A NIVEL PDF ENFOCARSE EN: TÉCNICAS DE ELICITACIÓN Y ESPECIFICACIÓN DE REQUERIMIENTOS (DEF, VENTAJAS, DESVENTAJAS, COMPONENTES, OBJETIVOS) // MODELOS DE DESARROLLO DE SOFTWARE (DEF, VENTAJAS, DESVENTAJAS) // REQUERIMIENTOS (FUNCIONALES, NO FUNCIONALES, EJEMPLOS SEGÚN UN PROBLEMA) // CALIDAD DE SOFTWARE Y DE PRODUCTO (DEF, MODELOS DE CALIDAD) // STAKEHOLDERS (DEF, IDENTIFICAR SEGÚN CASO, PUNTOS DE VISTA) // METODOLOGÍAS AGILES Y NO AGILES (DEF, DIFERENCIAS)

SOFTWARE

CLASE 1

Instrucciones, procedimientos, reglas, documentación y datos asociados que forman parte de las operaciones de un sistema de computación.

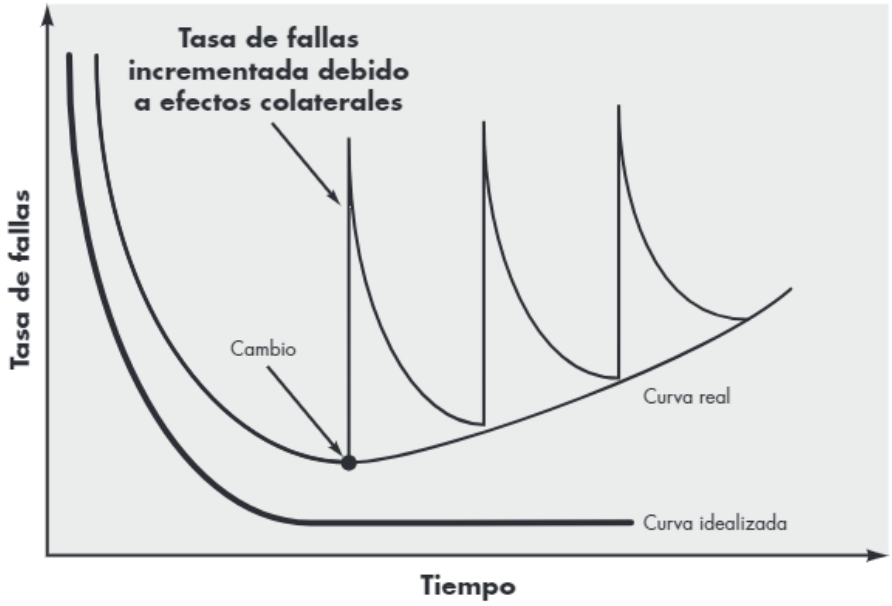
Características:

-Es un elemento lógico

-Se desarrolla, (no fabrica)

-No se desgasta

No sigue una curva de envejecimiento normal debido a que el problema esta en los cambios y no en el tiempo de operación.



Tipo de producto de software:

Genérico: Sistemas aislados que se venden al mercado abierto

Personalizado: Sistemas para un cliente en particular

Software libre:

1. Libertad para ejecutar el programa en cualquier sitio, con cualquier propósito y para siempre.
2. Libertad para estudiarlo y adaptarlo a nuestras necesidades. Esto exige el acceso al código fuente.
3. Libertad de redistribución, de modo que se nos permita colaborar con vecinos y amigos.
4. Libertad para mejorar el programa y publicar las mejoras. También exige el código fuente.

INGENIERIA DE SOFTWARE

Disciplina de la ingeniería que comprende todos los aspectos de la producción de software desde las etapas iniciales de la especificación del sistema incluyendo la evolución de éste, luego que se comienza a ejecutar.

Estudio del software desde la especificación de requerimientos hasta el fin del mismo

Está definida por el IEEE como el estudio de técnicas relacionadas con el uso de métodos sistemáticos, disciplinados y cuantificables para el desarrollo, operación y mantenimiento de software

Utiliza métodos sistemáticos cuantificables dentro de tiempos y costos estimados para el desarrollo, operación y mantenimiento

Participantes:

Gerente ejecutivo (dueño)

Gerente de proyecto (líder)

Profesionales especializados

Clientes

Usuarios finales

Stakeholders: todos aquellos que tienen una interacción con el sistema sea directa o indirectamente

INGENIERO DE SOFTWARE:

El Ingeniero debe dominar los aspectos técnicos, aprender habilidades requeridas para entender el problema, diseñar solución desarrollarla, tener un sentido de responsabilidad individual, aguda conciencia de las necesidades del equipo, atención al detalle, etc.

es el encargado del diseño de una solución para el problema que se le es planteado, necesita de conocimientos y técnicas específicas en el desarrollo de software

Responsabilidades:

Confidencialidad: Respetar la confidencialidad de sus empleados y clientes

Competencia: No falsificar el nivel de competencia y aceptar responsabilidades fuera de su capacidad

Derechos de la propiedad intelectual: Conocer las leyes vigentes sobre las patentes y copyright

Uso inapropiado de las computadoras: No debe utilizar sus habilidades técnicas para utilizar de forma inapropiada otras computadoras

TECNICAS DE COMUNICACIÓN

La comunicación es la base para obtener las necesidades del cliente y la principal fuente de errores

Fuentes de requerimientos:

-Documentación

De donde sacamos la info para obtener requerimientos

-Stakeholders

-Especificaciones de sistemas similares

Stakeholders:

Cualquier persona o grupo que se verá afectado por el sistema, directa o indirectamente como, por ejemplo:

-Usuarios finales

-Gerentes

-Ingenieros

-Expertos del dominio

Puntos de vista: IMPORTANTE !

Interactuadores: Personas u otros sistemas que interactúan directamente con el sistema. Pueden influir en los requerimientos del sistema.

Indirecto: Stakeholders que no utilizan el sistema ellos mismos pero que influyen en los requerimientos.

Dominio: Características y restricciones del dominio que influyen en los requerimientos del sistema.

ELICITACION DE REQUERIMIENTOS

Es el proceso de adquirir todo el conocimiento relevante necesario para producir un modelo de los requerimientos de un dominio de problema y es una actividad principalmente de carácter social, mucho más que tecnológico.

La elicitación proporciona una descripción verbal del sistema, una descripción visual puede consolidar la información.

Objetivos:

Conocer el dominio del problema para poder comunicarse con clientes y usuarios y entender sus necesidades.

Conocer el sistema actual.

Identificar las necesidades, tanto explícitas como implícitas, de clientes y usuarios y sus expectativas sobre el sistema a desarrollar.

Problemas:

Problemas de comunicación (falta o inadecuada comunicación)

Limitaciones cognitivas del desarrollador (falta de conocimiento sobre el tema)

Conducta humana (conflictos internos)

Problemas técnicos (como problemas muy complejos)

TECNICAS DE ELICITACION DISCRETAS

Métodos discretos: Muestreo de datos, documentos y formularios, observaciones de ambiente de trabajo e investigaciones y visitas al sitio. Estos son insuficientes para recopilar información por sí solos, por lo que deben utilizarse junto con uno o varios de los métodos.

Muestreo de la documentación, formularios y datos: Se hace en base a documentos existentes como bases de datos, sistemas en funcionamiento, diagramas o documentos de diseño, etc.

Investigaciones y visitas al sitio: Investiga el dominio, consulta otras organizaciones, etc.

Observación del ambiente de trabajo: El analista observa a los trabajadores y las actividades manteniendo bajo perfil, tomando nota de lo visto y con permiso del observado.

-Ventajas: Es confiable, económica y se puede ver lo que se hace exactamente

-Desventajas: Genera incomodidad, puede haber interrupciones y lo que se ve puede no ser como se hace habitualmente

TECNICAS DE ELICITACION INTERACTIVAS

Métodos interactivos: Entrevistas, cuestionarios, Brainstorming y planeación conjunta de requerimientos. Se basan en hablar con las personas en la organización y escuchar para comprender.

Planeación Conjunta de Requerimientos: Son reuniones altamente estructuradas para analizar problemas y definir requerimientos contando con alta participación de los integrantes y reduce el tiempo de exploración de requisitos a cambio de necesitar gran entrenamiento.

JRP

-Ventajas: Ahorra tiempo e involucra a los usuarios y a la gerencia activamente

-Desventajas: Es complicado de organizar y formar por el entrenamiento necesario

Brainstorming: Los participantes ofrecen ideas sin análisis hasta que no haya más ideas, buscando ideas creativas y en lo posible en gran cantidad. Ayuda a entender el dominio del problema y a resolver la falta de consenso entre usuarios.

Cuestionarios: Documento que permite al analista recabar información y opiniones de los encuestados en gran cantidad de un tema general principalmente a grandes o dispersos grupos obteniendo actitudes, creencias, comportamientos y características de los encuestados.

-Ventajas: Son rápidos, económicos, anónimos y fáciles de analizar

-Desventajas: Son rígidos, difíciles de hacer, pocas respuestas y no hace todas las preguntas

Tipos de preguntas:

-Abiertas: Abiertas a todas las opciones de respuesta

-Cerradas: Limitan o cierran todas las opciones de respuesta

Entrevistas: El analista recolecta información de las personas mediante una conversación con un propósito específico, basada en preguntas y respuestas en general.

