta se subdivide a su vez en subcaracterísticas:

- **Comportamiento temporal.** Los tiempos de respuesta y procesamiento y los ratios de *throughput* de un sistema cuando lleva a cabo sus funciones bajo condiciones determinadas en relación con un banco de pruebas (*benchmark*) establecido.
- **Utilización de recursos.** Las cantidades y tipos de recursos utilizados cuando el software lleva a cabo su función bajo condiciones determinadas.
- **Capacidad.** Grado en que los límites máximos de un parámetro de un producto o sistema software cumplen con los requisitos.

Compatibilidad: Capacidad de dos o más sistemas o componentes para intercambiar información y/o llevar a cabo sus funciones requeridas cuando comparten el mismo entorno hardware o software. Esta característica se subdivide a su vez en las siguientes subcaracterísticas:

- **Coexistencia.** Capacidad del producto para coexistir con otro software independiente, en un entorno común, compartiendo recursos comunes sin detrimento.
- **Interoperabilidad.** Capacidad de dos o más sistemas o componentes para intercambiar información y utilizar la información intercambiada.

Usabilidad: *Capacidad del producto software para ser entendido, aprendido, usado y resultar atractivo para el usuario, cuando se usa bajo determinadas condiciones.* Esta se subdivide a su vez en:

- **Capacidad para reconocer su adecuación.** Capacidad del producto que permite al usuario entender si el software es adecuado para sus necesidades.
- **Capacidad de aprendizaje.** Capacidad del producto que permite al usuario aprender su aplicación.
- **Capacidad para ser usado.** Capacidad del producto que permite al usuario operarlo y controlarlo con facilidad.
- **Protección contra errores de usuario.** Capacidad del sistema para proteger a los usuarios de hacer errores.
- **Estética de la interfaz de usuario.** Capacidad de la interfaz de usuario de agradar y satisfacer la interacción con el usuario.
- **Accesibilidad.** Capacidad del producto que permite que sea utilizado por usuarios con determinadas características y discapacidades.

Fiabilidad: Capacidad de un sistema o componente para desempeñar las funciones especificadas, cuando se usa bajo unas condiciones y periodo de tiempo determinados. Esta se subdivide en:

- **Madurez.** Capacidad del sistema para satisfacer las necesidades de fiabilidad en condiciones normales.
- **Disponibilidad.** Capacidad del sistema o componente de estar operativo y accesible para su uso cuando se requiere.
- **Tolerancia a fallos.** Capacidad del sistema o componente para operar según lo previsto en presencia de fallos hardware o software.
- **Capacidad de recuperación.** Capacidad del producto software para recuperar los datos directamente afectados y restablecer el estado deseado del sistema en caso de interrupción o fallo.

Seguridad: *Capacidad de protección de la información y los datos de manera que personas o sistemas no autorizados no puedan leerlos o modificarlos.* Esta se subdivide en:

- **Confidencialidad.** Capacidad de protección contra el acceso de datos e información no autorizados, ya sea accidental o deliberadamente.
- **Integridad.** Capacidad del sistema o componente para prevenir accesos o modificaciones no autorizados a datos o programas de ordenador.
- **No repudio.** Capacidad de demostrar las acciones o eventos que han tenido lugar, de manera que dichas acciones o eventos no puedan ser repudiados posteriormente.
- **Responsabilidad.** Capacidad de rastrear de forma inequívoca las acciones de una entidad.
- **Autenticidad.** Capacidad de demostrar la identidad de un sujeto o un recurso.

Mantenibilidad: *Representa la capacidad del producto software para ser modificado efectiva y eficientemente,* debido a necesidades evolutivas, correctivas o perfectivas. Esta se subdivide en:

- **Modularidad.** *Capacidad de un sistema o programa de ordenador* (compuesto de componentes discretos) que permite que un *cambio en un componente tenga un impacto mínimo en los demás*.
- **Reusabilidad.** *Capacidad de un activo que permite que sea utilizado en más de un sistema software* o en la construcción de otros activos.
- **Analizabilidad.** Facilidad con la que se puede evaluar el impacto de un determinado cambio sobre el resto del software, diagnosticar las deficiencias o causas de fallos en el software, o identificar las partes a modificar.
- **Capacidad para ser modificado.** *Capacidad del producto que permite que sea modificado de forma efectiva y eficiente* sin introducir defectos o degradar el desempeño.
- **Capacidad para ser probado.** *Facilidad con la que se pueden establecer criterios de prueba para un sistema o componente* y con la que se pueden llevar a cabo las pruebas para determinar si se cumplen dichos criterios.

Portabilidad: *Capacidad del producto o componente de ser transferido de forma efectiva y eficiente de un entorno* hardware, software, operacional o de utilización a otro. Esta característica se subdivide a su vez en las siguientes subcaracterísticas:

- **Adaptabilidad.** Capacidad del producto que le permite ser adaptado de forma efectiva y eficiente a diferentes entornos determinados de hardware, software, operacionales o de uso.
- **Capacidad para ser instalado.** Facilidad con la que el producto se puede instalar y/o desinstalar de forma exitosa en un determinado entorno.
- **Capacidad para ser reemplazado.** Capacidad del producto para ser utilizado en lugar de otro producto software determinado con el mismo propósito y en el mismo entorno.

Capítulo 1 - Introducción calidad de software

"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"

- **Proceso eficaz:** Establece la infraestructura que da apoyo a cualquier esfuerzo de elaboración de un producto de software de alta calidad.
- **Producto útil:** Entrega contenido, funciones y características que el usuario final desea, de igual importancia es que entrega estos activos de forma confiable y libre de errores.
- **Valor medible:** 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.

Principales objetivos estratégicos de las organizaciones.

Calidad según la RAE.

1. Propiedad o conjunto de propiedades inherentes a algo, que permiten juzgar su valor.

2. Condición o requisito que se pone en un contrato.

La calidad suele ser transparente cuando está presente, pero resulta fácilmente reconocible cuando está ausente (por ejemplo cuando el producto falla o el servicio es deficiente).

Vistas de la Calidad

- **Trascendental:** Se reconoce pero no se define. Es un ideal al que se intenta llegar.
- **Usuario:** Adecuación al propósito. Cuantificar y medir productos, establecer objetivos a ser alcanzados.
- **Fabricante:** Calidad durante producción y entrega del producto. Vista centrada en el proceso.
- **Producto:** Unida a las características del producto en sí mismo, "desde adentro". Usuario y fabricante "desde fuera"
- **Basada en el valor:** Cantidad que el cliente está dispuesto a pagar.

Orígenes de la calidad

Una *buena gestión de calidad busca que estas tres coincidan* lo más posible.

- **Realizada:** Persona que realiza el trabajo gracias a su habilidad en la ejecución de la tarea. Se potencia con la mejora de las habilidades personales y técnicas de los participantes en un proceso.
- **Planificada**: Es la que se ha pretendido obtener. Es la que aparece descrita en una especificación, en un documento de diseño o en un plano. Se potencia con la elaboración de una especificación que sirva de buena referencia a los participantes de un proceso.
- **Necesaria**: Es la que el cliente exige con mayor o menor grado de concreción, o al menos, le gustaría recibir. Se potencia con una adecuada obtención de información de la idea de calidad de los clientes.

Datos Históricos Importantes

Año	Hecho Importante
Épocas iniciales	Alan Turing, descifra los códigos secretos Enigma usados por los alemanes en la II Guerra Mundial para sus comunicaciones. Turing fue un pionero en el desarrollo de la lógica de los computadores modernos, y uno de los primeros en tratar el tema de la inteligencia artificial con máquinas.
Primera era	Durante los primeros años el software se hacía muy ajustado a las necesidades de cada persona.
Segunda era	Mitad de la época de los sesenta hasta finales de los setenta. Multi-programación y sistemas multi-usuario introdujeron nuevos conceptos de hci. Se empieza a hablar en milisegundos y ya no en minutos

Año	Hecho Importante
Tercera era	Mediados de los años 70 - más allá de una década. Sistemas distribuidos, múltiples computadoras, concurrencia, incrementó notablemente la complejidad de los sistemas informáticos. El final fue por la llegada de los microprocesadores. El microprocesador ha producido un extenso grupo de productos inteligentes, desde automóviles hasta hornos de microondas, desde robots industriales a equipos de diagnósticos de suero sanguíneo, pero ninguno ha sido más importante que la computadora personal.
Cuarta era	La cuarta era de la evolución de sistemas informáticos se aleja de las computadoras individuales y de los programas de computadoras, dirigiéndose al impacto colectivo de las computadoras y del software. Redes globales y locales, aplicaciones de software avanzadas. La industria del software ya es cuna de la economía mundial.

