INTRODUCCION A LA INGENIERÍA DE SOFTWARE

UNIDAD I

UNIVERSIDAD EVANGÉLICA BOLIVIANA
Jeremias 32:17

CONTENIDO

2.1. Definición de Ingeniería de Software
2.2. Historia de la Ingeniería de Software
2.3. Características del Software
2.4. Mitos del Software
2.5. Capas de la Ingeniería de Software
2.6. El proceso de Software
2.7. Software de alta calidad
2.8. Factores de calidad y productividad

…Típica apariencia del estudiante promedio cuando le preguntan acerca de Ingeniería de Software…

Introducción

- El término de *Ingeniería de Software* fue introducido a finales de los 60 a raíz de la crisis del software. Esta crisis fue el resultado de la introducción de la tercera generación del hardware.

- El hardware dejo de ser un impedimento para el desarrollo de la informática; redujo los costos y mejoro la calidad y eficiencia en el software producido

Introducción (continuación)

La crisis se caracterizo por los siguientes problemas:

- Imprecisión en la planificación del proyecto y estimación de los costos.
- Baja calidad del software.
- Dificultad de mantenimiento de programas con un diseño poco estructurado, etc.

Por otra parte se exige que el software sea eficaz y barato tanto en el desarrollo como en la compra.

También se requiere una serie de características como fiabilidad, facilidad de mantenimiento y de uso, eficiencia, etc.

2.1. DEFINICIÓN DE IS

- **Fritz Bauer, 1969:** *Más que una disciplina o una parte del conocimiento, La Ingeniería es un verbo, una palabra de acción, un modo de enfocar el problema.*
- *La Ingeniería del Software es el establecimiento y uso de principios robustos de la ingeniería a fin de obtener económicamente software que sea fiable y que funcione eficientemente sobre maquinas reales.*

DEFINICIÓN DE IS

- **Bohem, 1976:** *Ingeniería del Software es la aplicación practica del conocimiento científico en el diseño y construcción de programas de computadora y la documentación necesaria requerida para desarrollar, operar (funcionar) y mantenerlos.*

- **Mills, 1980:** *La Ingeniería de Software tiene como uno de sus principales objetivos la producción de programas que cumplan las especificaciones, y que se demuestren correctos, producidos en el plazo y costo adecuado*

DEFINICIÓN DE IS

- **Meyer, 1988:** La Ingeniería de Software es la *producción de software de calidad*.

- **IEEE 1993:** *La Ingeniería de Software es la aplicación de un enfoque sistemático, disciplinado y cuantificable hacia el desarrollo, operación y mantenimiento del software; es decir, la aplicación de Ingeniería de Software.*

RESUMIENDO…

- La ingeniería de software es una aplicación práctica del conocimiento científico para proveer metodologías y técnicas que ayuden a desarrollar sistemas de software a tiempo, y a su vez que aseguren que el desarrollador cumpla con las expectativas de calidad y permanezca dentro del presupuesto.

2.2. HISTORIA DE LA ING DE SW

- Ingeniería del Software, es el término utilizado por Fritz Bauer en la primera conferencia sobre desarrollo de software patrocinada por el Comité de Ciencia de la OTAN celebrada en Garmisch (Alemania), en octubre de 1968, previamente había sido utilizado por el holandés Edsger Dijkstra en su obra *The Humble Programmer*.
- Puede definirse según Alan Davis como "*la aplicación inteligente de principios probados, técnicas, lenguajes y herramientas para la creación y mantenimiento, dentro de un coste razonable, de software que satisfaga las necesidades de los usuarios*".

Historia de la ing de sw (continuación)

- Su origen se debió a que el entorno de desarrollo de sistemas software adolecía de:
  - Retrasos considerables en la planificación
  - Poca productividad
  - Elevadas cargas de mantenimiento
  - Demandas cada vez más desfasadas frente a las ofertas
  - Baja calidad y fiabilidad del producto
  - Dependencia de los realizadores

Historia de la ing de sw (continuación)

- Esto es lo que se ha denominado habitualmente "crisis del software", que históricamente se generó en los siguientes pasos:
-

  - Primera Fase. Los albores (1945-1955)

  Programar no es una tarea diferenciada del diseño de una máquina
  Uso de lenguaje máquina y ensamblador.

  - Segunda Fase. El florecimiento (1955-1965)

  Aparecen multitud de lenguajes
  Se pensaba que era posible hacer casi todo.

Historia de la ing de sw (continuación)

- Tercera Fase. La crisis (1965-1970)

  Desarrollo inacabable de grandes programas
  Ineficiencia, errores, coste impredecible
  Nada es posible.

  - Cuarta Fase. Innovación conceptual (1970-1980)

  Fundamentos de programación
  Verificación de programas
  Metodologías de diseño.

  - Quinta Fase. El diseño es el problema (1980-?)

  Entornos de programación
  Especificación formal
  Programación automática.

Historia de la ing de sw (continuación)

- **¿Cómo se define crisis?**

  La palabra crisis se define en el diccionario como "*un punto decisivo en el curso de algo; momento, etapa, o evento decisivo o crucial*". Sin embargo para el software no ha habido ningún punto crucial, sólo una lenta evolución.

  La crisis en la industria del software permanece durante muchos años, lo cual parece una contradicción para el término. Lo que si se podría decir es que **hay un problema crónico en el desarrollo de software**.

  Que ha venido originado por una falta de:

- Formalismo y metodología
- Herramientas de soporte
- Administración eficaz

Historia de la ing de sw (continuación)

- ***Actualmente*** está surgiendo una gran expectativa ante la evolución de la **Ingeniería del Software**, al ir apareciendo nuevos métodos y herramientas formales que van a permitir en el futuro un planteamiento de ingeniería en el proceso de elaboración de software.
- Dicho planteamiento permitirá dar respuesta a los problemas de:
  - Administración
  - Calidad
  - Productividad
  - Fácil mantenimiento

- Este último es uno de los grandes problemas, pues puede llegar a suponer un importe superior al 60% del total del coste del software.