-Ventajas: Hace al entrevistado sentirse en el proyecto, permite adaptar las preguntas al entrevistado y se puede conseguir una retroalimentación del entrevistado

-Desventajas: Son costosas, no aplicables a distancia y depende de la habilidad del entrevistador además de tomar tiempo y recursos humanos

Tipos de entrevistas:

Pueden usar todos los tipos de preguntas, nos centramos en el objetivo puntual de la entrevista si es que posee

-Estructuradas: Hay un conjunto de preguntas a hacer al entrevistado sobre algo puntual y no permite adquirir amplio conocimiento del dominio

-No estructuradas: Hay un tema general sin preguntas especificas listas e inicia con preguntas que no dependen del contexto

Tipos de preguntas en entrevistas:

-Abiertas: Preguntas a responder como quiera el entrevistado

-Ventajas: Permiten espontaneidad y revelan más preguntas

-Desventajas: Da datos irrelevantes y se puede perder el control de la entrevista

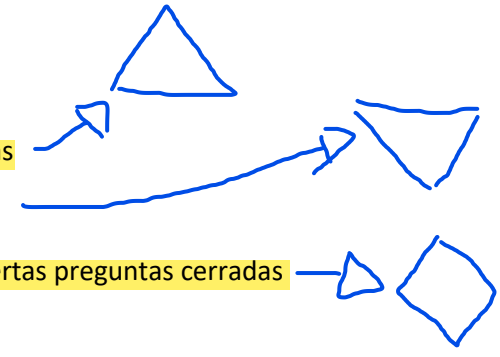
-Cerradas: Preguntas directas o cortas

-Ventajas: Ahorran tiempo, son fáciles de controlar y solo da datos relevantes

-Desventajas: No se obtienen detalles

-Sondeo: Preguntas enfocadas en un tema puntual

Organizaciones:



-Piramidal: preguntas cerradas -> preguntas abiertas

-Embudo preguntas abiertas -> preguntas cerradas

-Diamante: preguntas cerradas -> preguntas -> abiertas preguntas cerradas

Preparación:

-Leer los antecedentes

-Establecer los objetivos de la entrevista

-Seleccionar los entrevistados (minimiza la cantidad)

PASOS A SEGUIR PARA ARMAR UNA ENTREVISTA

-Planificación de la entrevista y preparación del entrevistado

-Selección del tipo de preguntas a usar y su estructura

CLASE 2

REQUERIMIENTOS

Es una característica o una descripción de algo que el sistema *o un componente del sistema* es capaz de hacer para cumplir con el propósito del sistema.

Errores:

Los errores de requerimientos pueden provocar la construcción de un sistema erróneo, un software que no satisface al usuario o desacuerdos entre clientes y desarrolladores.

Tipos de requerimientos:

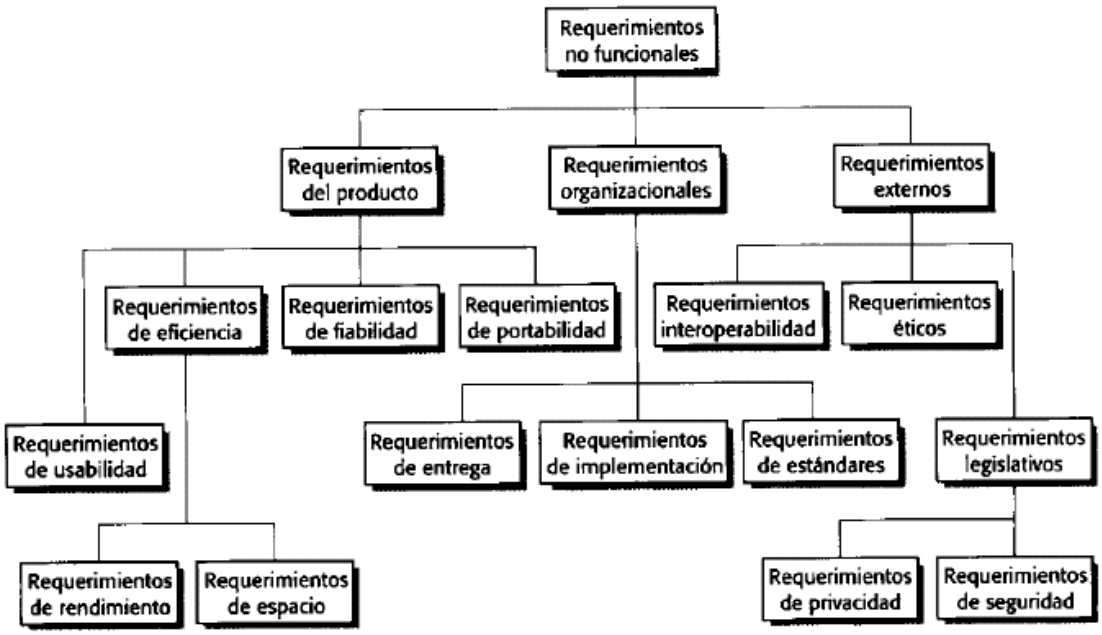
Funcionales: Describen que hace y como NO se comporta el sistema, las funcionalidades del mismo y son independientes de la implementación de la solución

No funcionales: Restricciones que limitan las elecciones de construcción de soluciones al problema

-Producto: Especifica el comportamiento del producto

-Organizacionales: Se derivan de las políticas y procedimientos existentes en la ____

-Externos: Son de tipo de Interoperabilidad, legales, privacidad, seguridad, éticos, etc

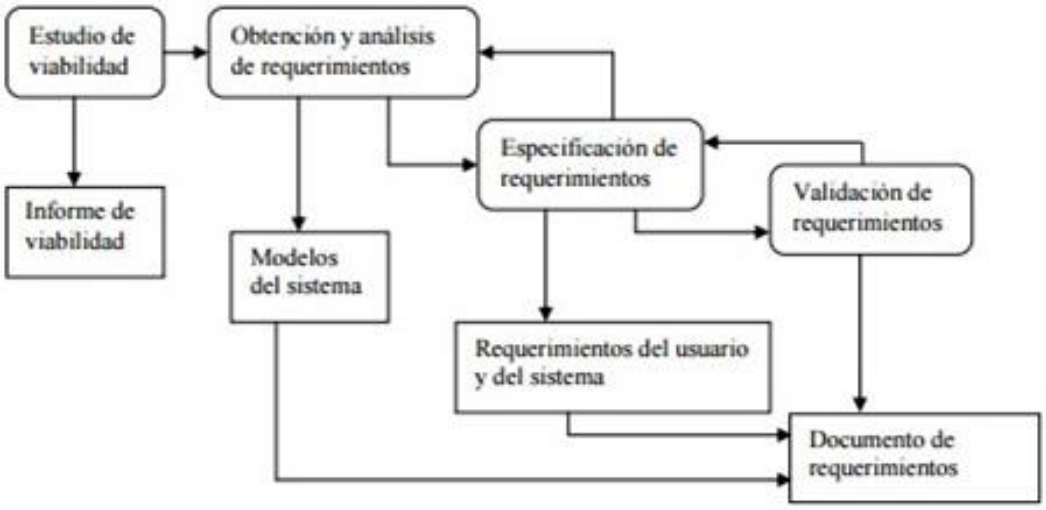


INGENIERIA DE REQUERIMIENTOS

Proceso por el cual se transforman los requerimientos declarados por los clientes a especificaciones precisas, no ambiguas, consistentes y completas del comportamiento del sistema, incluyendo funciones, interfaces, rendimiento y limitaciones.

Importancia:

Mejora la calidad de software, la comunicación con el equipo y la capacidad de predecir cronogramas de proyectos y gestionar las necesidades del proyecto estructuradamente



Estudio de viabilidad:

Elabora un informe que recomienda o no llevar a cabo el proyecto a partir de una descripción del proyecto

ESPECIFICACION DE REQUERIMIENTOS

El objetivo es que los desarrolladores expliquen al cliente como ven el sistema, las caracterizas y funcionalidades del sistema y que demostraciones llevar a cabo para que el cliente vea que el sistema resultante es lo que pidió

Como los desarrolladores ven el sistema y le demuestran al cliente que es lo que pidió

Propiedades:

-Necesario: Si no está provoca deficiencias

-Conciso: Fácil de leer y entender

-Completo: No necesita ampliarse

-Consistente: No contradice otros

-No ambiguo: De única implementación

-Verificable: Es testeable

Aspectos básicos:

-Funcionalidad

-Interfaces externas

-Rendimiento

-Atributos

-Restricciones de diseño

VALIDACION DE REQUERIMIENTOS

Es el proceso de certificar la corrección del modelo de requerimientos contra las intenciones del usuario. Comprueba que los requerimientos son los estipulados por el sistema para asegurar que no haya errores a futuro ya que cuanto más se tarde en descubrir, más cuesta corregirlo (Bola de nieve de defectos)

Se consideran valida la solución especificada en el modelo de requerimientos cuando es correcto para el usuario porque no existen los requerimientos de los requerimientos

Tipos de validaciones:

-Validez (todos los usuarios)

-Consistencia (sin contradicciones)

-Completitud (todos los requerimientos)

-Realismo (se pueden implementar)

-Verificabilidad (se pueden diseñar conjuntos de pruebas)

Técnicas de validación:

Pueden ser manuales o automáticas

-Revisión de requerimientos formal: Se conduce al cliente explicando las implicaciones de cada requerimiento

- Revisión de requerimientos informal: Se tratan los requerimientos con la mayor cantidad de stakeholders posibles

-Construcción de prototipos

-Generación de casos de prueba

TECNICAS DE ESPECIFICACION DE REQUERIMIENTOS

Estaticas: Se describe el sistema a través de las entidades u objetos, sus atributos y sus relaciones con otros. No describe cómo las relaciones cambian con el tiempo

NOS ENFOCAMOS EN EL ESTADO ACTUAL DEL SISTEMA Y NO LOS CAMBIOS QUE PUEDE AFRONTAR EN EL FUTURO

LO CONTRARIO A LA ESTÁTICA

Dinamicas: Se considera un sistema en función de los cambios que ocurren a lo largo del tiempo que lo obligan a cambiar su estado actual

Define al sistema como:

-Condiciones satisfechas por el sistema en un momento dado

-Reglas para reaccionar ante los estímulos que ocurren cuando se reúnen determinados conjuntos de condiciones

-Acciones a ser tomadas como un resultado.