Crisis del Software 1968

Refiere a la dificultad de escribir programas libres de defectos, fácilmente comprensibles y verificables.

Problemas:

- La planificación y estimación del coste es frecuentemente muy imprecisa.
- La "productividad" de la gente del software no responde a la demanda de sus servicios.
- La calidad del software no es la adecuada.

Proyectos de software fallaron:

- No fueron entregados a tiempo
- Se gastó mucho más de lo previsto

- Software de baja calidad
- Caro de mantener (corrección de fallas, modificaciones difíciles por cambio en requisitos, adaptación a nuevos dispositivos)
- Software no cumplía las expectativas del cliente
- No se recogen datos sobre el proceso de desarrollo de software. (No existen datos históricos).
    - - Sin datos históricos como guía, la estimación no es buena y los resultados son pobres
    - - No se puede evaluar la eficacia de nuevas herramientas.

Para superar la crisis:

- Aparición de *metodologías* concretas de *desarrollo*
- Concepción de la *Ingeniería del Software* como disciplina
- *Trabajo en equipo y especialización* (analistas, programadores, ...)

No hay una situación estable, sino una evolución permanente con avances continuos en la IS, forzados por el rápido abaratamiento y aumento de capacidad del hardware.

Son los sucesivos fracasos de las distintas metodologías para:

- Dominar la complejidad del software, lo que implica el retraso de los proyectos de software
- Las desviaciones por exceso de los presupuestos fijados y la existencia de deficiencias respecto a los requisitos del cliente.

Complejidad del Desarrollo de Software

- Si se espera construir un *software de alta calidad*, debe *entender el problema que se quiere resolver*.
- También debe ser capaz de *crear un diseño que esté de acuerdo con el problema* y que al mismo tiempo tenga *características que lleven al software a las dimensiones y factores de calidad*.
- En la Ingeniería del Software existen actividades sombrillas que ayudan a controlar la calidad, el cambio y el riesgo del software.
    - Seguimiento y Control del Proyecto de Software
    - Gestión de Riesgo
    - Aseguramiento de la Calidad
    - Revisiones Técnicas Formales
    - Medición
    - Gestión de Configuración del Software
    - Gestión de Reutilización
    - Preparación y Producción del Producto de Trabajo

Casos Reales de la Mala Calidad

- **California DMV:** El Proyecto no fue soportado por una gestión ejecutiva • No intervino el usuario • Planificación pobre • Diseño pobre • Objetivos no claros • El Proyecto estuvo condenado desde el inicio
- **CONFIRM**: Problemas con la definición correcta de requisitos. n Falta de involucramiento de los usuarios n Constante cambios en requisitos y especificaciones
- **Hoteles Hyatt**: ÉXITO, Involucramiento del usuario • Ayuda de una gestión ejecutiva • Clara descripción de requisitos • Planificación adecuada • Hitos pequeños en el desarrollo
- **Banco Itamarati**: ÉXITO Visión clara con objetivos específicos documentados. • Alto involucramiento de todos en el proceso. • Manejo de una relación cercana con sus clientes. • No tuvo una descripción clara de los requisitos de software. • Tuvo una buena planificación. • Personal altamente calificado. • Utilización de buena tecnología.

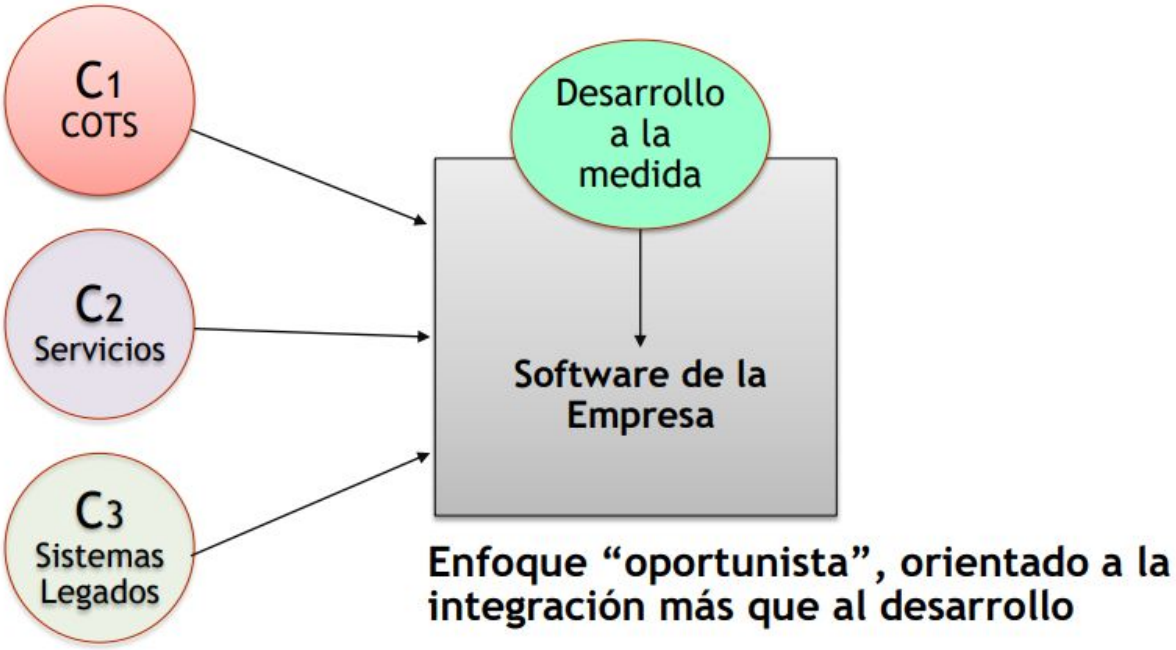
Conceptos asociados a la calidad

- ***Política de la calidad*:** intenciones y dirección de una organización relativas a la calidad tal como las expresan formalmente su alta dirección.
- ***Objetivo de la calidad*:** resultado a lograr en cuanto a la calidad.
- ***Sistema de gestión de la calidad:*** sistema de gestión para dirigir y controlar una organización con respecto a la calidad.

- ***Planificación de la calidad*:** parte de la gestión de la calidad enfocada al establecimiento de los objetivos de la calidad y a la especificación de los procesos operativos necesarios.
- ***Control de la calidad*:** parte de la gestión orientada al cumplimiento de los requisitos.
- ***Aseguramiento de la calidad*:** parte de la gestión de la calidad *orientada a proporcionar confianza en que se cumplirán los requisitos de la calidad.*
- ***Mejora de la calidad*:** parte de la gestión de la calidad orientada a *aumentar la capacidad de cumplir con los requisitos de la calidad.* La mejora continua es una actividad recurrente para mejorar el desempeño.

Capítulo 2 Sistemas Híbridos

La mayoría de software se puede construir mediante la integración de componentes de diversa naturaleza. • Comerciales • Código Libre • Componentes • Legados



Diversos componentes de proveedores externos a la organización que se integran con algún software hecho a medida.

Sistemas Híbridos: Los componentes de software utilizados en este tipo de enfoques arquitectónicos incluyen componentes desarrollados por terceros, generalmente conocidos como componentes "Off-The-Shelf" (OTS)

- Componentes comerciales (COTS)
- Componentes gratuitos y de código abierto (FOSS)
- Servicios Web
- Software desarrollado a la medida
- Sistemas legados

Componentes COTS (Commercial Off-The-Shelf)

"*Un componente es una unidad de composición de aplicaciones software, que posee un conjunto de interfaces y un conjunto de requisitos, y que ha de poder ser desarrollado, adquirido, incorporado al sistema y compuesto con otros componentes de forma independiente, en tiempo y espacio*" Clemens Szyperski

El adjetivo COTS se refiere a un tipo particular de componente, caracterizado por:

- ser de índole comercial,
- generalmente de grano grueso y de bajo coste,
- que es adquirido, seleccionado, probado, validado e integrado por desarrolladores de un sistema software basado en componentes para satisfacer ciertas necesidades del sistema, a partir de unos requisitos específicos.