Porque se crea la Ingeniería de Software??

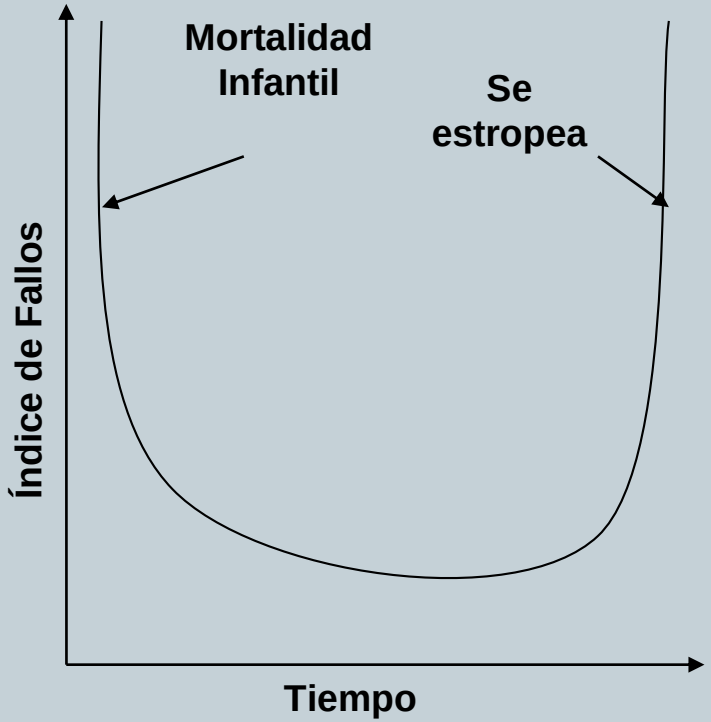
- La ingeniería de software se crea debido a las siguientes características:
- El producto debe ser confiable y realizar sólo las tareas especificadas en los requerimientos.
- El producto debe ser robusto. Esto quiere decir que el software se comporta de manera razonable, incluso en circunstancias no anticipadas desde el principio.
- El producto de software debe ser lo más reutilizable posible, de manera tal que pueda ser incorporado en otro producto de software si se requiere.
- El producto de software debe ser eficiente en el uso de los recursos del sistema.
- Se requiere desarrollar el software en una manera que lo haga evolutivo, de forma tal que se pueda agregar funcionalidad adicional sin efectos perjudiciales.
- El producto de software debe cumplir con los requerimientos de rendimiento especificados, es decir, debe cumplir algunas de las restricciones relacionadas al rendimiento.
- El producto de software tendrá mayor valor si es portable, es decir que puede trabajar bajo diferentes plataformas de software y ambientes (hardware, sistemas operativos, etc.).
- El producto de software debe ser utilizable, es decir, el aprendizaje de su uso debe ser los suficientemente sencillo por parte de personas no especialistas.

2.3. CARACTERÍSTICAS DEL SOFTWARE

- **El software se desarrolla o construye; no se fabrica.**
  - Aunque existen similitudes entre el desarrollo del software y la construcción del hardware, ambas actividades son fundamentalmente diferentes.
  - En ambas actividades la buena calidad se adquiere mediante un buen diseño, pero la fase de construcción del hardware puede introducir problemas de calidad que no existen (o son fácilmente corregibles) en el software.
  - Ambas actividades dependen de las personas, pero la relación entre las personas dedicadas y el trabajo realizado es completamente diferente para el software.
  - Ambas actividades requieren la construcción de un "producto" pero los enfoques son diferentes.
  - Los costos del software se concentran en la ingeniería. Esto significa que los proyectos de sw no se pueden manejar como si fueran proyectos de fabricación.

