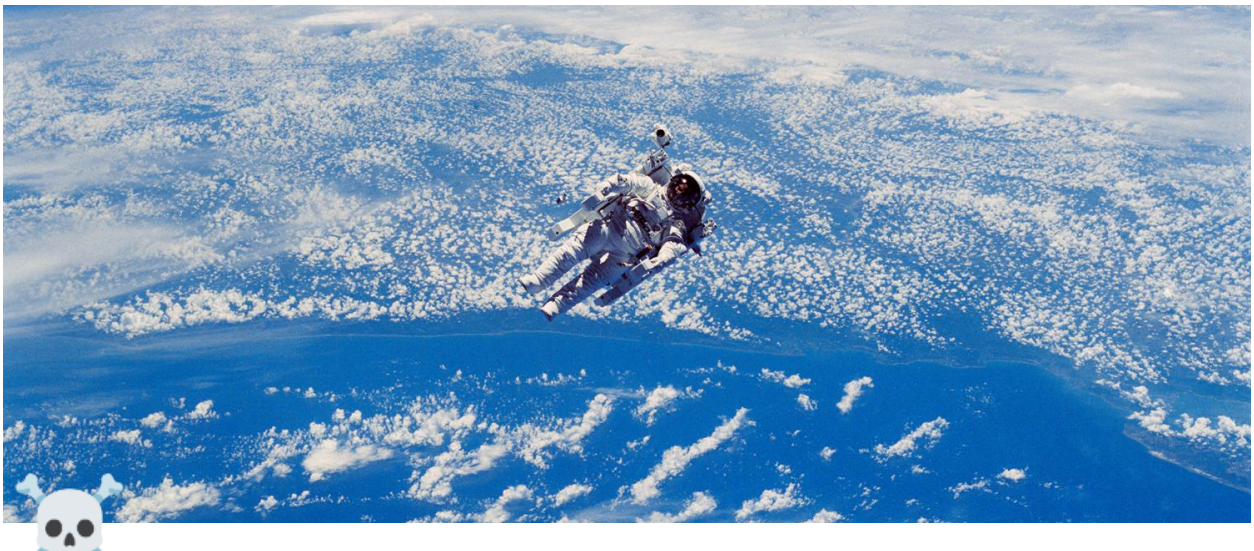


Resumen ING1 TEORIA

Clase 1

»¿Qué es Software?

▪ Es el conjunto de los programas de cómputo, procedimientos, reglas, documentación y datos asociados que forman parte de las operaciones de un sistema de computación (IEEE)

»Tipos de producto de software

▪ Genéricos: Sistemas aislados producidos por organizaciones desarrolladoras de software y que se venden en un mercado abierto.

▪ Personalizados: Sistemas requeridos por un cliente en particular. Desarrollados por la propia organización interesada o un contratista.

»Clasificación de Software

▪ De sistemas (sirve a otros programas)

▪ De gestión (proceso de información comercial)

▪ Científico (algoritmos de manejo de números)

▪ Empotrado (reside en memoria)

▪ De tiempo real (coordina/analiza/controla sucesos del mundo real)

▪ Basados en la Web (sitios)

▪ De Inteligencia artificial (uso de algoritmos no numéricos para resolver problemas complejos) ▪ Otros…

»Características del Software

▪ Es un elemento lógico

▪ El software se desarrolla, no se fabrica como otros productos

▪ Mayor costo en la ingeniería que en la producción.

▪ Tendencia importante de construcción por componentes, pero aún se construyen a medida

»Evolución del software

▪ El software no se desgasta.

▪ No sigue una curva clásica de envejecimiento.

▪ Es inmune a los males que desgastan al hardware.

▪ El problema no está en el tiempo de operación, sino en los cambios.

»¿Qué es la ingeniería 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.

»En resumen:

▪ La ingeniería de software trata de dar principios y métodos que permitan producir software confiable y eficiente, al menor costo posible.

▪ Para esto la ingeniería de software establece métodos, desarrolla herramientas automáticas o semiautomáticas y define procedimientos que establecen la relación de métodos y herramientas.