HISTORIAS DE USUARIO

Es una descripción corta y simple de un requerimiento de un sistema, que se escribe en lenguaje común del usuario y desde su perspectiva.

Conceptos:

-Debe poder escribirse en una nota pequeña (limitada)

-Administran rápidamente los requisitos sin mucho tiempo de administración y sin muchos documentos

-Responde rápidamente a los requisitos cambiantes

-Se pueden discutir con los clientes

-Se estima entre 10 horas y un par de semanas por historia

-Se pueden dividir en varias historias si son muy largas o complejas

Esquema: Como (rol) quiero (algo) para poder (beneficio).

Características:

-Independientes unas de las otras

-Negociables: No son contratos y permite la discusión para esclarecer su alcance explícitamente bajo pruebas de validación

-Valoradas para clientes o usuarios: Deben ser más importantes para el cliente o usuario más que para el desarrollador

-Estimables: Se puede estimar el tiempo que tomará completarla y esto permite estimar el tiempo total del proyecto

-Pequeñas: Se recomienda la consolidación de historias muy cortas en una sola historia y la separación de historias muy grandes en varias historias

-Verificables: Son verificables porque cubren requerimientos funcionales y la verificación debería automatizarse para que sea verificada en cada entrega de proyecto

Criterios de aceptación:

Son los criterios por los cuales se define si una historia de usuario fue desarrollada según la expectativa del Product Manager/Owner y si se puede dar como hecha

Se definen en la etapa inicial para entender mejor cómo se comporta el producto y complementar las historias de usuario

El cliente juega una parte fundamental en estas y es el que debería escribir mas

No son tan detallados como otras especificaciones y una gran cantidad de estos es un indiciador de hora de subdividir la historia (¡¡¡a partir de 4, que es mentira porque la tarjeta tiene más!!!)

Beneficios:

Son cortas, fáciles de mantener, invitan al cliente a participar y son ideales para proyectos con requisitos volátiles o para dividir los proyectos en pequeñas partes

Limitaciones:

Requiere constante contacto con el cliente, son difíciles de escalar, requiere desarrolladores competentes y si no tienen criterios de aceptación quedan muy abiertas a interpretación

EPICA

Conjunto de Historias de usuario que se agrupan por algún denominador común y la épica no es la funcionalidad

Características:

-Suelen abarcar varios equipos de desarrollo

-Son grupos de Historias de usuario

-Los clientes determinan las historias de cada épica y si se eliminan o agregan

-Permite estructurar los temas e iniciativas (objetivos)

-Dan flexibilidad y agilidad al proyecto

CLASE 3

CASOS DE USO

Modelado de las funcionalidades del sistema en eventos entre los usuarios y el sistema.
Facilitando y alentando la participación de los usuarios

Beneficios:

Captura requerimientos funcionales, permite comunicarse con el usuario, define líneas base para los planes de prueba y documentación del sistema y descompone el alcance del mismo en piezas manejables

Componentes:

Diagrama: Ilustra las interacciones entre el sistema y los actores

Escenarios: Interacciones entre el actor y el sistema para realizar la funcionalidad y eventos alternativos

Caso de uso: Representa una funcionalidad individual del sistema y describe la secuencia de actividades y de interacciones para alcanzarlo a través de su escenario.

Actor: Papel desempeñado por usuarios (personas, servidores, etc.) que interactúa con los casos de uso en el sistema iniciándolos

Relaciones:

-Asociaciones: Relación entre un actor y un CU en el que interactúan entre sí.

-Extensiones: Un CU extiende la funcionalidad de otro CU o varios y solo son iniciados por otro CU

-Uso o inclusión: Combina los pasos comunes de los CU cuando son 2 o más para reducir la redundancia

-Dependencia: Indica que un CU no puede realizarse hasta que se haya realizado otro CU.

-Herencia: Relación entre actores donde un actor hereda las funcionalidades de uno o varios actores.

Proceso de modelado:

Identificar a los actores en documentación, minutas o documentos de requerimientos

Identificar los CU para los requerimientos

Construir el diagrama

Realizar los escenarios

Características:

-Representan funcionalidades concretas

-Hay más de un paso por escenario

-Los condicionales en el curso normal son para excepciones

-Las precondiciones no son casos alternos, siempre están cumplidas

-Uses es para mínimo de 2, para 1 es extends

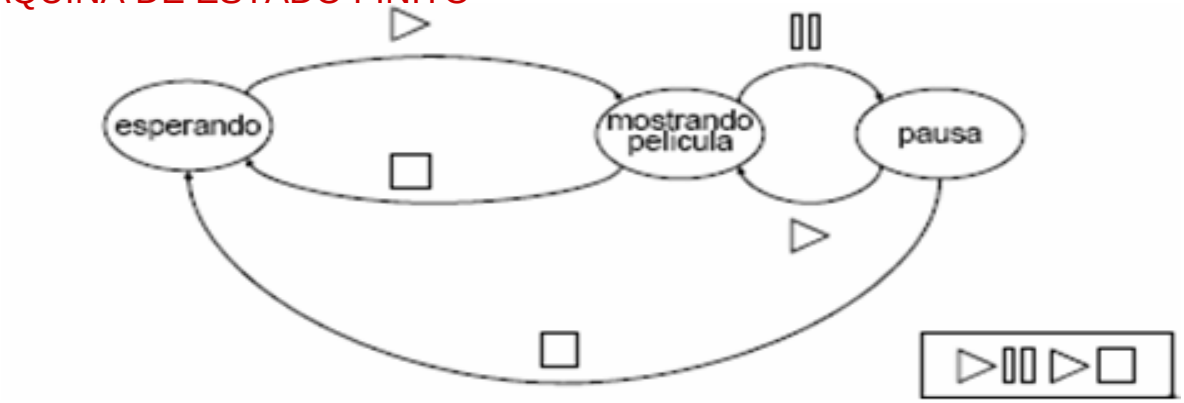
-El curso normal es la ejecución del CU sin alteraciones

CLASE 4

DIAGRAMA DE TRANSICION DE ESTADOS

~~Máquinas de estado infinito:~~ Conjunto de estados donde el sistema reacciona a ciertos eventos posibles (externos o internos)

MÁQUINA DE ESTADO FINITO



Máquinas de estado infinito: puede ser descrito como una 5-tupla (S,Σ,T,s,A) donde:

-S: Conjunto de estados

-Σ: Alfabeto

-T: Funcion de transicion

-s: Estado inicial

-A: Conjunto de estados de aceptación o finales

Pasos de construcción de DTE:

-Identificar los estados

-Si hay un estado complejo se puede explotar (Identificar estados complejos)

-Desde el estado inicial, se identifican los cambios de estado con flechas

-Se analizan las condiciones y las acciones para pasar de un estado a otro

-Se verifica la consistencia:

Se han definido todos los estados

Se pueden alcanzar todos los estados

Se pueden salir de todos los estados

En cada estado, el sistema responde a todas las condiciones posibles (normales y anormales)

REDES DE PETRI

CLASE 5

Utilizadas para especificar sistemas de tiempo real en los que son necesarios representar aspectos de concurrencia ya que en estos ocurren varias tareas o procesos sincronizados para permitir la comunicación entre ellos y pueden ocurrir en paralelo, en un orden impredecible, en un solo procesador.

Componentes:

Los eventos son transiciones (T)

Los estados son lugares/sitios (P)

Los arcos indica la relación entre sitios y transiciones y viceversa (son una flecha).

El lugar tiene tokens que son un número o puntos dentro del sitio. Esta asignación de tokens a lugares constituye la marcación (cuántos tokens hay en cada lugar al inicio).

Tras la marcación inicial se puede simular la ejecución de la red. El número de tokens asignados a un sitio es ilimitado.

Definición: Es una 4-tupla C = (P, T, I, O) donde

-P: Lugares

-T: Transiciones

-I: Función de entrada

-O: Función de salida

Es un multígrafo (1 o más arcos por nodo), bipartito, dirigido

Tokens:

-Los tokens asociados a cada estado sirven para manejar la coordinación de eventos y estados.

-Tras un evento, los tokens pueden viajar a otro estado

-Las reglas de disparo hacen que los token viajen cuando se cumplen las condiciones

-La ejecución es controlada por el número y distribución de los tokens.

-La ejecución de una Red de Petri se realiza disparando transiciones habilitadas.