Beneficios del Desarrollo de Software basado en Componentes

- Reutilización de Software
- Simplifica las pruebas
- Simplifica el mantenimiento
- Mayor calidad

- Ciclos de desarrollo más cortos
- Mejor ROI
- Funcionalidad mejorada
- Independencia SW y HW

Desventajas del Uso de Componentes

- No existe control sobre las nuevas versiones
- Costos asociados
    - Costos de licenciamiento
    - Costos de integración
    - Costos de soporte
- Licencias y propiedad intelectual
- Dependencia del fabricante
- La integración no es trivial

Integración / Adaptación / Parametrización

Tipos de parametrización

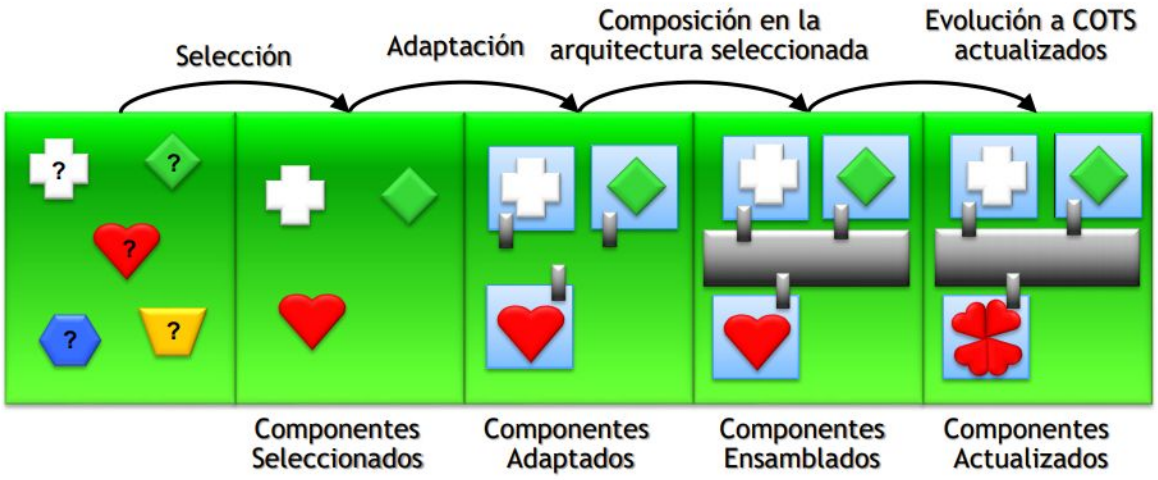
- Personalización
- Rendimiento
- Simplemente poner el componente a trabajar

La mayoría de componentes requieren algún tipo de parametrización: • MS-Office: un par de tardes • Sistemas ERP: varios meses

Dificultades

- Especificación de requisitos
- Selección de componentes adecuados
- Adaptación e integración a una arquitectura común
- Identificación de las necesidades estratégicas para las cuales es demandado el software
- Agrupación de servicios relacionados en dominios atómicos que estructuran la arquitectura genérica del sistema y describen la funcionalidad mínima cubierta por cada componente

Ciclo de Vida y Retos



Mercado en continua evolución:

- Cientos de nuevos productos y actualizaciones
- Nuevos productos no son idénticos a los anteriores
- Reemplazo de componentes puede ser muy difícil
- Reentrenamiento y testeo pueden ser requeridos

Mashups

El término mashup tiene su origen en el dominio musical refiriéndose a artistas que mezclan algunas piezas de música, usualmente desde diferentes estilos musicales, en un simple registro.

Los mashups son aplicaciones desarrolladas para integrar contenido y funcionalidad cuya fuente es la web. Integran elementos heterogéneos disponibles en la web tales como RSS|Atom feeds, Web Services, APIs, contenido traído desde sitios de terceros o widgets (tales como Google maps).

Casos de Estudio Mashups

Liveplasma, Caribbean Resorts, Skypicker

El Concepto de Calidad (i)

La calidad es un concepto que ha mantenido ocupado a los filósofos por más de 2000 años. Las raíces de describir y definir la calidad caen en la vista de la belleza según Plato. R.W.

Las discusiones anteriores pueden ser traducidas a un uso moderno de la palabra calidad.

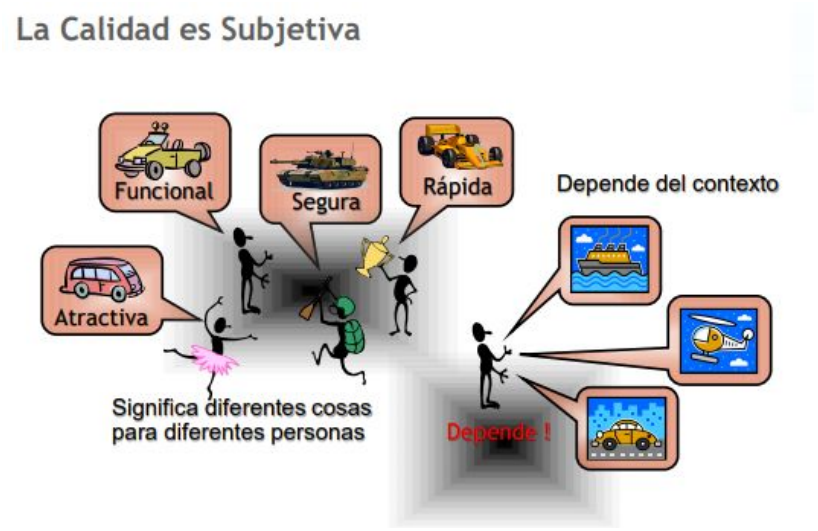
Originalmente esta no contenía una noción de evaluación como buena o mala, pero en el uso contemporáneo, significa que un software es bueno al decir que es un software de calidad.

La ISO/IEC e IEEE define la calidad de las siguientes maneras:

- El grado con el cual un sistema, componente o proceso reúne requisitos específicos.
- La habilidad de un producto, servicio, sistema, componente o proceso para reunir las necesidades, expectativas o requisitos de un cliente o usuario.
- La totalidad de características de una entidad que tratan de su habilidad para satisfacer las necesidades implicadas.
- La conformidad con las expectativas del usuario, conformidad de sus requisitos, satisfacción del cliente, confiabilidad y nivel de defectos presentes.
- El grado con el cual un sistema, componente o proceso reúne las necesidades o expectativas del usuario

¿Qué es más importante, reunir características de calidad para el usuario o para el cliente?

No está claro si la alta calidad significa satisfacer a la persona que usará el sistema o a la persona que pagará el sistema. Incluso si cumplimos con todos los requisitos explícitos, nuestro sistema no necesariamente tiene alta calidad porque no cumple con las expectativas del usuario.



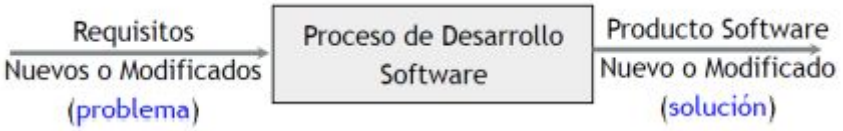
Concepto de Software

- "Instrucciones (programas de computadora) que cuando se ejecutan proporcionan la función y el rendimiento deseados".
- "Estructuras de datos que permiten a los programas manipular adecuadamente la información".
- "Documentos que describen la operación y el uso de programas".
- "Es un conjunto de elementos u objetos que forman una configuración que incluye Programas, Documentos y Datos"

"Programas de computadora, procedimientos, documentación y datos pertinentes a la operación de un sistema de cómputo" IEEE

¿Qué es un proceso de desarrollo de software?

- Definición de un conjunto de actividades cuyo objetivo es el desarrollo o evolución de un producto software.
- Actividades genéricas en todos los procesos de software.
  - ***Especificación:*** Lo que se espera que el sistema haga y las restricciones sobre ello.
  - ***Desarrollo***: Producción misma del producto o sistema de software.
  - ***Validación***: Comprobación con el cliente de lo que éste necesita.
  - ***Evolución***: Cambio en respuesta de las demandas organizacionales / cliente.