»El Ingeniero de Software debe:

▪ Tener una combinación de conocimientos científicos, metodológicos, tecnológicos y administrativos.

▪ Estar familiarizado con la aplicación de métodos formales: lógica, estadística, simulación y con el uso de notaciones de modelización, especificación, diseño, programación

▪ Poder aplicar metodologías de documentación, análisis, especificación, diseño, implementación y prueba. Debe conocer las ventajas y limitaciones de cada notación y cada técnica. Debe saber cómo y cuándo aplicarlas.

▪ Conocer las tecnologías y productos: sistemas operativos, lenguajes, herramientas CASE, bases de datos, sistemas generadores de interfaces, bibliotecas de código.

▪ Conocer técnicas de administración de proyectos: planificación, análisis de riesgos, control de calidad, seguimiento de proyectos, control de subcontratistas, etc.

▪ En los últimos años se observa una especialización de los ingenieros de software por dominio de aplicación o por actividad

RESPONSABILIDAD PROFESIONAL Y ÉTICA

»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

»Existen diferentes organizaciones cono ACM o IEEE que sugieren diferentes códigos de ética a respetar

REQUERIMIENTOS

»Un Requerimiento (o requisito) es una característica del sistema o una descripción de algo que el sistema es capaz de hacer con el objeto de satisfacer el propósito del sistema

»Definición IEEE-Std-610

1. Condición o capacidad que necesita el usuario para resolver un problema o alcanzar un objetivo
2. Condición o capacidad que debe satisfacer o poseer un sistema o una componente de un sistema para satisfacer un contrato, un estándar, una especificación u otro documento formalmente impuesto.
3. Representación documentada de una condición o capacidad como en 1 o 2.

»Fuentes de Requerimientos

▪ Documentación

▪ Los **stakeholder**

▪ El termino **Stakeholder** se utiliza para referirse a cualquier persona o grupo que se vera afectado por el sistema, directa o indirectamente.

▪ Entre los Stakeholders se encuentran:

▪ Usuarios finales, Ingenieros, Gerentes, Expertos del dominio

▪ Especificaciones de sistemas similares

PUNTOS DE VISTA

»Clasificar los stakeholders según los punto de vista:

▪ Punto de vista de los **interactuadores**: representan a las personas u otros sistemas que interactúan directamente con el sistema. Pueden influir en los requerimientos del sistema de algún modo.

▪ Punto de vista **indirecto**: representan a los stakeholders que no utilizan el sistema ellos mismos pero que influyen en los requerimientos de algún modo.

▪ Punto de vista del **dominio**: representan las características y restricciones del dominio que influyen en los requerimientos del sistema.

PUNTOS DE VISTA

»Los más específicos son:

1. Los proveedores de servicios al sistema, los receptores de servicios del sistema
2. Los sistemas que deben interactuar
3. Las regulaciones y estándares a aplicar
4. Las fuentes de requerimientos
5. Los puntos de vista de las personas que lo van a desarrollar, administrar y mantener.
6. Puntos de vista del marketing y otros que generan requerimientos sobre las características del sistema

ELICITACIÓN DE REQUISITOS

»Es el proceso de adquirir (“eliciting”) [sonsacar] todo el conocimiento relevante necesario para producir un modelo de los requerimientos de un dominio de problema

»Objetivos:

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

▪ Conocer el sistema actual (manual o informatizado).

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

»La elicitación de requisitos es una actividad principalmente de carácter social, mucho más que tecnológico.

»Los problemas que se plantean son por tanto de naturaleza psicológica y social, más que técnicos.

»Problemas de comunicación

Del lado del cliente

▪ Dificultad para expresar claramente las necesidades.

▪ No ser conscientes de sus propias necesidades.

▪ No entender cómo la tecnología puede ayudar.

▪ Miedo a parecer incompetentes por ignorancia tecnológica.

▪ No tomar decisiones por no poder prever las consecuencias, no entender las alternativas o no tener una visión global.