-Una transición está habilitada cuando cada lugar de entrada tiene al menos tantos tokens como arcos hacia la transición.

-Disparar una transición habilitada remueve los tokens de los lugares de entrada y distribuye los tokens en los lugares de salida

Transiciones:

-La ocurrencia de las transiciones depende del estado del sistema.

-Una condición puede ser V (con token) o F (sin token)

-La ocurrencia de un evento está sujeta a que se den ciertas condiciones (pre) y al ocurrir el evento causa que se hagan verdaderas las post-condiciones.

-Las RP son asincrónicas y el orden en que ocurren los eventos es uno de los permitidos

-La ejecución es no determinística

-El disparo de una transición es instantáneo

Sincronización: Comparten información y recursos, controladamente para asegurar la integridad y correcta operación del sistema

Características:

Se puede estudiar y analizar estos sistemas de eventos discretos mediante herramientas analísticas estadísticas para comprender su comportamiento

Las RP pueden ser aplicadas para la modelación de sistemas de eventos discretos, las cuales ofrecen una forma de representación gráfica y matemática de los sistemas modelados.

La formalidad matemática de la RP proporcionan herramientas de análisis para analizar los posibles estados a los que el sistema modelado pudiera alcanzar.

TABLAS DE DECISION

CLASE 6

Es una herramienta que representa las reglas lógicas para decidir acciones a ejecutar en función de las condiciones y la lógica de decisión de un problema específico.

Componentes: condiciones (V o F) y acciones simples y 2^N Reglas donde N es la cantidad de condiciones

Pasos para construir:

-Identificar las condiciones y las acciones.

-Completar la tabla teniendo en cuenta:

Si hay condiciones que son opuestas, debe colocarse una de ellas porque por la negativa se “obtendrá” la otra. (Si son n condiciones excluyentes, colocar n-1 en la tabla).

Las condiciones deben ser atómicas.

-Se construyen las reglas

Tipos de especificación:

-Completas: Determinan acciones (1 o más) para todas las reglas posibles.

-Redundantes: Reglas que especifican las mismas acciones para las mismas condiciones que otras reglas ya presentes en la tabla

-Contradictorias: Reglas que especifican acciones diferentes para las mismas condiciones.

Redundancia y Contradicción

	Reglas						
C1	V	V	. .	..	..	F	F
C2	V	V	.	.	.	V	V
C3	V	F	.	.	..	F	F
A1			.	.	..	X	X
A2	X				.		
A3		X	.	.		X	X

Redundante

	Reglas						
C1	V	V	. .	. .	..	F	F
C2	V	V	.	.	.	V	V
C3	V	F	.	.	..	F	F
A1			.	.	..		X
A2	X				.	X	
A3		X	.	.		X	

Contradictoria

Si tienes 2 reglas en donde sea evidente que una alternativa no representa una diferencia en el resultado usa el " - "

Condición 1:	S	S
Condición 2	S	N
Acción 1	X	X

Condición 1:	S
Condición 2	_
Acción 1	X

Verificar la integridad y precisión de las tablas de decisión (SIMPLIFICA). Pueden ocurrir cuatro problemas principales al desarrollar tablas de decisión: que estén incompletas, que existan situaciones imposibles, contradicciones y redundancia.

ANALISIS ESTRUCTURADO

Permite una representación gráfica para lograr una comprensión más profunda del sistema a construir y comunicar a los usuarios lo comprendido. Sin centrarse en los aspectos físicos y enfocándose en el procesamiento de los datos en los procesos.

Busca reconocer como se mueven los datos, sus procesos o transformaciones y sus resultados

DIAGRAMA DE FLUJO DE DATOS

Otro nombre es diagrama de burbujas

Permite visualizar un sistema como una red de procesos funcionales, conectados entre sí por conductos y almacenamientos de datos. Representa la transformación de entradas a salidas. Es utilizado por sistemas operacionales que las funciones del sistema son de gran importancia

Componentes:

Rectángulo: Representa un elemento de un sistema que produce información para ser transformada por el software o recibe información producida por el software

Burbuja: Representa un proceso o transformación aplicado a los datos (o al control) y los modifica.

Flecha: Representa uno o más elementos de datos

Rectángulo abierto: Representa un almacén de datos

Pasos para construir:

-Redactar la lista de actividades de la organización para determinar:

-Entidades externas

-Flujos de datos

-Procesos

-Almacenes de datos

-Crear un diagrama de contexto que muestre las entidades externas y los flujos de datos desde y hacia el sistema.

-Dibujar el Diagrama 0 (siguiente nivel), con procesos generales y los almacenes correspondientes

-Dibujar un diagrama hijo por cada uno de los procesos del Diagrama 0

MODELOS DE PROCESO

CLASE 7

Proceso: Es un conjunto ordenado de tareas generalmente realizadas en el mismo orden en un servicio o producto

PROCESO DE SOFTWARE

Proceso de software: Es un conjunto de actividades y resultados asociados que producen un producto de software.

Actividades de proceso de software:

-Especificación de software: Proceso de comprender y definir qué servicios se requieren del sistema y la identificación de las restricciones sobre la operación y desarrollo del sistema (ingeniería de requerimientos) RECORDAR DEFINICION MÁS ARRIBA

-Desarrollo del software: Proceso de convertir una especificación del sistema en un sistema ejecutable. Incluye los procesos de diseño y programación. Se crea una descripción de la estructura del software que se va a implementar, los modelos y estructuras de datos, las interfaces, etc.

-Validación del software: Proceso de buscar que un sistema cumpla sus especificaciones y las expectativas del cliente y cuenta con inspecciones y revisiones en distintas etapas. Por ejemplo, se prueba con datos de prueba simulados.

-Evolución del software: El mantenimiento, los cambios y las mejoras son actividades a tener en cuenta en el proceso de desarrollo de software.

MODELOS DE PROCESO DE SOFTWARE

Representación simplificada de un proceso de software que presenta una visión de ese proceso incluyendo partes de los procesos y productos de software y el papel de las personas involucradas.

Características:

-Establece todas las actividades.

-Utiliza recursos, está sujeto a restricciones y genera productos intermedios y finales.

-Puede estar compuesto por subprocesos.

-Cada actividad tiene entradas y salidas definidas.

-Las actividades se organizan en una secuencia. → ES SECUENCIAL

-Existen principios que orientan sobre las metas de cada actividad.

-Las restricciones pueden aplicarse a una actividad, recurso o producto.

Modelos prescriptivos: Prescriben un conjunto de elementos del proceso: actividades del marco de trabajo, acciones de la ingeniería del software, tareas, aseguramiento de la calidad y mecanismos de control y están interrelacionados entre sí.

Modelos descriptivos: Describen como se realizan en la realidad

Modelos tradicionales: Conjunto de fases o actividades que no toma en cuenta la evolución del software

-Clásico, lineal o cascada

-Modelo V

-Basado en prototipos

Modelos evolutivos: Se adaptan a la evolución de los requisitos del sistema en el tiempo

-En espiral (mediante fases)

-Evolutivo (mediante fases)

-Incremental

MODELO EN CASCADA

Se representa en cascada, es secuencial (terminar el anterior para pasar al siguiente) y es simple

Dificultades:

-Necesita terminar para obtener resultados

-Las fallas graves ocurren al final lo que las hace difícil de manejar

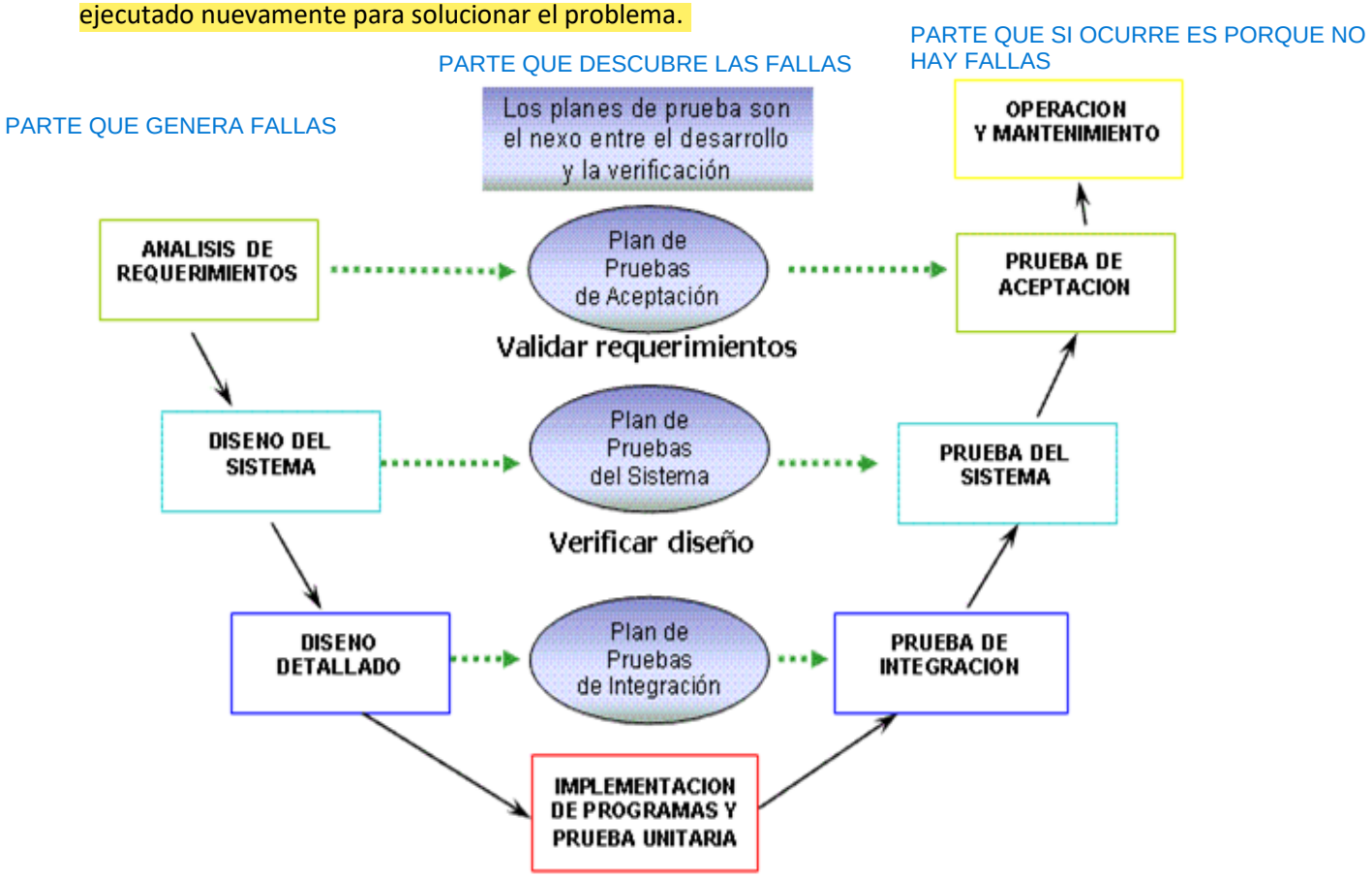
-Necesita más pruebas al final

-No se adapta bien a los cambios

MODELO EN V

Demuestra cómo se relacionan las actividades de prueba con las de análisis y diseño. Sugiere que la prueba unitaria y de integración también sea utilizada para verificar el diseño del programa.

Funcionamiento: La vinculación entre los lados derecho e izquierdo implica que, si se encuentran problemas durante la verificación y validación, entonces el lado izquierdo de la V puede ser ejecutado nuevamente para solucionar el problema.



MODELO DE PROTOTIPOS

Prototipo: Producto parcialmente desarrollado que permite que se examinen algunos aspectos del sistema propuesto, y decidan si éste es adecuado o correcto para el producto terminado.

Tipos:

-Evolutivo: Busca obtener el sistema final y permite construir rápidamente partes del o el sistema para analizar y tener una comprensión de lo que se necesita y lo que se propone como solución

-Descartable: Sin funcionalidad, se hace con herramientas de modelaje y se desecha

Proyectos candidatos para usar prototipos (cuando los usas):

-Usuarios que no examinarán los modelos abstractos

-Usuarios que no determinarán sus requerimientos inicialmente

-Sistemas con énfasis en los formatos de E/S más que en los detalles algorítmicos

-Sistemas en los que haya que explorar aspectos técnicos

-Si el usuario tiene dificultad al tratar con los modelos gráficos para modelar los requerimientos y el comportamiento

-Si se enfatiza el aspecto de la interfaz humana

Características recomendadas:

-El sistema ha de ser experimentable

-El costo ha de ser barato (<10%)

-Debe ser de desarrollo rápido

-Se enfoca en la interfaz de usuario

-Equipo de desarrollo reducido

-Herramientas y lenguajes adecuados

MODELO DE DESARROLLO POR FASES

El sistema se desarrolla por fases permitiendo que se entregue por partes (sistema operacional y sistema en desarrollo)

Tipos:

-Incremental: Se subdivide en subsistemas por funcionalidad, se entrega por subsistemas

-iterativo: Entrega un sistema completo y va aumentado sus funcionalidades de los subsistemas con nuevas versiones

MODELO EN ESPIRAL

Características:

-Combina las actividades de desarrollo con la gestión del riesgo

-Trata de mejorar los ciclos de vida clásicos y prototipos.

-Incorpora objetivos de calidad

-Elimina errores y alternativas no atractivas al comienzo

-Permite iteraciones, vuelta atrás y finalizaciones rápidas

-Cada ciclo identifica primero los objetivos de la porción correspondiente y sus alternativas

Restricciones: Cada ciclo se completa con una revisión que incluye todo el ciclo anterior y el plan para el siguiente

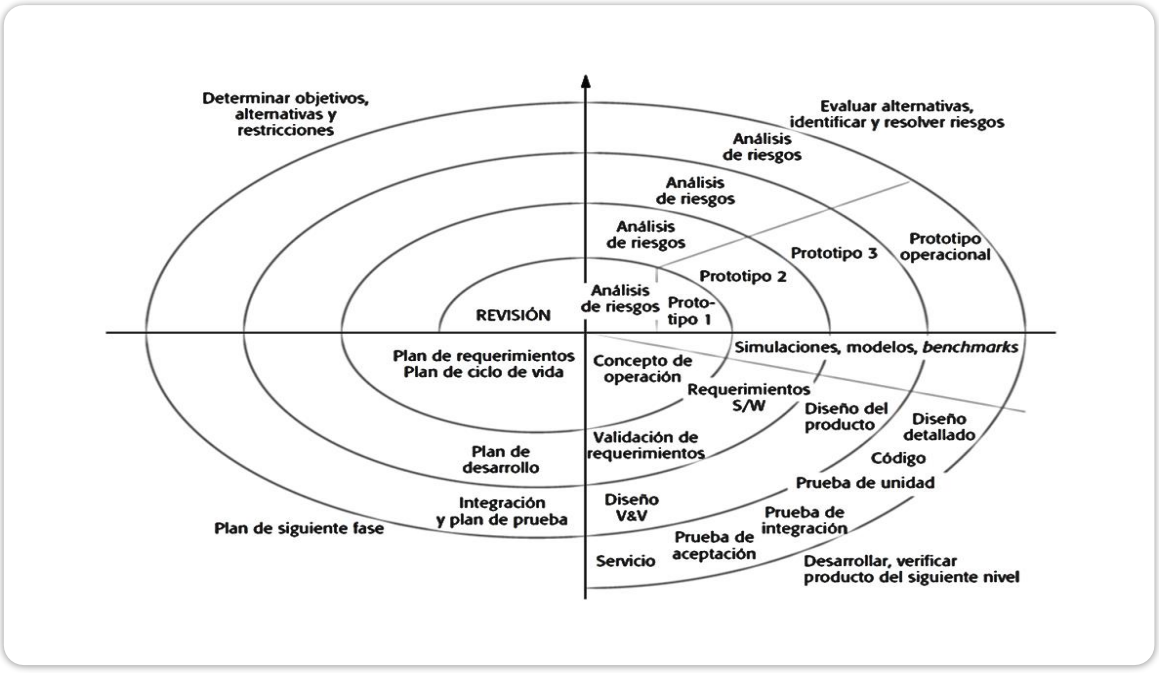
Sectores:

-Establecimiento de objetivos: Se identifican restricciones, se traza un plan de gestión, se identifican riesgos

-Valoración y reducción del riesgo: Se analiza cada riesgo identificado y se determinan acciones

-Desarrollo y validación: Se determina modelo de desarrollo

-Planeación: El proyecto se revisa y se toma decisiones para la siguiente fase



CLASE 8

METODOLOGIAS AGILES

Es un enfoque iterativo e incremental (evolutivo) de desarrollo de software que emerge como una posible solución a las metodologías con énfasis en el control del proceso, definiendo roles, actividades, herramientas y documentación detallada.

Objetivos:

-Producir software de alta calidad con un costo efectivo y en el tiempo apropiado.

-Esbozar los valores y principios que permiten desarrollar software rápidamente y responder a los cambios durante el proyecto

-Ofrecer una alternativa a los procesos de desarrollo de software rígidos y dirigidos por la documentación

Dan prioridad a las tareas que dan resultados directos y que reducen la burocracia tanto como sea posible

Valores:

-*Individuos e interacciones* más que procesos y herramientas.

-*Software operante* más que documentaciones completas.

-*Colaboración con el cliente* más que negociaciones contractuales.

-*Respuesta al cambio* más que apegarse a una rigurosa planificación.

Principios:

1. Satisfacer al cliente con entregas fáciles y continuas de software
2. Se capturan los cambios para que el cliente obtenga ventajas competitivas
3. Entregas frecuentes de software cada pocas semanas con intervalos mínimos
4. Usuarios y desarrolladores deben trabajar juntos durante todo el proyecto.
5. Construir proyectos alrededor de motivaciones individuales.
6. Proporcionar el ambiente, soporte y confianza en el equipo y fomentar el dialogo cara a cara
7. El software que funciona es la medida clave de progreso
8. Promueven un desarrollo sostenible para que los stakeholders, desarrolladores y usuarios puedan seguir el paso
9. Atención continua a la excelencia técnica y buen diseño incrementa la agilidad.
10. Simplicidad (maximizar la cantidad de trabajo no dado) es esencial.
11. Que los equipos definían las mejores arquitecturas, requerimientos y diseños

12. Que los equipos reflexionen regularmente para mejorar su efectividad y comportamiento

DIFERENCIAS ENTRE LAS METODOLOGÍAS

Metodología ágil	**Metodología No Ágil**
Pocos Artefactos	Más Artefactos
Pocos Roles	Más Roles
No existe un contrato tradicional o al menos es bastante flexible	Existe un contrato prefijado
El cliente es parte del equipo de desarrollo (además in-situ)	El cliente interactúa con el equipo de desarrollo mediante reuniones
Grupos pequeños (< 10 integrantes) y trabajando en el mismo sitio	Grupos grandes
Menos énfasis en la arquitectura	La arquitectura es esencial

Desventajas:

-Los representantes del cliente están sujetos a otras presiones y no intervienen por completo en el desarrollo del software.