2.3. CARACTERÍSTICAS DEL SOFTWARE

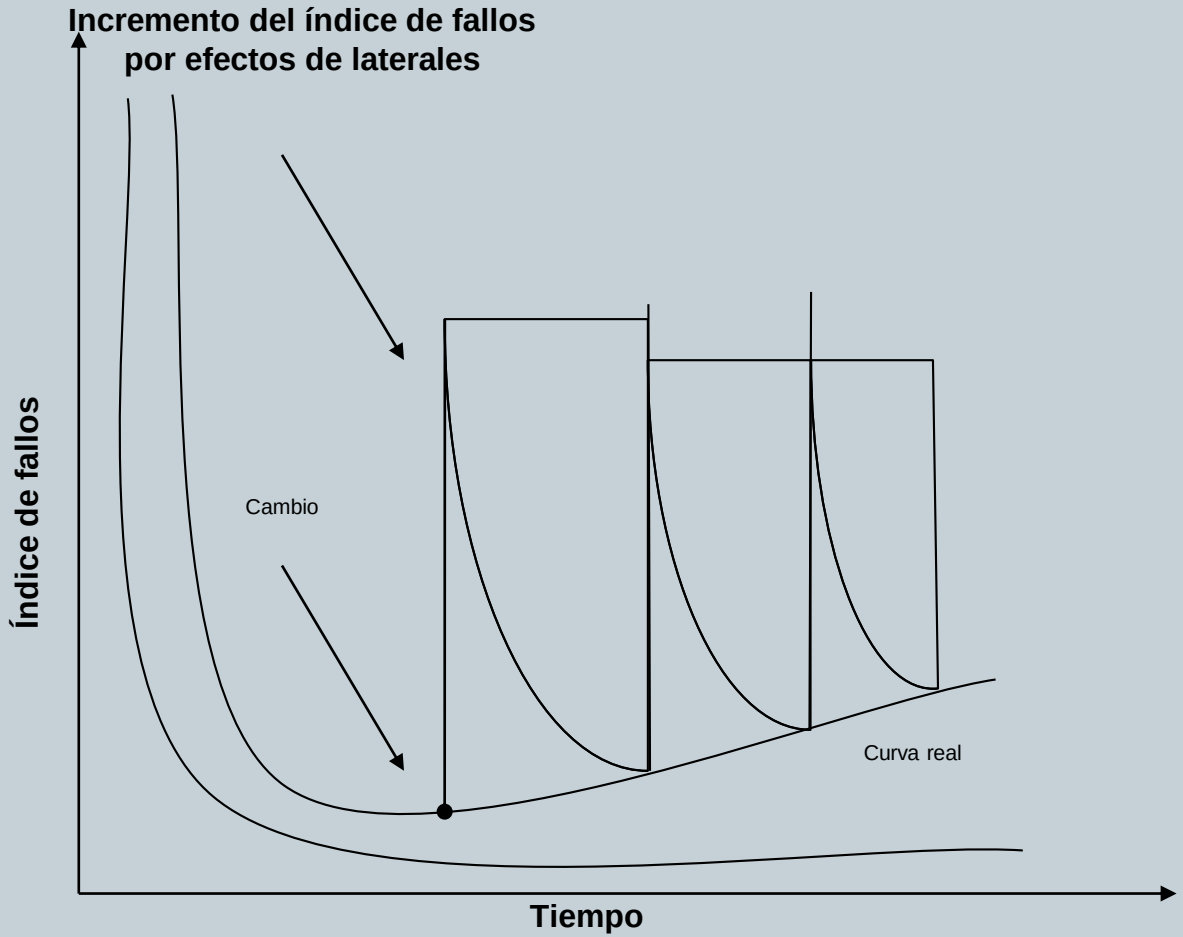
- **El software no se desgasta**



Curva de Bañera

Pressman Roger S. Ingeniería del software, Ed. Mc Graw Hill, 5ª ed

Curva de fallos para el Software



Pressman Roger S. Ingeniería del software, Ed. Mc Graw Hill, 6ª ed

2.3. CARACTERÍSTICAS DEL SOFTWARE

- **Aunque la industria tiene una tendencia hacia la construcción por componentes, la mayoría del software aún se construye a la medida.**
  - *En el mundo del hardware,* ***la reutilización*** *de componentes es una parte natural del proceso de ingeniería. En el mundo del software* ***es el inicio****.*

2.4. MITOS DEL SOFTWARE

❑ *En la actualidad, la mayoría de los profesionales reconocidos en la ingeniería del software identifican los mitos en su real dimensión: actitudes equivocadas que han causado problemas serios a los administradores y al personal técnico por igual. Sin embargo, las antiguas actitudes y viejos hábitos son difíciles de modificar, por lo que aún subsisten creencias falsas sobre el software.*

- ***Mitos de los administradores***
- ***Mitos de los Clientes***
- ***Mitos de los Desarrolladores***

Mitos de los administradores

Mito 1. *Ya se tiene un libro lleno de estándares y procedimientos para la construcción de software. ¿Esto proporcionará a mi gente todo el conocimiento necesario?*

Mito 2. *Si se está atrasado en el itinerario es posible contratar más programadores para así terminar a tiempo.*

Mito 3. *Si decido subcontratar el proyecto de software a un tercero, puedo relajarme y dejar que esa compañía lo construya.*

Mitos de los Clientes

Mito 1. *Un enunciado general de los objetivos es suficiente para comenzar a escribir programas; los detalles se pueden afinar después.*

Mito 2. *Los requerimientos del proyecto cambian de manera continua, pero el cambio puede ajustarse con facilidad porque el software es flexible.*

Mitos de los Desarrolladores

Mito 1. *Una vez que el programa ha sido escrito y puesto a funcionar, el trabajo está terminado.*

Mito 2. *Mientras el programa no se esté ejecutando, no existe forma de evaluar su calidad.*

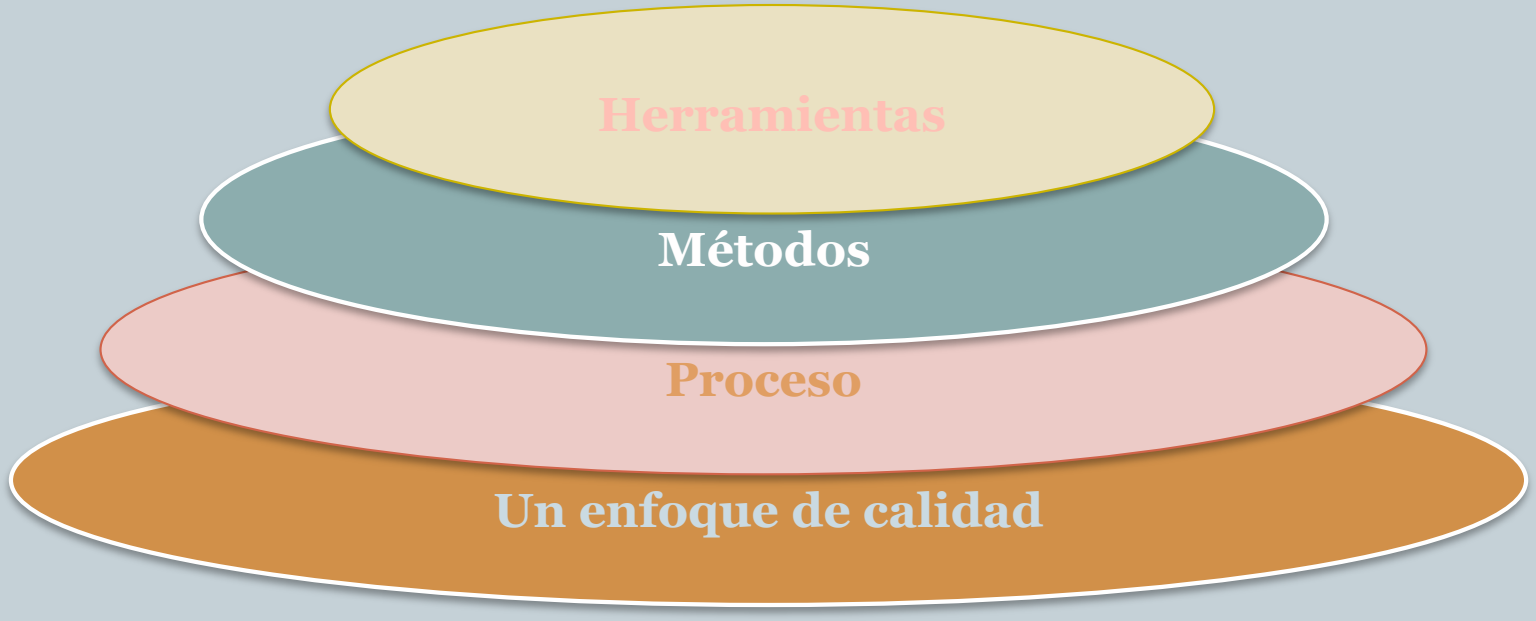
Mito 3. *El único producto del trabajo que puede entregarse para tener un proyecto exitoso es el programa en funcionamiento.*