Del lado del desarrollador

▪ Cultura y vocabulario diferentes.

▪ Intereses distintos en el sistema a desarrollar.

▪ Medios de comunicación inadecuados (diagramas ▪ que no entienden los clientes y usuarios). ▪ Conflictos personales o políticos.

»Limitaciones cognitivas (del desarrollador)
▪ No conocer el dominio del problema.
▪ Hacer suposiciones sobre el dominio del problema.
▪ Hacer suposiciones sobre aspectos tecnológicos.

▪ Hacer simplificaciones excesivas.
»Conducta humana
▪ Conflictos y ambigüedades en los roles de los participantes.
▪ Pasividad de clientes, usuarios o ingenieros de requisitos.
▪ Temor a que el nuevo sistema lo deje sin trabajo.
»Técnicos
▪ Complejidad del dominio del problema.
▪ Complejidad de los requisitos.
▪ Múltiples fuentes de requisitos.
▪ Fuentes de información poco claras.

TÉCNICAS DE ELICITACIÓN

Recopilación de información: **Métodos discretos**

1. Muestreo de la documentación, los formularios y los datos existentes.
2. Investigación y visitas al lugar.
3. Observación del ambiente de trabajo.

Métodos interactivos

1. Cuestionarios.
2. Entrevistas.
3. Planeación conjunta de Requerimientos (JRP o JAD).
4. Lluvia de Ideas - Brainstorming .

DISCRETOS

MUESTREO DE LA DOCUMENTACIÓN, LAS FORMAS Y LOS DATOS EXISTENTES

»Recolección de hechos a partir de la documentación existente

▪ Documentos que describen la funcionalidad del negocio que esta siendo analizada

▪ Declaración de la misión y plan estratégico de la organización

▪ Objetivos formales del departamento en cuestión

▪ Políticas, restricciones, procedimientos operativos

▪ Formularios de operaciones realizadas

▪ Bases de Datos

▪ Sistemas en funcionamiento

▪ Documentación de sistemas anteriores

▪ Diagramas

▪ Diccionario o Repositorios de proyecto

▪ Documentos de diseño

▪ Manuales de operación y/o entrenamiento

INVESTIGACIÓN Y VISITAS AL SITIO

»Investigar el dominio

»Patrones de soluciones (mismo problema en otra organización)

»Revistas especializadas

»Buscar problemas similares en internet

»Consultar otras organizaciones

OBSERVACIÓN DEL AMBIENTE DE TRABAJO

»El analista se convierte en observador de las personas y actividades con el objeto de aprender acerca del sistema.

»Lineamientos de la observación:

▪ Determinar quién y cuándo será observado

▪ Obtener el permiso de la persona y explicar el porqué será observado

▪ Mantener bajo perfil

▪ Tomar nota de lo observado

▪ Revisar las notas con la persona apropiada

▪ No interrumpir a la persona en su trabajo

»Ventajas

▪ Datos confiables

▪ El analista puede ver exactamente lo que se hace (tareas difíciles de explicar con palabras)

▪ Análisis de disposiciones físicas, tránsito, iluminación, ruido

▪ Económica en comparación con otras técnicas

»Desventajas

▪ La gente se siente incómoda siendo observada

▪ Algunas actividades del sistema pueden ser realizadas en horarios incómodos

▪ Las tareas están sujetas a interrupciones

▪ Tener en cuenta que la persona observada puede estar realizando las tareas de la forma “correcta” y no como lo hace habitualmente

INTERACTIVOS

Cuestionarios

»Documento que permite al analista recabar información y opiniones de los encuestados