-Priorizar los cambios puede ser difícil en sistemas con muchos participantes por las prioridades de los cambios de cada uno

-Bajo la presión de fechas de entrega, es posible que se carezca de tiempo para realizar las simplificaciones al sistema.

-A las empresas les cuesta cambiarse a un modelo de trabajo donde los procesos sean informales y estén definidos por equipos de desarrollo.

-Es complejo reglamentar los documentos de requerimientos por la minimización de la documentación

EXTREME PROGRAMMING (XP)

Se basa en los valores de la sencillez, la comunicación, la retroalimentación, la valentía y el respeto
Consiste en llevar a todo el equipo reunido en la presencia de prácticas simples, con suficiente información para ver dónde están y para ajustar las prácticas a su situación particular.

Practicas/características:

Testing: Los programadores escriben pruebas unitarias continuas, las cuales deben correr sin problemas para que el desarrollo continúe.

Refactoring: Actividad constante de reestructuración del código para eliminar código duplicado, mejorar su legibilidad, simplificarlo y hacerlo más flexible a los cambios

Programación de a Pares: Se escribe de a 2 programadores en 1 maquina

Propiedad Colectiva del Código: Cualquiera puede cambiar código en cualquier parte del sistema en cualquier momento. Fomentando las nuevas ideas y evitando dependencias.

Integración Continua: Cada pieza de código es integrada en el sistema una vez que esté lista. Reduciendo la fragmentación de los esfuerzos por falta de comunicación

Semana de 40-horas: Se debe trabajar un máximo de 40 horas por semana. Para evitar desmotivar al equipo y retrasar el proyecto. Priorizando la planificación sobre el trabajo extra

Cliente en el Lugar de Desarrollo: El cliente tiene que estar presente y disponible todo el tiempo para el equipo.

Estándares de Codificación: Los programadores siguen reglas de código que enfatizan la comunicación a través del mismo.

Simplicidad de código: Se prioriza la creación de soluciones simples y directas

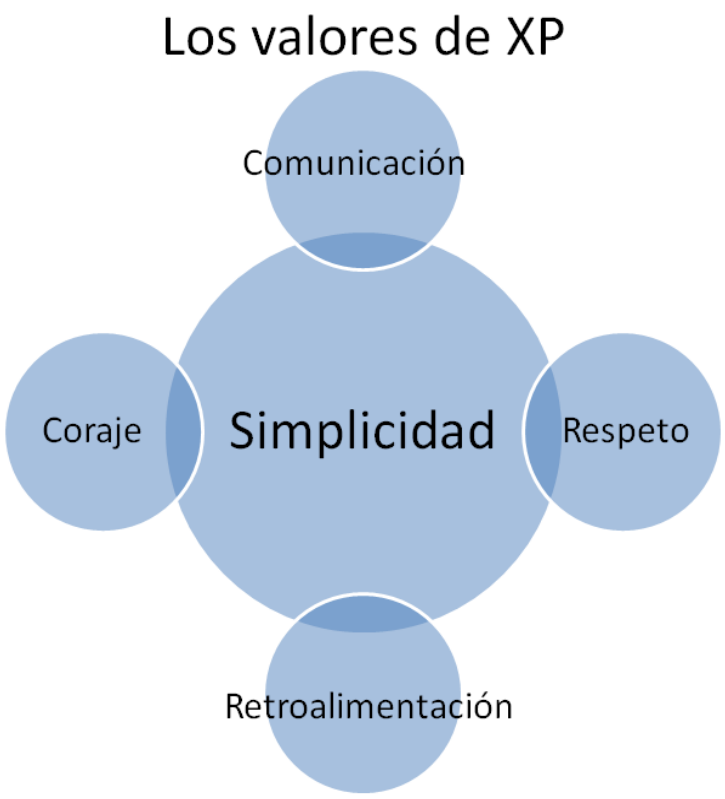


Imagen elaborada por oficinaproyectosinformatica.blogspot.com

Elementos clave:

-Historias de usuario

-Roles

-Proceso

-Prácticas

Roles:

-Programador: Tomas decisiones técnicas, construyen el sistema y diseñan, programan y realizan pruebas

-Jefe de proyecto: Organiza y guía reuniones y asegura las condiciones para el proyecto

-Cliente: Determina que construir y cuando, estable las pruebas funcionales y es parte del equipo

-Entrenador: Responsable del proyecto y pasa a segundo plano cuando el equipo madura

-Encargado de pruebas: Ayuda al cliente con las pruebas funcionales y asegura que se superen

-Rastreador: Observa sin molesta y conserva datos históricos

Ciclo de vida:

-Exploración: El cliente plantea las HU, el equipo se familiariza con las herramientas, tecnologías y prácticas y se crea un prototipo (tarda de semanas a meses)

-Planificación: El cliente le da a las HU prioridades, se estima el esfuerzo necesario, se hacen acuerdos de la primera entrega y se hace un cronograma (tarda unos días)

-Iteración: El plan de entrega está compuesto por iteraciones de menos de 3 semanas, el cliente elige las HU de cada iteración y al final entra a producción

-Producción: Requiere de pruebas adicionales y revisiones de rendimiento antes de ser trasladado al entorno del cliente y discutir agregar características a la versión actual por los cambios de la fase

-Mantenimiento: Mantiene el sistema en funcionamiento al mismo tiempo que desarrolla nuevas iteraciones y puede necesitar nuevo personal y cambios en su estructura

-Muerte: Ya no hay más HU que agregar, se genera la documentación final y no hay más cambios a la arquitectura. Si no hay beneficios buenos o presupuesto el proyecto muere

SCRUM

Un marco de trabajo para el desarrollo colaborativo para productos complejos, caracterizado por la aplicación regular de buenas prácticas, con el fin de atacar todos los problemas que surgen durante el desarrollo. Proporciona un paradigma de trabajo que soporta la innovación, permite la auto organización del equipo y garantiza entregas parciales y regulares, priorizadas por su beneficio al receptor del proyecto.

Principios:

-Eliminar el desperdicio: no generar artefactos ni perder el tiempo haciendo cosas sin valor para el cliente.

-Construir la calidad con el producto: Inyectar la calidad directamente en el código desde el inicio.

-Crear conocimiento: En la práctica no se puede tener el conocimiento antes de empezar el desarrollo.

-Diferir las decisiones: Tomar las decisiones en el momento adecuado y esperar hasta ese momento ya que es, si se puede esperar, mejor.

-Entregar rápido: Debe ser una de las ventajas competitivas más importantes.

-Respetar a las personas: Proveer un ambiente que motive y respete a las personas.

-Optimizar el todo: Optimizar todo el proceso, ya que es una unidad, para lograr tener éxito y avanzar

Roles:

-Propietario: Conoce y marca las prioridades del proyecto o producto

-Jefe: Garantiza el seguimiento de la metodología, guiando reuniones y apoyando al equipo ante desafíos, actuando como paraguas frente a presiones externas

-Equipo: Implementan la funcionalidad o funcionalidades elegidas por el propietario

-Usuario/cliente: Beneficiarios finales del producto y pueden aportar ideas, sugerencias o necesidades al ver los avances

Artefactos:

-Product Backlog: Lista maestra de toda la funcionalidad deseada en el producto, la funcionalidad esta ordenada por prioridad

-Sprint Backlog: Lista de funcionalidades que el equipo se comprometió a desarrollar durante un Sprint específico

-Burndown Chart: Muestra un acumulativo del trabajo hecho día a día

Proceso: Es iterativo e incremental y las fases de desarrollo se solapan ejecutando una tras otra, hasta que se crea suficiente

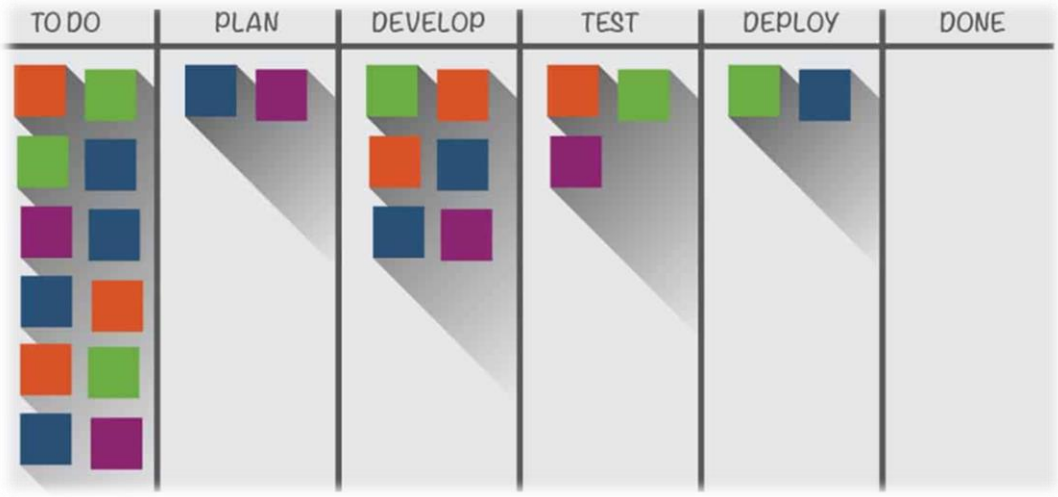
Uso: Se usa en proyectos con muchos requerimientos dinámicos y que usan tecnología de punta