La Calidad de Software

- En general se entiende que un *producto de software* posee calidad adecuada si provee *valor (satisfacción) a los usuarios, produce una ganancia, genera pocas quejas por parte de sus clientes* y contribuye de alguna manera a los objetivos de la calidad (o por lo menos no es opuesto). Krasner
- Calidad es el conjunto de características de un producto que satisfacen las necesidades de los clientes y en consecuencia, hacen satisfactorio el producto.
- Calidad consiste en no tener deficiencias en el producto o proceso
- "La 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 o usuario". IEEE, Standard 610-1990.
- Un producto de software …
    - Se desarrolla, no se fabrica.
    - Se trata de un producto lógico, sin existencia física.
    - No se degrada con el uso.
    - Por la complejidad del SW y la ausencia de controles adecuados, se suele entregar el SW conscientemente con defectos (incluso públicamente declarados).
    - Un gran porcentaje de la producción se hace aún a medida en vez de emplear componentes existentes y ensamblar.
    - Es muy flexible. Se puede cambiar con facilidad e incluso reutilizar fragmentos

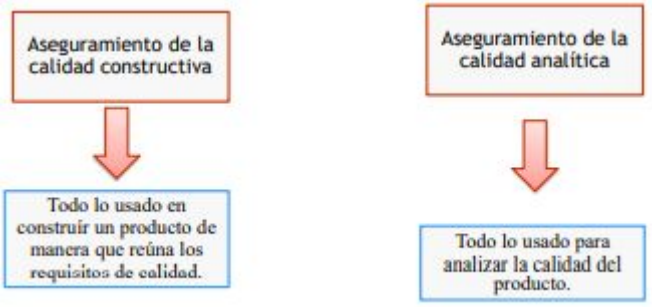
Trade-off

- Los atributos de calidad son frecuentemente conflictivos y aumentan los costes de desarrollo, por lo que hay una necesidad de ponderación y equilibrio (trade-off). *La Ingeniería del Software tiene como objetivo el desarrollo "rentable" de software de alta calidad.*

Aseguramiento de la Calidad

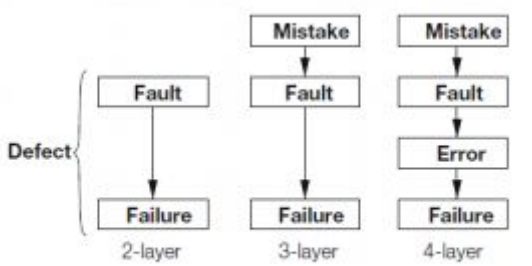
Definición:

- Patrón sistemático y planificado de todas las acciones necesarias para proveer una confianza adecuada de que un ítem o producto está de acuerdo a sus requisitos técnicos. (ISO/IEC/IEEE 24765:2010)
- Los estándares son importantes dado que contienen la encapsulación de las mejores prácticas, evitan errores del pasado, son un marco para procesos y proporcionan continuidad (personal nuevo entra a laborar y entiende la organización sólo conociendo los estándares que utilizan).



- ***Failure***-> Salida incorrecta, visible al usuario.
    - También se puede considerar a una desviación de un requerimiento.
    - Aquí es necesario incluir la expectativa del usuario.
- ***Fault***-> Causa de un potencial fallo.
    - Conocido también como bug
    - Cuando se trata de una omisión es muy difícil de detectar.
- ***Defect***-> Los defectos son los superconjuntos de failures y faults.
- ***Mistake***-> Acción humana que produce un fallo.

- ***Error***-> Es una parte del estado del sistema que puede ocasionar un failure



Testing can only reveal failures while inspections find faults

Para encontrar defectos empleamos técnicas de "*Verificación y Validación*"

- ***Verificación*** -> Actividad de evaluar si el resultado de un trabajo dado reúne los requisitos especificados.
- ***Validación*** -> Actividad de evaluar si el resultado de un trabajo y sus requerimientos reúnen las expectativas de los stakeholders.
- ***Análisis estático*** -> El proceso de evaluar un sistema o componente basado en su forma, estructura, contenido o documentación.
- ***Análisis dinámico*** -> El proceso de evaluar un sistema o componente basado en su comportamiento durante la ejecución.

Marco de Trabajo de la Gestión de la Calidad

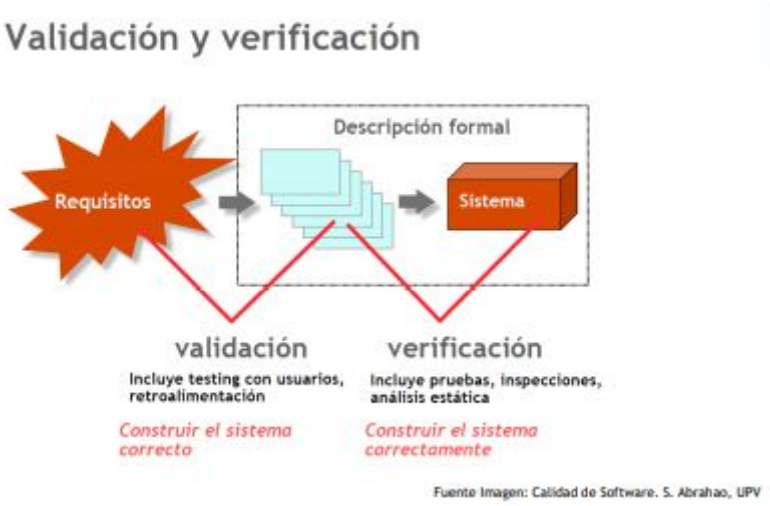
Ubicar la calidad desde una perspectiva en relación del desarrollo y adquisición de productos de software.
Los siguientes términos y sus definiciones son provistos para establecer la base de la calidad:

- Objeto: Los tipos de objetos o entidades a los cuales la calidad puede ser aplicada:
  - Producto
  - Proceso
  - Servicio
  - Recurso
  - Artefacto
  - Actividad
  - Medida o métrica
  - Ambiente
  - Colección de entidades u objetos
- Proceso
- Requisito / Requerimiento
- Usuario: Lo definimos como cliente o usuario final:
  - Usuario como cliente y usuario final
  - Usuario es representado por un comprador
  - Tanto el usuario final como el comprador pueden acceder a la organización de desarrollo o mantenimiento
- Evaluación
- Medida y Medición
- Calidad

Tipos de Calidad

Existen 3 tipos de calidad relacionados entre sí:

- **Calidad Necesaria:** Calidad que pide el cliente y la que le gustaría recibir.
- **Calidad Programada:** Es el nivel de calidad que se propone obtener el fabricante.
- **Calidad Realizada:** Es la calidad que se puede obtener debido a las personas que realizan el trabajo o a los medios utilizados



Triángulo de la calidad en el software

Triángulo de Hierro

En todo proyecto existen 3 variables relacionadas, es el llamado "triángulo de hierro"

- **El alcance:** cuántos requisitos o tareas hay que realizar
- **El tiempo o planificación*:*** cuánto durará el proyecto
- **El coste o recursos:** cuánto dinero, personas, etc

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 verificaciones y equilibrios que ayudan a evitar que el proyecto caiga en el caos.
   - Las prácticas de Ingeniería del Software ayudan al desarrollador analizar el problema y diseñar una solución sólida.
   - Las actividades sombrilla (administración del cambio y revisiones técnicas)
2. Un producto útil
   - Entrega contenido, funciones y características que el usuario final desea.
   - Entrega estos activos en forma confiable y libre de errores.
   - Satisface los requisitos
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.

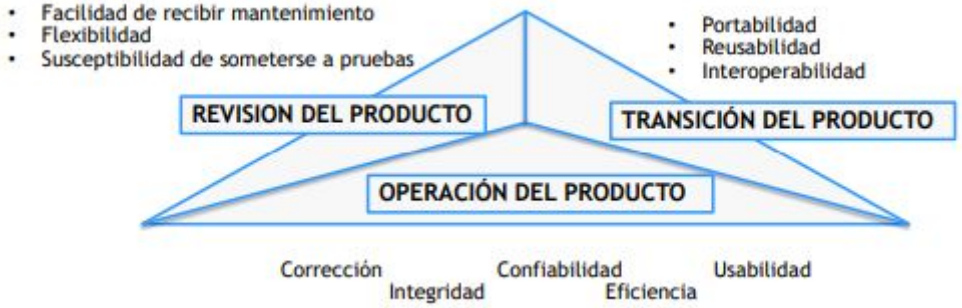
Dimensiones de la Calidad de Gavin

La calidad se aborda desde un punto de vista multidimensional.

- **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 la funcionalidad libre de errores?
- **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 disminuya con el tiempo?
- **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?
- **Servicio**: ¿Existe la posibilidad de que el software reciba mantenimiento (cambios) o correcciones (depuración) en un período aceptablemente breve, se puede adquirir la información necesaria para cambios?
- **Estética**: Subjetividad, aún así la mayoría concuerda en la elegancia, un flujo único y una "presencia"
- **Percepción**: Basada en el pasado y la reputación del fabricante del software.

Factores de Calidad de McCall

McCall, Richards y Walters proponen una clasificación útil de los factores que afectan la calidad del software.