▪ Recolectar hechos de un gran número de personas

▪ Detectar un seguimiento generalizado

▪ Detectar problemas entre usuarios

▪ Cuantificar respuestas

»Ventajas

▪ Respuesta rápida

▪ Económicos

▪ Anónimos

▪ Estructurados de fácil análisis

»Desventajas

▪ Número bajo de respuestas

▪ No responde a todas las preguntas

▪ Preguntas rígidas

▪ No se puede analizar el análisis corporal

▪ No se pueden aclarar respuestas incompletas

▪ Difíciles de preparar

»Tipos de Cuestionario

▪ **Formato libre (Abiertos)**

▪ Diseñado para ofrecer al encuestado más flexibilidad en la respuesta

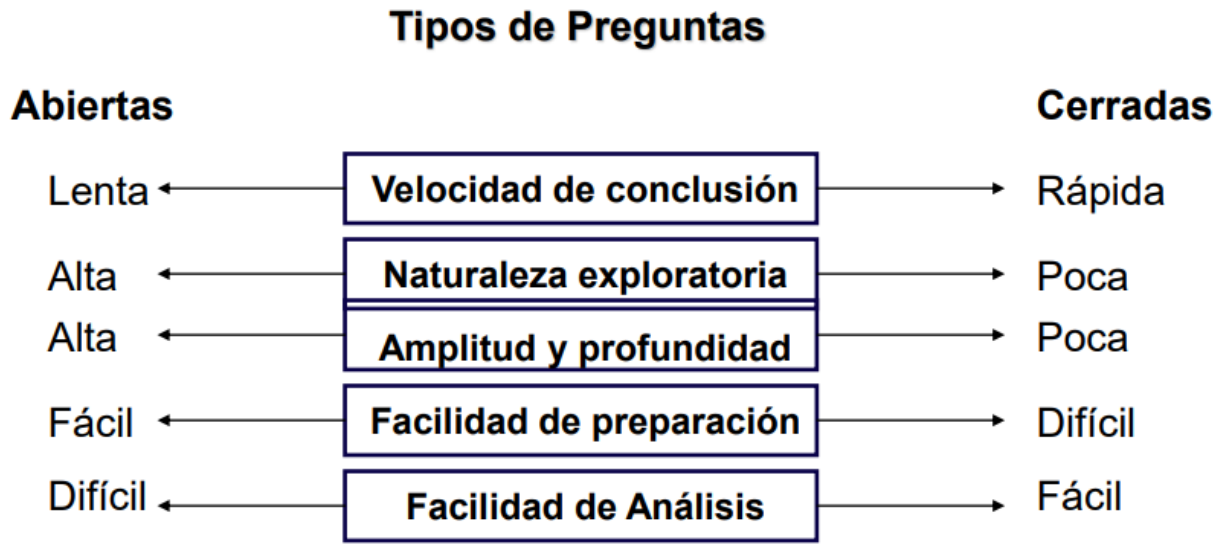
▪ ¿Qué reportes recibe actualmente?

▪ ¿Hay problemas con los reportes?

▪ **Formato fijo (Cerrados)**

▪ Requieren la selección de una respuesta entre respuestas posibles predefinidas

▪ ¿Es útil el reporte que utiliza actualmente? SI NO



»Tipo de información obtenida

▪ Actitud

▪ Lo que las personas dicen que quieren

▪ Creencias

▪ Lo que las personas creen que es verdad

▪ Comportamiento

▪ Lo que realmente hacen

▪ Características

▪ De las personas o cosas

»Cuándo usar Cuestionarios

▪ Las personas están dispersas

▪ Diferentes oficinas o ciudades

▪ Muchas de personas involucradas

▪ Clientes o usuarios

▪ Queremos obtener opiniones generales

▪ Queremos identificar problemas generales

»Redacción de un Cuestionario

▪ Evitar la parcialidad

▪ No incluir tendencias a una respuesta

▪ Evitar preguntas ofensivas

▪ Si el encuestado se siente ofendido, posiblemente no responda de forma correcta

▪ Dirigir las preguntas a los encuestados que las pueden responder

▪ El cuestionario debe ser preparado para el grupo de personas que lo van a responder

▪ No usar un Cuestionario genérico para todos

ENTREVISTAS

»Técnica de exploración mediante la cual el analista de sistemas recolecta información de las personas a través de la interacción cara a cara

»Es una conversación con un propósito especifico, que se basa en un formato de preguntas y respuestas en general