KANBAN (看板)

Es una herramienta que limita el trabajo en curso, visualiza el flujo de trabajo y mide el tiempo de entrega promedio

Se suele combinar con otras metodologías ya que no define roles, no prescribe pasos, es sencilla de implementar, administra el flujo de trabajo y visualiza el proceso de desarrollo explícitamente

Evita explícitamente los problemas de un proceso cascada mediante lotes pequeños de trabajo y el desarrollo incremental. Se van moviendo las tarjetas entre columnas mostrando la evolución del desarrollo. Cada columna tiene un límite y cuando se termina una tarea se mueve a otra columna creando espacio libre en la columna



DESAROLLO DE SOFTWARE BASADO EN MODELOS

Se acorta a MBD

Dice que la construcción de un sistema de software debe ser precedida por la construcción de un modelo

Modelo: conceptualización del dominio del problema y actúa como una especificación precisa de los requerimientos que el sistema de software debe satisfacer

DESAROLLO DE SOFTWARE DIRIGIDO POR MODELOS

Se acorta a MDD

A diferencia del MBD el MDD asigna a los modelos un rol central y activo al nivel del código fuente y se convierten en entidades productivas de las que se deriva la implementación automáticamente. Es una evolución del MBD por lo que sus principios son reformulaciones y asociaciones de ideas anteriores

Puntos clave:

-Mayor nivel de abstracción en la especificación del problema a resolver y de la solución correspondiente.

-Aumento de confianza en la automatización asistida por computadora para soportar el análisis, el diseño y la ejecución.

-Uso de estándares industriales para facilitar las comunicaciones, la interacción entre diferentes aplicaciones y productos, y la especialización tecnológica.

-Los modelos son los conductores primarios en todos los aspectos del desarrollo de software.

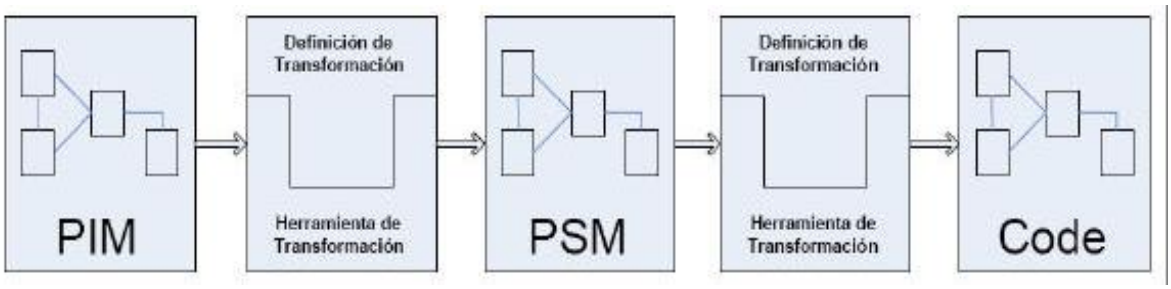
Modelos:

-PIM: Modelo de un sistema sin información de la plataforma o la tecnología usada para implementarlo

-PSM: Modelo de un sistema con información de la tecnología específica usada para su implementación sobre una plataforma específica

-Transformación de modelos: Proceso de conversión de un modelo a otro modelo del mismo sistema (incluye un PIM, un modelo de plataforma, una transformación y un PSM como mínimo)

-Transformación: Colección de reglas no ambiguas de las formas en que un modelo (o parte de él) puede ser usado para crear otro modelo (o parte de él).



La técnica de transformación se asemeja al proceso de abstracción y refinamiento presentado por Dijkstra

Beneficios:

-Incremento en la productividad (modelos y transformaciones).

-Adaptación a los cambios tecnológicos.

-Adaptación a los cambios de requisitos.

-Consistencia (automatización).

-Re-uso (de modelos y transformaciones).

-Mejoras en la comunicación con los usuarios y la comunicación entre los desarrolladores (los modelos permanecen actualizados).

-Captura de la experiencia (cambio de experto).

-Los modelos son productos de larga duración (resisten cambios).

-Posibilidad de demorar decisiones tecnológicas

CALIDAD

CLASE 9 YAY

Su significado sigue siendo ambiguo ya que varía de persona a persona, pero no es mesurable y tampoco por ser un producto de lujo se asegura que sea de calidad

Definiciones más aceptadas:

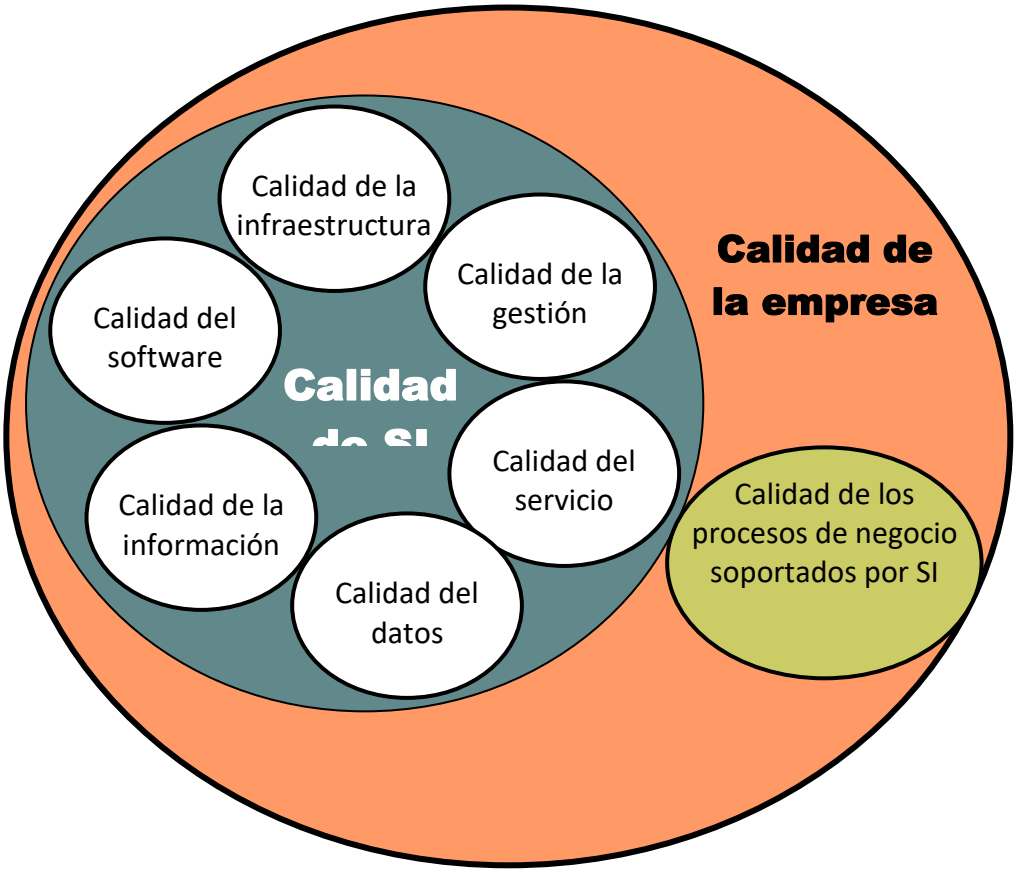
-Conjunto de propiedades o características de un producto o servicio que le confieren aptitud para satisfacer unas necesidades expresadas o implícitas

-El grado en el que un conjunto de características inherentes cumple con los requisitos

SISTEMAS DE INFORMACION

Conjunto de personas, datos, procesos y tecnología de información que interactúan para recopilar, procesar, guardar y proporcionar como salida la información necesaria para brindar soporte a una organización

La calidad debe apreciarse desde un todo, donde cada parte que la componen debe tener su análisis de calidad.