Mito 4. *La Ing de Sw obligará a emprender la creación de una documentación voluminosa e innecesaria y de manera invariable tornará más lento el proceso.*

2.5. CAPAS DEL SOFTWARE

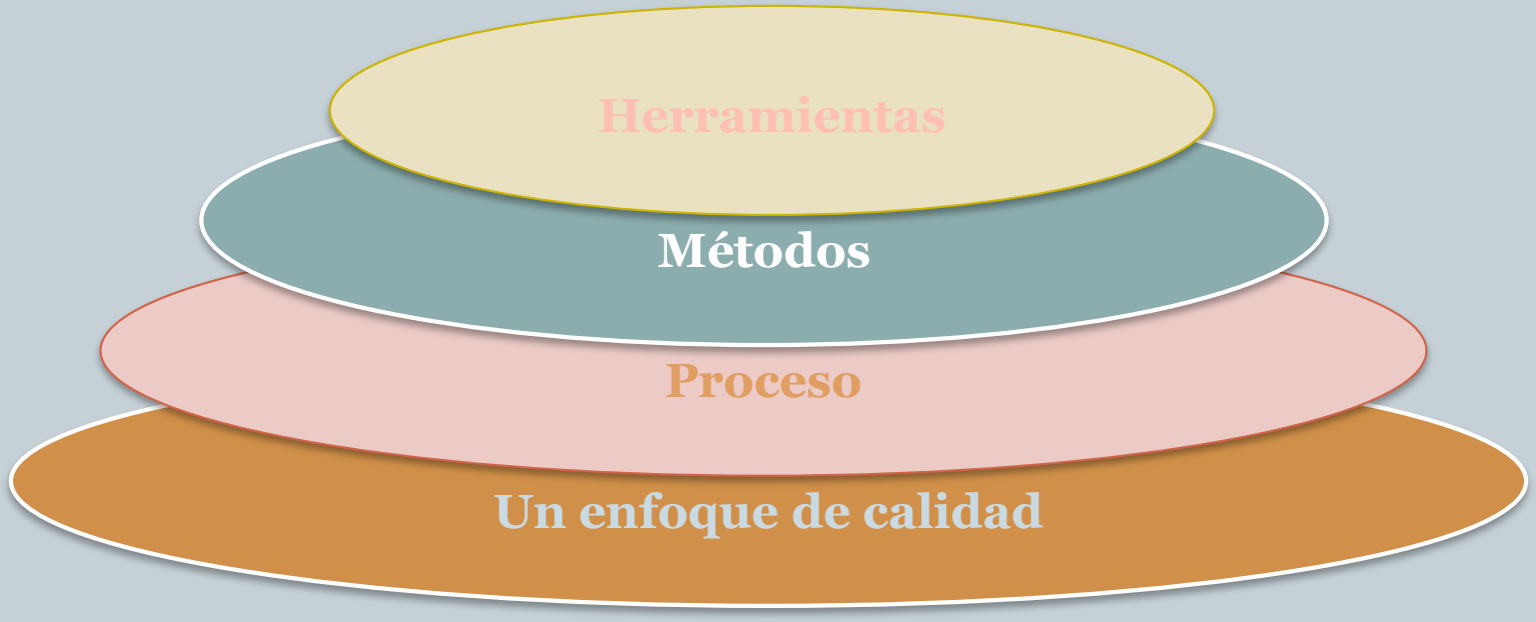
La Ingeniería del Software es una tecnología estratificada, y debe estar sustentada en un compromiso con la calidad.