»Conocer opiniones y sentimientos del entrevistado

»Ventajas

▪ El entrevistado se siente incluido en el proyecto

▪ Es posible obtener una retroalimentación del encuestado

▪ Es posible adaptar las preguntas de acuerdo al entrevistado

▪ Información no verbal

▪ Observando las acciones y expresiones del entrevistado

»Desventaja

▪ Costosas

▪ Tiempo y recursos humanos

▪ Las entrevistas dependen en gran parte de las habilidades del entrevistador

▪ No aplicable a distancia

»Tipos de entrevistas

▪ **Estructuradas (Cerradas)**

▪ El encuestador tiene un conjunto específico de preguntas para hacérselas al entrevistado

▪ Se dirige al usuario sobre un requerimiento puntual

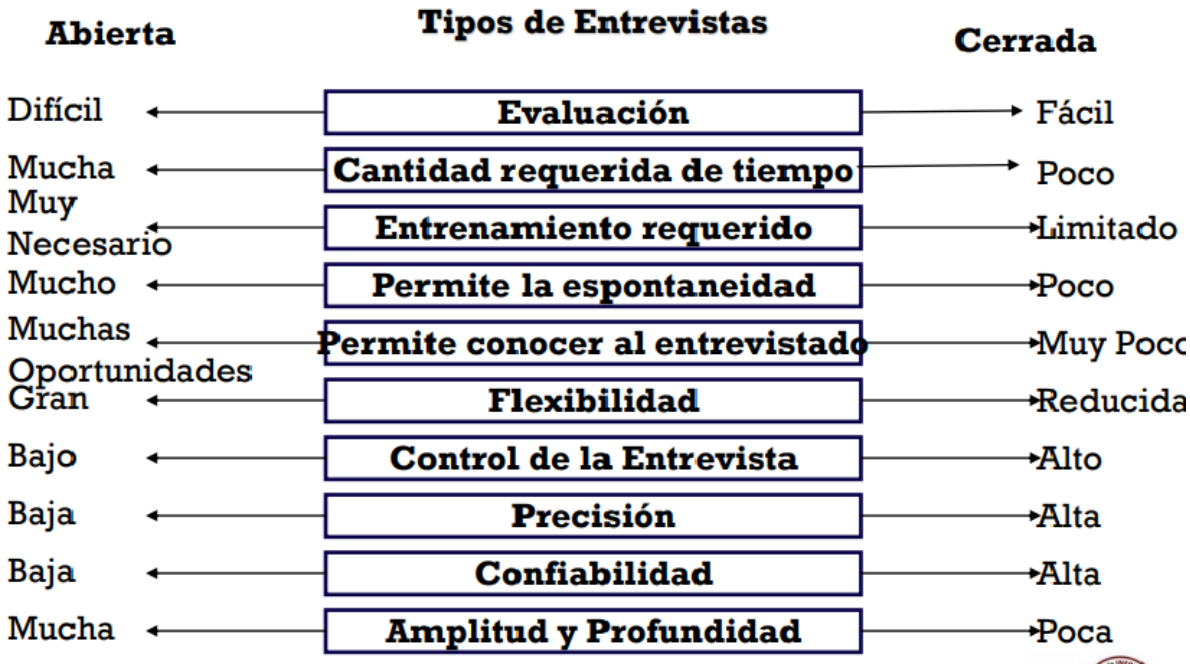
▪ No permite adquirir un amplio conocimiento del dominio

▪ **No estructuradas (Abiertas)**

▪ El encuestador lleva a un tema en general

▪ Sin preparación de preguntas específicas

▪ Iniciar con preguntas que no dependen del contexto, para conocer el problema, la gente involucrada, etc.