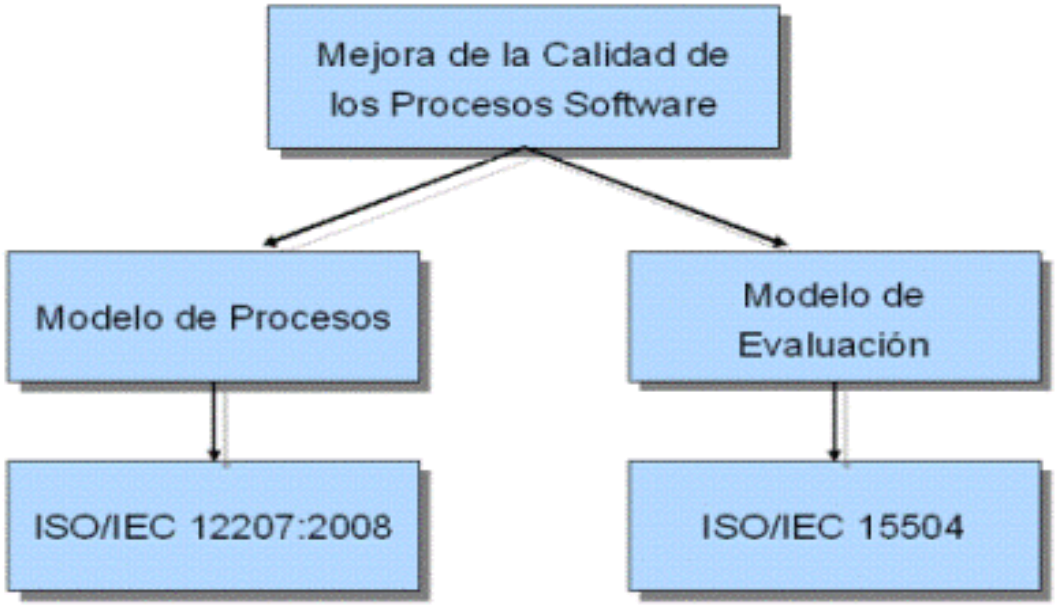
CALIDAD DE SOFTWARE

Se divide en calidad del producto y calidad del proceso (dependientes entre si) y es de gran importancia en la industria por la gestión de calidad y las técnicas de gestión de calidad

PASAR A LA TABLA DEL FINAL (MODELOS PARA EVALUAR LA CALIDAD)

Calidad del producto: La estandarización del producto define las propiedades que debe satisfacer el producto software resultante.

Calidad del proceso: La estandarización del proceso define la manera de desarrollar el producto software.



CMM (1993)-CMMI (2000)

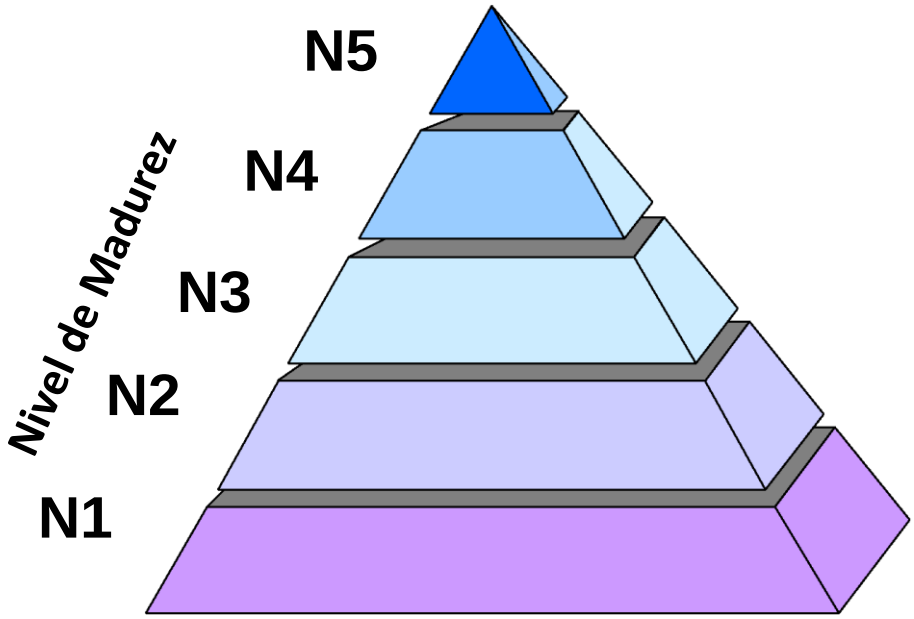
CMM: Modelo de evaluación de los procesos de una organización. Es el marco de referencia para desarrollar procesos efectivos y proporciona un marco estructurado para evaluar los procesos actuales de la organización, establecer prioridades de mejora e implementar esas mejoras

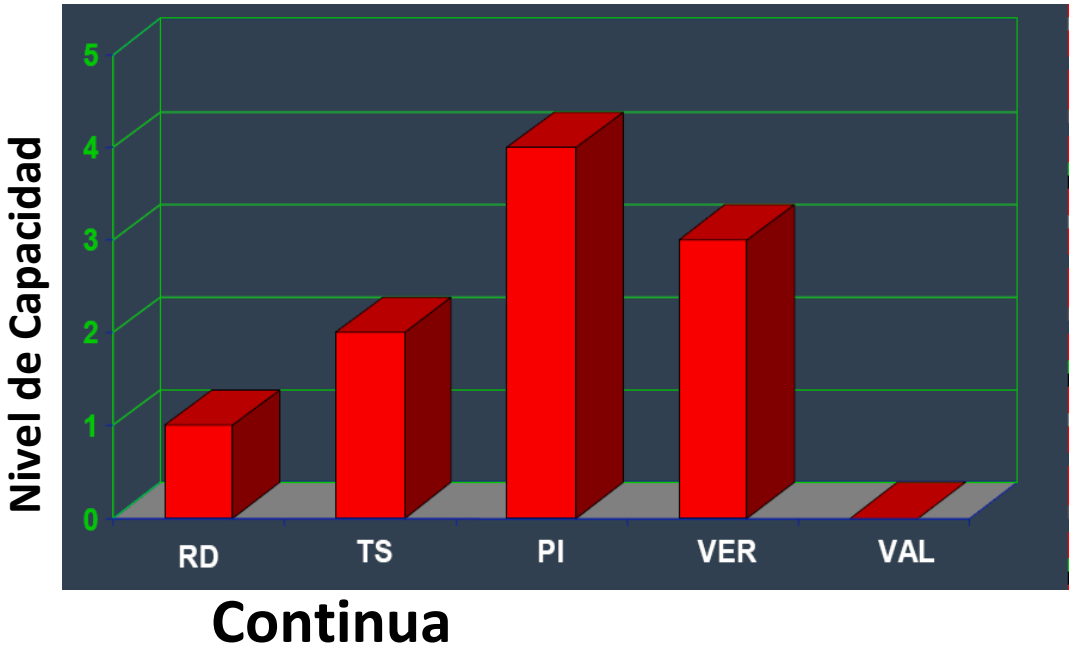
CMMI:

-Escalonado: Esta centrado en la madurez de la organización (igual al CMM)

-Continuo: Enfoca las actividades de mejora y evaluación en la capacidad de los diferentes procesos. Los niveles de capacidad (tiene 6) indican qué tan bien se desempeña la organización en un área de proceso individual.

Por etapas (escalonado)





NIVELES DE MADUREZ

-Inicial: Proceso impredecible, poco controlado y reactivo

-Gestionado: Proceso caracterizado por proyectos y frecuentemente reactivo

-Definido: Proceso caracterizado por la organización y proactivo

-Gestionado Cuantitativamente**:** El proceso es controlado cuantitativamente

-Optimizado Enfoque en la mejora del proceso

ISO

ISO 9000: Conjunto de normas de gestión de la calidad aplicables a cualquier tipo de organización para obtener mejoras en la organización y certificar a la organización para el mercado global

ISO – 9001:2015: Quality management system – Requirements

IRAM – ISO 9001:2015: Sistema de gestión de la calidad – Requisitos (traducida por IRAM)

(IRAM significa Instituto Argentino de Normalización y Certificación)

ISO 90003:2004: Se basa en ISO 9001:2000

-Directrices para la interpretación en el proceso de software

-Guía para identificar que el proceso de software satisfaga los requisitos de la ISO 9001

ISO 9001: Asegura que su negocio cumpla con los requisitos legales y del cliente.

-Aumenta el rendimiento de su organización. El Sistema de Gestión de la Calidad, ayuda a implementar procesos simplificados y mejorar la eficiencia operacional.

-Asegura la toma de decisiones y mejore la satisfacción del cliente.

-Optimiza sus operaciones para así cumplir y superar los requisitos de sus clientes.

-Mejora su rendimiento financiero.

あ

え

い

お

う

トメイク

Calidad de producto de software	Se evalúa la calidad mediante	ISO/IEC 25000	Esta compuesto por distintos modelos. Define características que pueden estar presentes o no en el producto. La norma nos permite evaluar si están presentes o no , y de que manera evaluarlas. EJ: Seguridad, Compatibilidad, Seguridad. Etc.
Calidad de proceso de desarrollo de software	Se evalúa la calidad mediante	ISO/IEC 12207	ISO/IEC 12207 establece un modelo de procesos para el ciclo de vida del software. Define cómo debería ser el modelo de proceso para ser completo y con calidad. Actividades, tareas etc.
		ISO/IEC 15504 (reemplazada por ISO 33000)	Es una norma internacional para establecer y mejorar la capacidad y madurez de los procesos Define que se debe tener en cuenta para evaluar el modelo de proceso y concluir si es completo y con calidad.
		ISO/IEC 90003	Proporciona una guía sobre cómo aplicar la ISO 9001 en procesos de software
		CMMI	Proporciona un marco estructurado para evaluar los procesos actuales de la organización, establecer prioridades de mejora, e implementar esas mejoras. Se utiliza para organizaciones desarrolladoras de software de medianas a grandes dimensiones
Calidad de Procesos/Servicios en general	se evalúa mediante	ISO 9001	La Norma ISO 9001 determina los requisitos para establecer un Sistema de Gestión de la Calidad. Forma parte de la familia ISO 9000, que es un conjunto de normas de “gestión de la calidad” aplicables a cualquier tipo de organización con el objetivo de obtener mejoras en la organización y, eventualmente arribar a una certificación, punto importante a la hora de competir en los mercados globales.