Capa del Proceso



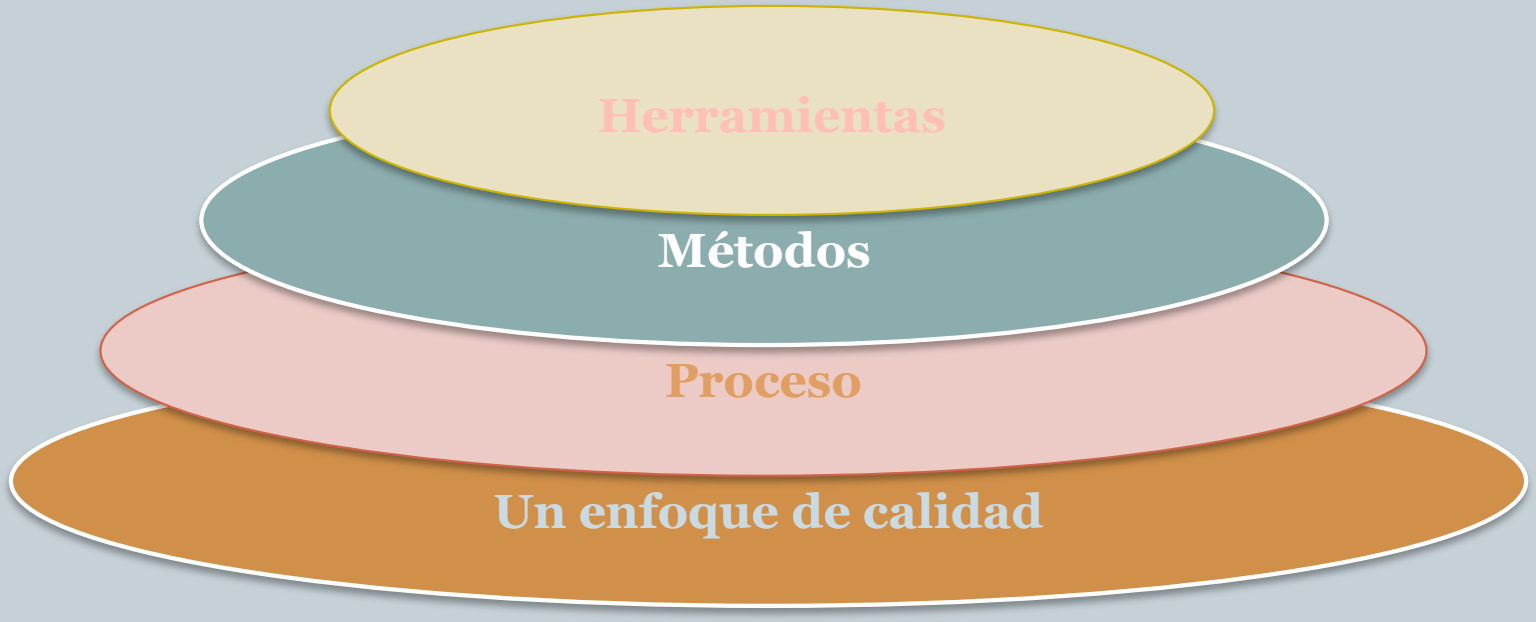
Las áreas claves del Proceso forman la base del control de gestión de proyectos del software y establecen contexto en el que se aplican los métodos técnicos, se obtienen productos del trabajo (modelos, documentos, datos, informes, formularios, etc.), se establecen hitos, se segura la calidad y el cambio se gestiona adecuadamente.

Capa de los Métodos



Los métodos de la Ingeniería del Software indican “como” construir técnicamente el software. Los métodos abarcan una gran gama de tareas que incluyen análisis de requisitos, diseño, construcción de programas, pruebas y mantenimiento.

Capa de Herramientas



Las herramientas de la Ingeniería de Software proporcionan un enfoque automático o semi-automático para el proceso y para los métodos. Cuando se integran herramientas para que la información creada por una herramienta la pueda utilizar otra, se establece un sistema de soporte para el desarrollo del software llamado *Ingeniería del Software Asistida por Computadora (CASE)*.

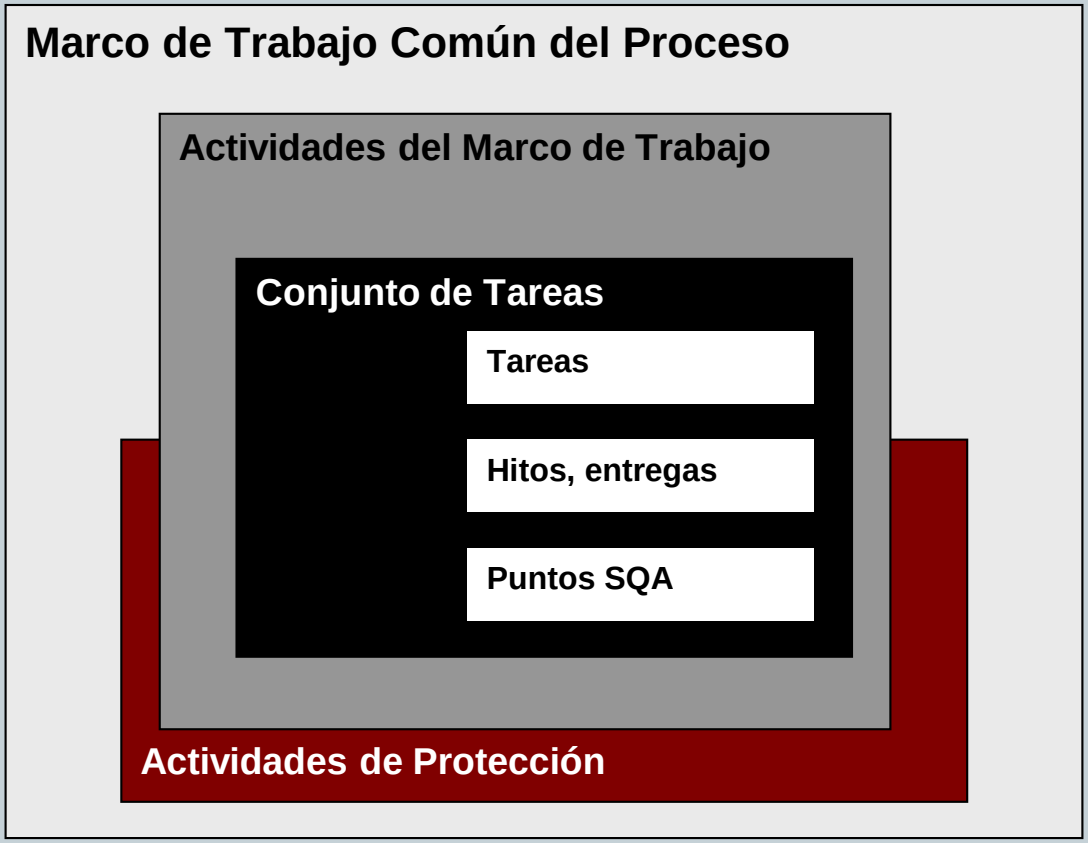
2.6. EL PROCESO DEL SOFTWARE

El proceso de desarrollo de software

2.6. EL PROCESO DEL SOFTWARE

- Un proceso de software establece un *marco común del proceso* definiendo un pequeño numero de actividades del marco de trabajo que son aplicables a todos los proyectos del software, con independencia de su tamaño o complejidad.