Factores de Calidad de McCall

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

Calidad del Producto de Software

La calidad del producto significa que el producto debe reunir los requisitos de su especificación.
El software debe ser entregado con la funcionalidad requerida (functional requirements) con los atributos de calidad requeridos (non-functional requirements)
Puede existir tensión entre los requisitos de calidad de los clientes (eficiencia, confiabilidad, …) y los requisitos del desarrollador (mantenibilidad, reusabilidad)
Las especificaciones de software a menudo son incompletas e inconsistentes.
Los atributos de calidad están frecuentemente en conflicto e incrementan los costos

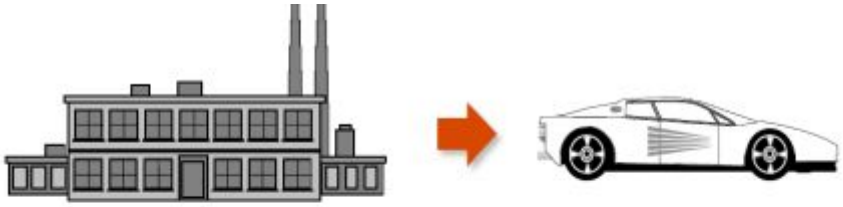
Los factores que intervienen de una u otra manera en la calidad del producto son:

- ***Calidad del proceso*** - Un buen proceso es usualmente produce un buen producto - En la manufactura de productos es vital un buen proceso
- ***Calidad del personal*** - Para pequeños proyectos, las capacidades de los desarrolladores son determinantes. - Tamaño del proyecto x Calidad de personal = Constante?
- ***Desarrollo de tecnología*** - Es particularmente significativo para proyectos pequeños
- ***Presupuesto y calendario*** - En todos los proyectos si una planificación de tiempo es no realista, la calidad del producto sufre sustancialmente.

Calidad de Proceso

- Hay poca evidencia en que cumplir un modelo de procesos asegure la calidad del producto, la estandarización de los procesos garantiza la uniformidad en la salida de los mismos lo que puede incluso institucionalizar la creación de malos productos. - Kitchenham, B. y Pfleeger, S. L. Se puede ver como una colección estructurada de prácticas que describen las características de un proceso efectivo. Se usa para:
    - Definir las prioridades y objetivos de mejora
    - Guía para la mejora
    - Definir un lenguaje común.
- Áreas del proceso relacionadas con la gestión de la calidad:
    - Aseguramiento de la calidad en el proceso y en el producto
    - Verificación del proceso
    - Validación del proceso

Calidad de Proceso vs Calidad de Producto



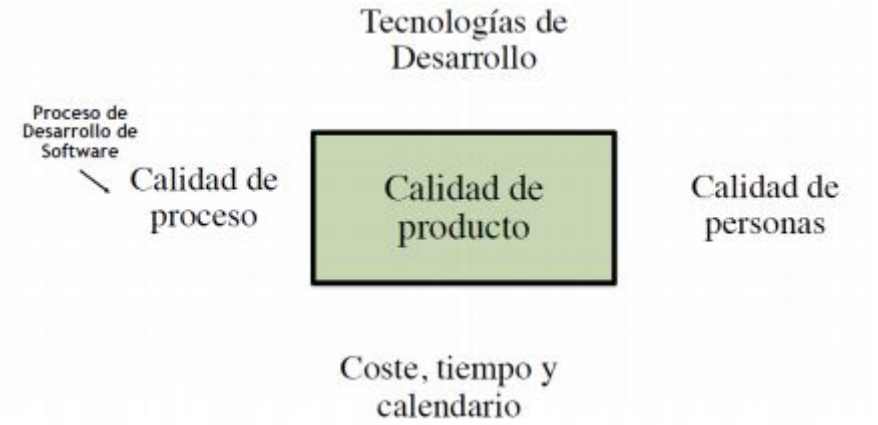
Calidad de Proceso

- El objetivo es introducir características de propósito especial para controlar:
  - Coste
  - Tiempo de Entrega
  - Calidad
  - Competitividad

Calidad de Producto

- Se centra en evaluar la calidad de los productos sin importar el proceso detrás de ellos.
- Los productos software deben ser capaces de satisfacer los **requisitos explícitos e implícitos de los clientes.**

Factores que afectan la calidad de productos



Capítulo 3: Modelos de Calidad de Software

- Los modelos de calidad han sido un tópico de investigación durante algunas décadas, estos son medios aceptados para soportar el control de calidad de los sistemas de software. El ISO/IEC 25010 es usado principalmente para definir la calidad, utilizado para evaluar la calidad de un sistema dado.

Ventajas de los Modelos de Calidad

- Corregir los procesos de software.
- Certificar la competitividad internacional requerida para competir en los mercados.
- Cambiar la actitud del personal de la empresa.
- Desarrollar y mejorar el nivel del personal.
- Lograr competitividad en una empresa de software.
- Reducir los costos en los procesos.
- Asegurar la satisfacción de los clientes.
- Tener productos de software con un valor agregado.
- Tener aceptación de los clientes.

Que es un modelo de calidad del software

- Es: “El conjunto de características y las relaciones entre ellas que proveen la base para la especificación de los requisitos de calidad y la evaluación de la calidad.” ISO/IEC 8402.
- Los modelos de calidad permiten:
  - Definición estructurada de criterios de evaluación
  - Especificación de requisitos con relación a ellos
  - Descripción de componentes en un marco común
  - Definición de métricas y prioridades

Métricas - Concepto

- Las métricas del producto se dividen en dos clases:
  - ***Métricas dinámicas***, que son recogidas por las mediciones hechas en un programa en ejecución.
  - ***Métricas estáticas***, que son recogidas por las mediciones hechas en las representaciones del sistema como el diseño, el programa o la documentación.
- Las métricas del software se pueden clasificar en MEDIDAS DIRECTAS e INDIRECTAS.
  - ***Directas:*** Una métrica de un atributo que no depende de ninguna métrica de otro atributo.
  - ***Indirecta:*** Se deriva de una o más métricas de otros atributos.

Ejemplo de métricas directas

- Longitud del Texto del Cuerpo de una Página
  - Medido por cantidad de palabras, etc.
- Cantidad de Enlaces Rotos Internos
  - Medidos por la presencia de errores del tipo 404, (410 ?)
- Cantidad de Imágenes con Texto Alternativo
  - Medido por la presencia de la etiqueta ALT (con texto no nulo) en cada una de las imágenes vinculadas a las páginas de un sitio Web

Ejemplo de métricas indirectas

- Porcentaje de Enlaces Rotos de un Sitio

$$PorcentajeEnlacesRotos = \frac{CantidadEnlacesRotosInternos + CantidadEnlacesRotosExternos}{CantidadTotalEnlaces} \times 100$$

- Porcentaje de Presencia de la propiedad ALT.

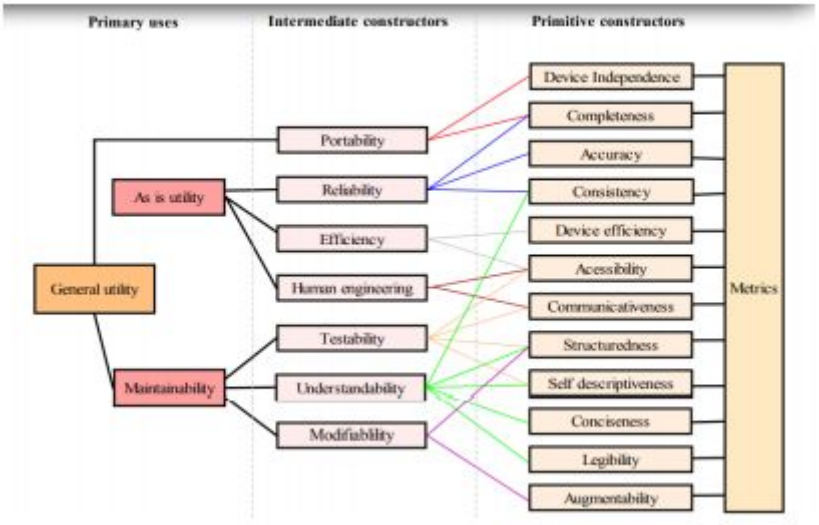
$$PorcentajePresenciaALT = \frac{CantidadImágenesALT}{CantidadTotalImágenes} \times 100$$

• Estructura de los modelos de calidad del software

Modelos de Calidad: Estructura

- Todos los modelos de calidad comparten:
  - Un catálogo de factores de calidad (fijo? desechable?)
  - Diferentes niveles de abstracción (Numero de capas? jerarquía? grafo?)
- Algunos autores recomiendan su descripción en forma de un modelo conceptual que describa:
  - La forma del modelo
  - Propiedades de las métricas
  - Elementos medibles
  - Aspectos de formalización (definiciones)

Modelos de Calidad (Boehm – 1978)



Tipos de Modelos de Calidad

Existen algunos tipos de modelos de calidad:

- ***Modelos de Calidad Jerárquicos***:
  - El primero publicado fue en 1970.
  - Usan una descomposición Jerárquica en factores de calidad (Mantenibilidad, Confiabilidad)
  - Uno de los más populares es el modelo FURPS
    - Funcionalidad - Functionality
    - Usabilidad - Usability
    - Confiabilidad - Reliability
    - Rendimiento - Performance
    - Soporte – Supportability

      FURPS:
      - El principal objetivo de FURPS es una descomposición y checklist
      - Ayuda a definir la calidad como base para los requisitos.
  - *La principal idea es que se pueda descomponer la calidad a un nivel donde ésta pueda ser medida y de ahí evaluada.*
  - Esta clase de modelos trajeron las bases para el estándar ISO / IEC 9126 en 1991, la ISO/IEC 25010, mantiene una nueva clasificación pero guarda la descomposición jerárquica general.

- ***Modelos de Calidad Basados en Meta-Modelos:***
  - Describen cómo modelos de calidad válidos son estructurados.
  - Incluyen mediciones y evaluaciones.
  - Los modelos de calidad basados en meta-modelos muestran el concepto complejo de las necesidades de calidad más estructuras en modelos de calidad que abstraen características y métricas.
- ***Modelos de Calidad Implícitos:***
  - Capturan las propiedades del producto, proceso u organización.
  - Estiman y predicen esos factores de calidad.
  - Un ejemplo de esos modelos son los reliability growth models o los maintainability index (MI), un modelo de regresión desde las métricas de código o Vulture, un modelo de aprendizaje de máquina basado en bases de datos de vulnerabilidad y archivos de versión

Tipos de modelos de calidad

- **Modelos fijos:** Existe un catálogo de partida del cual se elige un subset de características de calidad
  - Pros: reutilizable, comparable, rápido de utilizar
  - Contras: inflexible
  - Ejm: Modelo de McCall, Boehm, FURPS
- **Modelos a la medida :** Determinación de factores de calidad basada en necesidades del contexto
  - Pros, contras: Lo contrario del caso anterior
  - Ejm: IEEE 1061(1998), Goal Question Metric (GQM)
- **Modelos mixtos:** Un modelo de alto nivel que puede ser refinado
  - Pros, contras: balanceados

Modelos de Definición: Son usados en varias fases de un proceso de desarrollo de software.

- **Durante la Ingeniería de Requisitos:** Definen factores de calidad y requisitos para sistemas de software. Constituyen un método para acordar con el cliente la calidad.
- **Durante la implementación:** Sirven como base para modelar y codificar. Proveen recomendaciones directas sobre la implementación y constituyen enfoques constructivos para conseguir alta calidad de software.

Modelos de Evaluación

- Extienden los modelos de definición.
- Evalúa la calidad del modelo de definición
- Los modelos de evaluación pueden ser usados durante la ingeniería de requisitos para especificar y controlar los requisitos de calidad.
- Constituyen la piedra angular para las certificaciones de calidad.

Modelos Predictivos

- Sirven para predecir el número de defectos de un sistema o módulos específicos, tiempos medios entre fallos, tiempo de reparación y esfuerzos de mantenimiento.
- Ejemplo: Modelo RGMs emplean detección de defectos desde las fases de prueba y operación para predecir la futura confiabilidad de los sistemas de software.

Modelos Multi-Propósito

- Modelos de calidad que integran los 3 propósitos.
- Su ventaja es que se evalúa y predice en el mismo modelo los requisitos de calidad.
- Asegura una alta consistencia.
- Ejemplo: COQUAMO

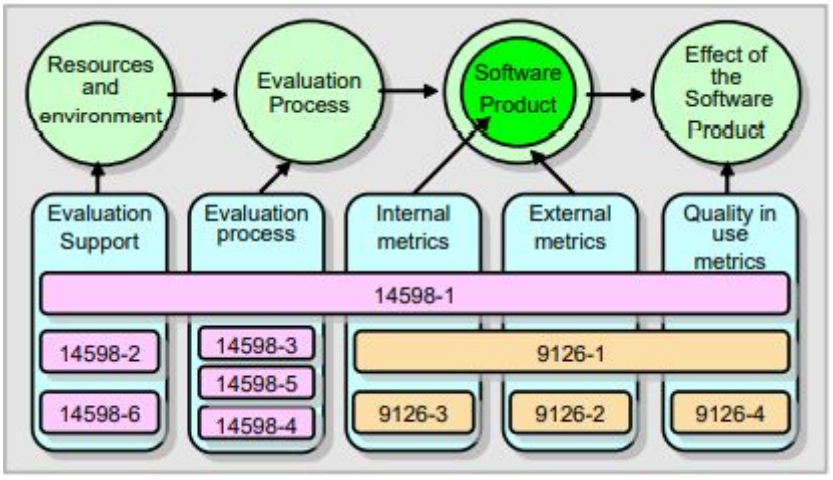
Estándares de modelos de calidad del software

Nivel de Calidad	Modelo de Calidad	Estándar de Calidad
Proceso	CMMi, TickIT, Bootstrap, Personal SW Process (PSP), Team SW Process (TSP), Practical SW Measurement (PSM), Six Sigma for Software	ISO 90003, ISO 12207, ISO 15504 (SPICE), IEEE/EIA 12207, ISO 20000, ITIL, Cobit 4.0
Producto	Gilb, GQM, McCall, Furps, Bohem, SATC, Dromey, C-QM, Metodología SQA, Web EQM	ISO 9126-1, ISO 25000 (SQuaRE), IEEE Std 1061-1998

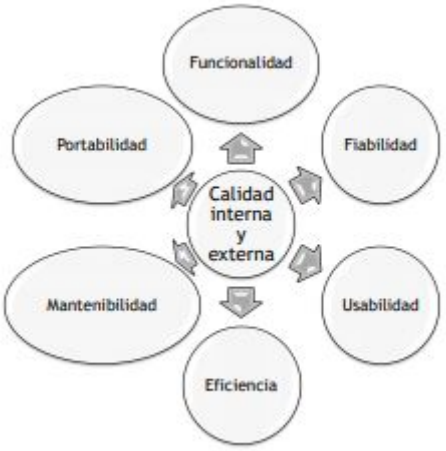
Estándares de modelos de calidad: ISO/IEC 9126

- Modelo mixto con un catálogo de partida más elaborado:
- 6 características, 27 subcaracterísticas…descomponibles en atributos (jerarquía multinivel)
- Antiguamente un estándar único: • ISO/IEC 9126, 1991
- Actualmente un estándar multiparte:
    - ISO/IEC 9126: Software quality (part1 1, 2001; 2&3, 2003; 4: 2004)
    - ISO/IEC 14598: Software Product Evaluation (6 partes)
- Recientemente reemplazado: • ISO/IEC CD 25000, SQuaRE (Software Quality Requirements and Evaluation)

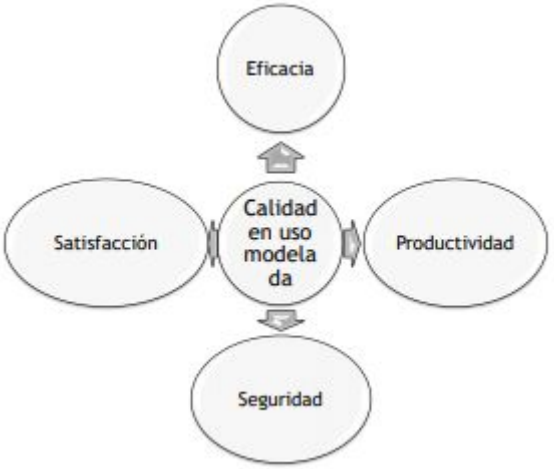
ISO / IEC 9126 y 14598



ISO IEC 9126-1



- Calidad en uso modelada



ISO / IEC 9126-2

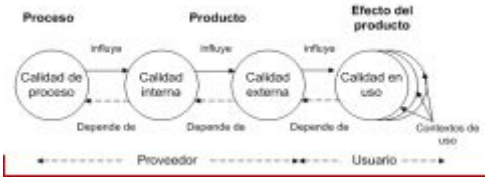
Describe las métricas externas que son utilizadas para especificar o evaluar el comportamiento del software cuando es operado por el usuario

ISO / IEC 9126-3

Esta parte describe las métricas internas que se pueden utilizar para describir propiedades internas, que pueden ser evaluadas por la inspección sin poner en funcionamiento el software.

ISO / IEC 9126-4

Esta parte describe las métricas de calidad en uso que se pueden utilizar para especificar o evaluar el efecto del producto software cuando son operados por el usuario en determinados contextos de uso.



ISO / IEC 14598

ISO/IEC 14598-1 Visión general de todo el estándar y explicación de las diferencias entre la evaluación del producto software y el modelo de calidad definido en la ISO / IEC 9126.

ISO/IEC 14598-2 Requisitos y guías para las funciones de planificación y gestión de la evaluación del producto. ISO/IEC 14598-3 Requisitos y guías para la evaluación del producto software cuando la evaluación se lleva a cabo en paralelo al desarrollo del mismo.

ISO/IEC 14598-4 Requisitos y guías para la evaluación del producto software cuando este ha sido adquirido y se requiere utilizar un producto existente o pre-desarrollado.