Marco común del Proceso de Software



Pressman Roger S. Ingeniería del software, Ed. Mc Graw Hill, 6ª ed

"Un proceso define quién está haciendo qué, cuando y cómo lograr cierta meta"

Ivar Jacobson, Grady Booch y James Rumbaugh

2.7. SOFTWARE DE ALTA CALIDAD

¿Qué es calidad?

- El grado en que un sistema, componente, o proceso cumple con los requerimientos especificados, y las necesidades y/o expectativas del cliente o usuario.

Otras definiciones de CALIDAI

- **ISO 9000**: "*Calidad: grado en el que un conjunto de características inherentes cumple con los requisitos*"
- **Real Academia de la Lengua Española**: "*Propiedad o conjunto de propiedades inherentes a una cosa que permiten apreciarla como igual, mejor o peor que las restantes de su especie*"
- **Philip Crosby:** "Calidad es cumplimiento de requisitos"
- **Armand V. Feigenbaum:** "Satisfacción de las expectativas del cliente".
- **Genichi Taguchi:** "Calidad es la menor pérdida posible para la sociedad".
- **William Edwards Deming:** "Calidad es satisfacción del cliente".
- **Walter A. Shewhart:** "La calidad como resultado de la interacción de dos dimensiones: dimensión subjetiva (lo que el cliente quiere) y dimensión objetiva (lo que se ofrece).
- .

2.7. SOFTWARE DE ALTA CALIDAD

¿Qué es calidad de software?

Pressman (2002) se refiere a la calidad del software como

"La concordancia con los requisitos funcionales y de rendimiento explícitamente establecidos, con los estándares de desarrollo explícitamente documentados, y con las características implícitas que se espera de todo software desarrollado profesionalmente".

Las definiciones anteriores resaltan tres puntos importantes:

- Los ***requisitos*** del software son la base de las medidas de calidad. La falta de concordancia con los requisitos es una falta de calidad.
- Los ***estándares*** especificados definen un conjunto de criterios de desarrollo que guían la forma en que se aplica la ingeniería del software. Si no se siguen esos criterios, casi siempre habrá falta de calidad.
- Existe un conjunto de ***requisitos implícitos*** que a menudo no se mencionan. Si el software se ajusta a sus requisitos explícitos pero falla en alcanzar los requisitos implícitos, la calidad del software queda en entredicho.

Control de Calidad

El control de calidad es una serie de inspecciones, revisiones y pruebas utilizadas a lo largo del proceso del software para asegurar que cada producto cumpla con los requisitos que le han sido asignados.

Garantía de Calidad

La garantía de calidad consiste en la auditoria y las funciones de información de la gestión. El objetivo de la garantía de calidad es proporcionar la gestión para informar de los datos necesarios sobre la calidad del producto, por lo que se va adquiriendo una visión más profunda y segura de que la calidad del producto esta cumpliendo sus objetivos.

Costos de Calidad

- El *costo de la calidad* incluye todos los costos acarreados en la búsqueda de la calidad o en búsqueda de las actividades relacionadas en la obtención de la calidad.

- El <u>costo de calidad</u> se puede dividir en:
  - Costos de prevención
  - Costos de evaluación
  - Costos de fallos.

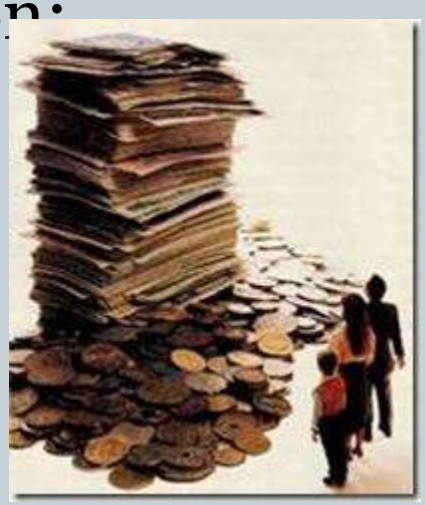
Costos de Prevención

- En el costo de prevención se incluye

  - Planificación de calidad,
  - Revisiones de técnicas formales,
  - Equipo de pruebas,
  - Formación.

Costos de Evaluación

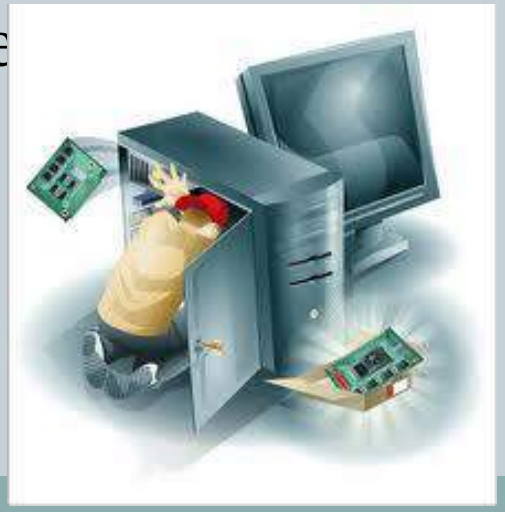
- En el costo de evaluación se incluyen actividades para tener una visión mas profunda de la condición del producto la primera vez a través de cada proceso.
- Algunos ejemplos de costos de evaluación:
  - Inspección en el proceso y entre procesos,
  - Calibrado y mantenimiento del equipo,
  - Pruebas.