ISO/IEC 14598-5 Requisitos y guías para la evaluación del producto cuando esta es llevada a cabo por evaluadores independientes. ISO/IEC 14598-6 Provee las guías para la documentación del módulo de evaluación.

Estándar ISO 25000 (SQuaRE)

Revisa el ISO / IEC 9126 e incorpora las mismas características de calidad con algunas enmiendas.

Aplicaciones de los Modelos de Calidad

- Aplicaciones exploradas por diversos autores:
  - Especificaciones de software
  - Diseño arquitectónico del software

- Soporte a la implementación del software
- Soporte a la evaluación del software
- Soporte para la certificación del software
- Identificación de riesgos
- Otros:
  - Soporte a decisiones económicas en relación al rendimiento del software

Capítulo 4
Construcción de Modelos de Calidad de Software

- La calidad no es una propiedad fija de un sistema de software, depende de las necesidades y objetivos de los stakeholders.
- Se deben mapear esas necesidades a propiedades técnicas.
- No solamente planearemos qué calidad queremos tener sino adicionalmente cómo la construiremos y la aseguraremos.

Alternativas para abordar la construcción de modelos de calidad

- Los requisitos no-funcionales describen las propiedades que no representan la funcionalidad primaria del sistema
- El problema es cómo elicitar y evaluar los requisitos de calidad en una forma estructurada y comprensible
- Los principales enfoques para elicitar requisitos de calidad son:
  - Chequear diferentes tipos de requisitos y construir prototipos
  - Usar escenarios positivos y negativos (Casos de Uso)
  - Refinar metas de calidad por gráficos de objetivos
- Un ejemplo para chequear tipos de requisitos son los checklists para SLAs.
- Especificar requisitos de calidad y construir modelos de calidad pueden fusionarse en un solo proceso.

Factores técnicos

- Descomposición de factores en factores de más bajo nivel de abstracción
- Los factores técnicos se refieren a la calidad intrínseca del producto de software.

Factores No-Técnicos

Carvallo, Franch y Quer destacan que se pueden incluir en los modelos de calidad factores no-técnicos, y ellos así lo hacen; como ejemplo tenemos:

- **Proveedor**: Características del proveedor que pueden influenciar en la calidad del producto de software
- **Negocio**: Características del contrato entre proveedor y cliente que pueden influenciar la calidad del producto del software
- **Producto**: Características de los aspectos comerciales del producto que pueden influenciar su calidad.

Métricas del Software

- Las métricas son un buen método para medir, entender, monitorear, predecir y probar el desarrollo de software y los proyectos de mantenimiento (Briand et al., 1996)
- La medición persigue tres objetivos fundamentales:
  - **Entender** qué ocurre durante el desarrollo y el mantenimiento.

- **Controlar** qué es lo que ocurre en los proyectos de desarrollo.
- **Mejorar** los procesos y productos.

Problemas:

- Todavía se falla en dar objetivos medibles cuando desarrollamos productos de software.
- Fallamos en medir diferentes componentes para calcular costes reales de proyectos.
- No intentamos cuantificar la calidad de los productos que producimos.
- Solemos ver informes que hacen afirmaciones como que el 80% de los costes del software son de mantenimiento o que hay una media de 55 errores en cada 1.000 líneas de código.
- Pero no se dice:
  - Cómo se han obtenido esos resultados
  - Cómo se han ejecutado los experimentos
  - Qué entidades fueron medidas y cómo
  - Los márgenes de error.

Atributos Medibles

- En software hay tres clases de entidades cuyos atributos podemos medir:
  - **Procesos**: actividades del desarrollo:
    - El tiempo (duración del proceso)
    - El esfuerzo (asociado al proceso)
    - El número de incidentes de un tipo específico
  - **Productos**: entregables, artefactos o documentos generados en el ciclo de vida del software:
    - La fiabilidad del código
    - La entendibilidad de un documento de especificación
    - La mantenibilidad de un código fuente. • La longitud, funcionalidad, modularidad o corrección sintáctica de los documentos de especificación
  - **Recursos**: todos aquellos elementos que hacen de entrada a la producción de software:
    - El personal
    - Los materiales
    - Las herramientas y métodos
    - El coste
    - La productividad

Utilidad de las Métricas del Software

- Estimación del esfuerzo de los Modelos y medidas de productividad
- Modelos y medidas de calidad
- Modelos de fiabilidad
- Modelos de evaluación y rendimiento
- Métricas de complejidad estructural

Métodos de construcción de modelos de calidad

Pasos para construir el modelo a través de la Ingeniería de Requisitos

1. Identificar a los stakeholders relevantes
2. Definir metas generales
3. Analizar documentos relevantes y elicitar nueva información
4. Escoger y definir actividades
5. Definir metas de calidad
6. Identificar artefactos afectados y escoger entidades
7. Escoger entidades y analizar material relevante
8. Escoger factores y definir nuevas características de producto
9. Especificar requisitos de calidad

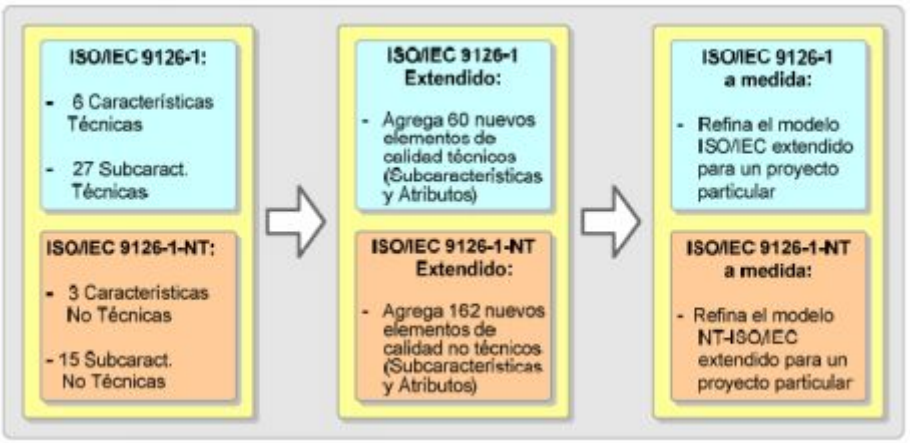
Otras propuestas..

- Existen propuestas generales:
  - Bohem-78: 6 pasos
  - GQM: 5 pasos
  - Dromey: 5 pasos

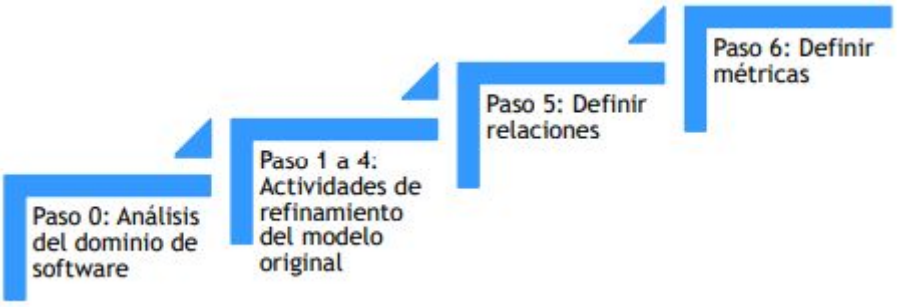
El método IQMC para la construcción de modelos de calidad

IQMC : Conjunto de guías y técnicas para la identificación de los factores de calidad apropiados que deben ser incluidos en un modelo de calidad que permita analizar la calidad de componentes pertenecientes a un cierto dominio de software.

Extensión del ISO 9126-1

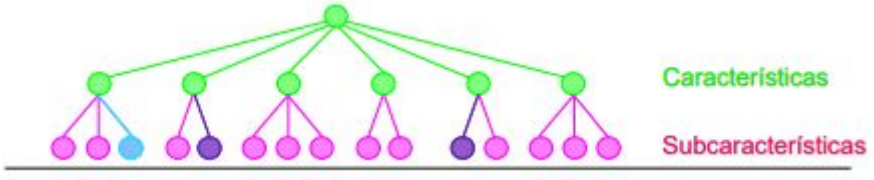


Método IQMC



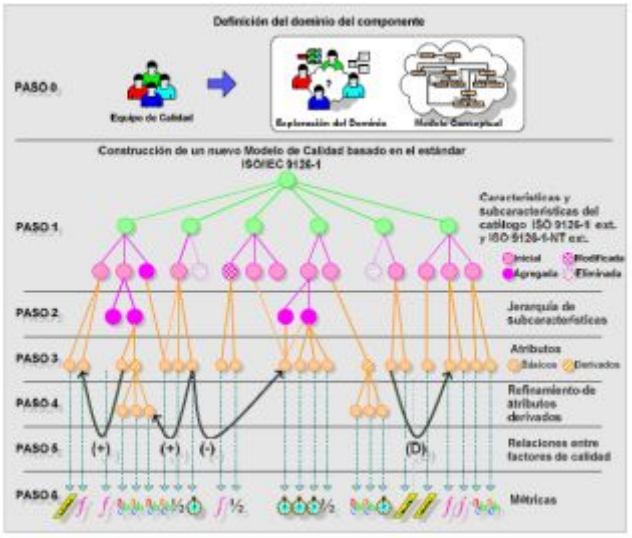
- El estándar IQMC consiste en 7 pasos que aunque se presentan como secuenciales, pueden ser simultáneos de ser necesario.
- Es importante hacer hincapié en la diferencia de detalle que existe en el catálogo entre los factores técnicos y no-técnicos.
- **Paso 0**
  - Estudio del ámbito del software.
  - Paso opcional que puede soslayarse en caso de poseer el conocimiento suficiente.

- **Paso 1**
  - Determinación de subcaracterísticas de calidad. Teniendo en cuenta que partimos del catálogo ISO/IEC 9126-1 extendido, el añadido de subcaracterísticas no será muy habitual.



- **Paso 2**
  - Refinamiento de la jerarquía de sub-características. Se descomponen las sub-características del más bajo nivel de abstracción formando jerarquías de sub-características
  - Al igual que en el paso anterior, el añadido de sub-características no será muy habitual, excepto en el caso de la descomposición de la subcaracterística.
- **Paso 3**
  - Refinamiento de sub-características en atributos.

    - Descomposición de esos atributos de calidad en unos más concretos.

- **Paso 4**
    - Atributos básicos y derivados
    - *Refinamiento de atributos derivados en básico*s. Se descomponen los atributos derivados hasta obtener los atributos básicos.
- **Paso 5**
    - Determinar las métricas para los atributos básicos.
    - Establecimiento de relaciones entre factores de calidad.
    - Permiten conocer las dependencias entre los distintos factores de calidad del modelo.
- **Paso 6**
    - Determinación de métricas para los atributos. Se determinan las métricas para los atributos identificados.
- 

Goal Question-Metric

- Técnica definida por Basili y Weiss (1984) y Rombach (1990), para seleccionar y generar métricas tanto del proceso como de los resultados del proyecto.

Fases del método GQM

- **Planificación**: Se selecciona, define, caracteriza y planifica un proyecto para la aplicación de la medición, obteniéndose como resultado un plan del proyecto.
- **Definición**: Se define y documenta el programa de la medición (objetivos, preguntas, métricas e hipótesis).
- **Recopilación de Datos**: Se reúnen los datos reales de la medición.
- **Interpretación:** Se procesan los datos recopilados respecto a las métricas definidas en forma de resultados de medición, que proporcionan respuestas a las preguntas definidas a partir de las cuales se puede evaluar el logro del objetivo planteado.

Niveles GQM

- Se constituyen 3 niveles dentro de esta técnica:
    - Nivel Conceptual (Objetivos-Goals)
    - Nivel Operacional (Preguntas – Questions)
    - Nivel Cuantitativo (Métricas – Metrics)

Capítulo 5: Métricas

Entidad

- Un objeto que va a ser caracterizado mediante una medición de sus atributos.

- Una entidad puede ser un proceso, un producto, un servicio, un proyecto, un servicio o un recurso concreto.
- Una entidad puede ser física –tangible- (p.e. un ordenador) o abstracta (p.e. un programa en C)
- Relaciones:
    - Una entidad puede pertenecer a una o más categorías de entidad.
    - Una medición se realiza sobre los atributos de una entidad.

Atributo

- Una propiedad medible (mensurable), física o abstracta, que comparten todas las entidades de una categoría de entidad.
- Relaciones:
    - Un atributo puede pertenecer a una categoría de entidad.
    - Una medición se realiza sobre los atributos de una entidad.
    - Un atributo tiene definida cero, una o varias métricas.

Forma de medir

- Conjunto de operaciones cuyo objeto es determinar el valor de una medida. Una forma de medir puede ser un método de medición, función de cálculo o modelo de análisis.

Métrica

- Una forma de medir (método de medición, función de cálculo o modelo de análisis) y una escala, definidas para realizar mediciones de uno o varios atributos.
- Relaciones:
    - Una métrica está definida para uno o más atributos.
    - Dos métricas pueden relacionarse mediante una función de transformación.
    - Una métrica puede expresarse en una unidad (sólo para métricas cuya escala sea de tipo intervalo o ratio).

Métrica Directa

- Una métrica de la cual se pueden realizar mediciones sin depender de ninguna otra métrica y cuya forma de medir es un método de medición.

Métrica Indirecta

- Una métrica cuya forma de medir es una función de cálculo, es decir las mediciones de dicha métrica utilizan las medidas obtenidas en mediciones de otras métricas directas o indirectas.

Indicador

- Una métrica cuya forma de medir es un modelo de análisis, es decir, las mediciones de dicha métrica utilizan las medidas obtenidas en las mediciones de otras métricas (directas, indirectas o indicadores) junto con criterios de decisión.

Medición (Acción de Medir)

- La acción que permite obtener el valor de una medida para un atributo de una entidad, usando una forma de medir.

Medida

- Resultado de una medición.

Método de Medición

- La forma de medir una métrica directa. Secuencia lógica de operaciones, descritas de forma genérica usadas para realizar mediciones de un atributo respecto de una escala específica.

Instrumento de Medición

- Instrumento que asiste o es útil a un método de medición.
- Relaciones: Un instrumento de medición asiste a uno o más métodos de medición.

Necesidad de información

- Información necesaria para gestionar un proyecto (sus objetivos, hitos, riesgos y problemas).
- Relaciones:
    - Una necesidad de información se asocia con un concepto medible.
    - Una necesidad de información se satisface con uno o más indicadores.

Modelo (de calidad)

- Un marco conceptual que especifica una serie de conceptos medibles y sus relaciones, para una determinada categoría de entidad.
- Relaciones:
    - Un modelo (de calidad) está definido para una determinada categoría de entidad.

- Un modelo (de calidad) evalúa uno o varios conceptos medibles.

Unidad

- Una cantidad particular, definida y adoptada por convenios, con la que se puede comparar otras cantidades de la misma clase para expresar sus magnitudes respecto a esa cantidad particular.
- Relaciones:
  - Una unidad sirve para expresar una o varias métricas cuyo tipo de escala sea intervalo o ratio.

Escala

- Un conjunto de valores con propiedades definidas.
- Relaciones:
  - Toda escala es de un cierto "Tipo de Escala".

Tipo de escala

- Indica la naturaleza de la relación entre los valores de la escala [ISO 15939]
- Relaciones • A un tipo de escala pertenecen una o más escalas.
- Ejemplos • Nominal, ordinal , intervalo, ratio y absoluta.

Función de cálculo

- La forma de medir una métrica indirecta. Algoritmo o cálculo realizado para combinar dos o más métricas directas y/o indirectas.
- Relaciones:
  - Una función de cálculo usa cero o más métricas directas.
  - Una función de cálculo usa cero o más métricas indirectas.
  - Una función de cálculo utiliza al menos una métrica (sea directa o indirecta)
  - Una función de cálculo define una o más métricas indirectas.

$LCFH = \frac{LCF}{HPT}$ [Métrica indirecta definida en base a 2 métricas directas]

$HTP = \sum_{dias} HPD$
[Métrica indirecta definida en base a sólo una métrica directa]

Criterio de Decisión

- Valores umbrales, objetivos, o patrones usados para determinar la necesidad de una acción o investigación posterior, o para describir el nivel de confianza de un resultado dado.
- Relaciones:
  - Un criterio de decisión es utilizado en uno o más modelos de análisis.

- LCFH/LCFHvm < 0'70 => PROD='muy baja'.
- 0'70 ≤ LCFH/LCFHvm < 0'90 => PROD='baja'.
- 0'90 ≤ LCFH/LCFHvm < 1'10 => PROD='normal'.
- 1'10 ≤ LCFH/LCFHvm < 1'30 => PROD='alta'.
- 1'30 ≤ LCFH/LCFHvm => PROD='muy alta'.

Modelo de Análisis

- La forma de medir un indicador. Algoritmo o cálculo realizado para combinar una o más métricas (directas, indirectas o indicadores) con criterios de decisión asociados.
- Relaciones:
  - Un modelo de análisis define uno o más indicadores.
  - Un modelo de análisis utiliza uno o más criterios de decisión.
  - Un modelo de análisis usa una o más métricas.