Costos de Fallo

- Los costos de fallo son los costos que desaparecerían si no surgieran defectos antes del envió de un producto a los clientes.
- Estos costos se pueden subdividir en *costos de fallo interno y costos de fallo externo*.

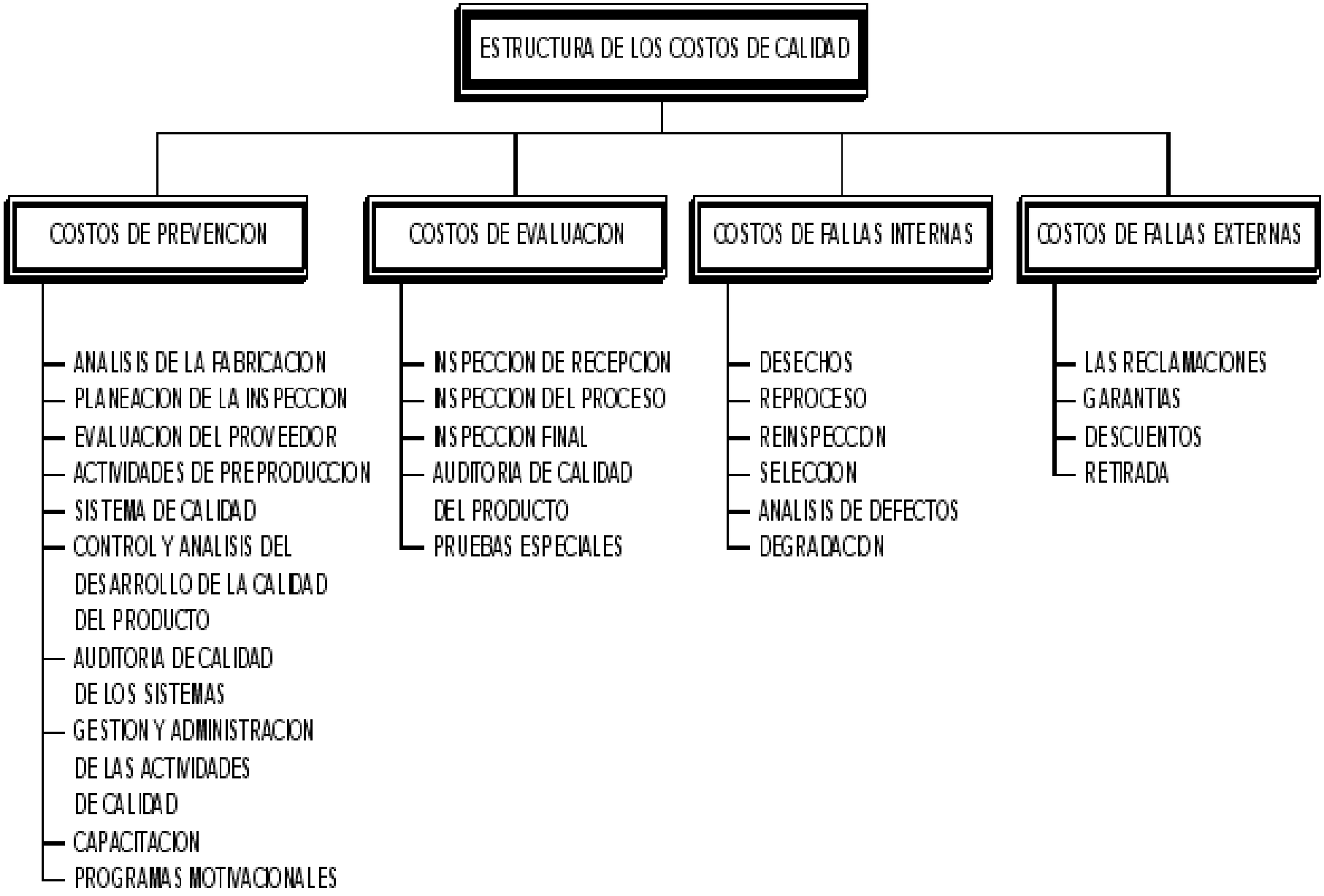
- ***Los internos*** se producen cuando se detecta un error en el producto antes de su envió. Entre e
  - Revisión,
  - Reparación,
  - Análisis de las modalidades de fallos.



Continuación…

- Los **costos de fallo externo** son los que se asocian a los defectos encontrados una vez enviado el producto al cliente.

- Algunos ejemplos de costos de fallo externo:
  - Resolución de quejas,
  - Devolución y sustitución de productos,
  - Soporte de línea de ayuda,
  - Trabajo de garantía.


ESTRUCTURA DE LOS COSTOS DE CALIDAD
COSTOS DE PREVENCION
COSTOS DE EVALUACION
COSTOS DE FALLAS INTERNAS
COSTOS DE FALLAS EXTERNAS
ANALISIS DE LA FABRICACION
PLANEACION DE LA INSPECCION
EVALUACION DEL PROVEEDOR
ACTIVIDADES DE PREPRODUCCION
SISTEMA DE CALIDAD
CONTROL Y ANALISIS DEL DESARROLLO DE LA CALIDAD DEL PRODUCTO
AUDITORIA DE CALIDAD DE LOS SISTEMAS
GESTION Y ADMINISTRACION DE LAS ACTIVIDADES DE CALIDAD
CAPACITACION
PROGRAMAS MOTIVACIONALES
INSPECCION DE RECEPCION
INSPECCION DEL PROCESO
INSPECCION FINAL
AUDITORIA DE CALIDAD DEL PRODUCTO
PRUEBAS ESPECIALES
DESECHOS
REPROCESO
REINSPECCION
SELECCION
ANALISIS DE DEFECTOS
DEGRADACION
LAS RECLAMACIONES
GARANTIAS
DESCUENTOS
RETIRADA

2.8. FACTORES DE CALIDAD Y PRODUCTIVIDAD (McCall)

- Los factores que afectan la calidad del software se pueden categorizar en dos amplios grupos:
  - *Factores que se pueden medir directamente (por ejemplo, defectos por punto de función) y*
  - *Factores que se pueden medir solo indirectamente (por ejemplo, facilidad de uso o de mantenimiento).*

2.8. FACTORES DE CALIDAD Y PRODUCTIVIDAD (McCall)

Facilidad de Mantenimiento

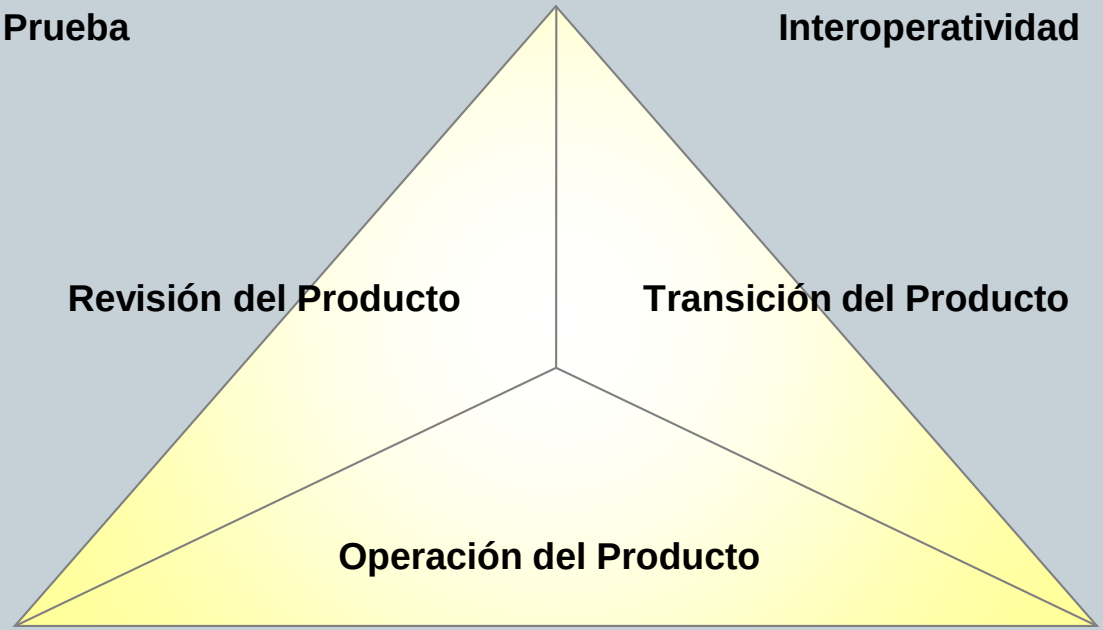
Flexibilidad

Facilidad de Prueba

Portabilidad

Reusabilidad

Interoperatividad



Corrección **Fiabilidad** **Usabilidad** **Integridad** **Eficiencia**

Factores de calidad de McCall

- **Corrección.** *Hasta dónde satisface un programa su especificación y logra los objetivos propuestos por el cliente.* ¿Hace lo que quiero?. Dicho de otra forma, es la capacidad de los productos de software para realizar con exactitud sus tareas, tal y como se definen en las especificaciones.

- La corrección es la cualidad principal. Si un sistema no hace lo que se supone que debe hacer, poco importan el resto de consideraciones que hagamos sobre él si es rápido, si tiene una bonita interfaz de usuario, etc.

Factores de calidad de McCall

- **Fiabilidad**. *Hasta donde se puede esperar que un programa lleve a cabo su función con la exactitud requerida*. ¿Lo hace de forma fiable todo el tiempo?.
- **Eficiencia**. *Es la capacidad de un sistema de software para exigir la menor cantidad posible de recursos hardware, tales como tiempo del procesador, espacio ocupando memoria interna y externa o ancho de banda utilizado en los dispositivos de comunicación*. ¿Se ejecutara en mi hardware lo mejor que pueda?

Factores de calidad de McCall

- **Integridad**. *Es la capacidad de los sistemas software de proteger sus diversos componentes (programas, datos, etc.) contra modificaciones y accesos no autorizados*. Hasta donde se puede controlar el acceso al software o a los datos por personas no autorizadas. ¿Es Seguro?
- **Usabilidad** *(facilidad de manejo). Es la facilidad con la cual personas con diferentes formaciones y aptitudes pueden aprender a usar los productos de software y aplicarlos a la resolución de problemas*. También cubre la facilidad de instalación, de operación y de supervisión.

Factores de calidad de McCall

- **Facilidad de mantenimiento**. *El esfuerzo necesario para localizar y arreglar un error en un programa. ¿Puedo corregirlo.*
- **Flexibilidad**. *El esfuerzo es necesario para modificar un programa que ya esta en funcionamiento*. ¿Puedo cambiarlo?.
- **Facilidad de prueba**. *El esfuerzo necesario para probar un programa y asegurarse de que realiza correctamente su función*. ¿Puedo probarlo?.

Factores de calidad de McCall

- **Portabilidad**. *Es la facilidad de transferir los productos software a diferentes entornos hardware y software.*
- **Reusabilidad** *(capacidad de reutilización). Es la capacidad de los elementos de software de servir para la construcción de muchas aplicaciones diferentes.*
- **Interoperatividad**. *El esfuerzo necesario para acoplar un sistema con otro. Interoperatividad es la facilidad de combinar unos elementos de software con otros*. ¿Podré hacerlo interactuar con otro sistema?.