»Preguntas Abiertas

- **Ventajas**
- Revelan nueva línea de preguntas
- Hacen más interesante la entrevista
- Permiten espontaneidad
- **Desventajas**
- Pueden dar muchos detalles irrelevantes
- Se puede perder el control de la entrevista
- Parece que el entrevistador no tiene los objetivos claros

»Preguntas cerradas

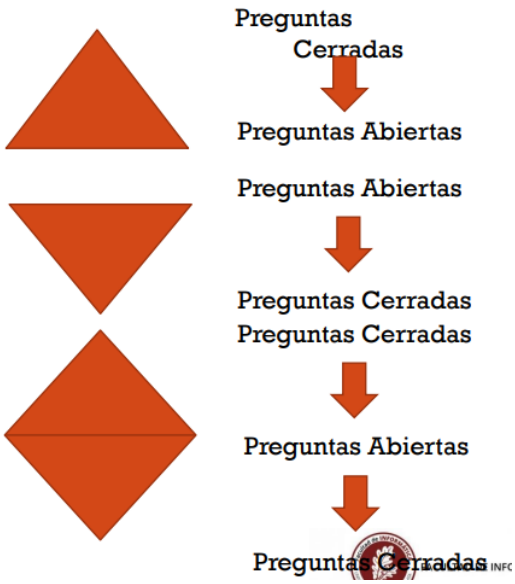
- **Ventajas**
- Ahorran tiempo
- Se mantiene más fácil el control de la entrevista
- Se consiguen datos relevantes
- **Desventajas**

▪ Pueden aburrir al encuestado

▪ No se obtienen detalles

»Organización de una entrevista

- Piramidal (Inductivo)
- Embudo (Deductivo)
- Diamante (Comb. de las anteriores)



PLANEACIÓN CONJUNTA DE REQUERIMIENTO (JRP)

»Proceso mediante el cual se conducen reuniones de grupo altamente estructurados con el propósito de analizar problemas y definir requerimientos

▪ Requiere de extenso entrenamiento

▪ Reduce el tiempo de exploración de requisitos

▪ Amplia participación de los integrantes

▪ Se trabaja sobre lo que se va generando

▪ Alguna bibliografía la menciona como JAD (Joint Application Design)

»Ventajas

▪ Ahorro de tiempo

▪ Usuarios involucrados

▪ Desarrollos creativos

»Desventajas

▪ Es difícil organizar los horarios de los involucrados

▪ Es complejo encontrar un grupo de participantes integrados y organizados

»Participantes de JRP

▪ Patrocinador

▪ Miembro de la dirección con autoridad sobre los departamentos que participan, es el responsable del proyecto, toma las decisiones finales

▪ Facilitador

▪ Dirige las sesiones y tiene amplias habilidades de comunicación y negociación

▪ Usuarios y Gerentes

▪ Los usuarios comunican los requerimientos y los gerentes los aprueban

▪ Secretarios

▪ Llevan el registro de la sesión y van publicando los resultados realizados con herramientas CASE

▪ Equipos de TI

▪ Escuchan y toman nota de los requerimientos

»Cómo planear las sesiones de JRP

1. Selección de una ubicación para las sesiones de JRP
2. Selección de los participantes
3. Preparar la agenda

»Beneficios del JRP

1. JRP involucra activamente a los usuarios y la gerencia en el proyecto de desarrollo
2. JRP reduce el tiempo de la etapa de requerimientos
3. Si se incorporan prototipos, los mismos ya confirman el diseño del sistema

LLUVIA DE IDEAS (BRAINSTORMING)

»Técnica para generar ideas al alentar a los participantes para que ofrezcan tantas ideas como sea posible en un corto tiempo sin ningún análisis hasta que se hayan agotado las ideas.

»Se promueve el desarrollo de ideas creativas para obtener soluciones.

»Se realizan reuniones del equipo involucrado en la resolución del problema, conducidas por un director.

»Los principios en que se basa esta técnica son:

▪ “Cuantas más ideas se sugieren, mejores resultados se conseguirán”.

▪ La producción de ideas en grupos puede ser más efectiva que la individual.

▪ Las ideas de una persona pueden hacer que aparezcan otras por “contagio”.

▪ A veces las mejores ideas aparecen tarde.

▪ Es mejor elegir sobre una variedad de soluciones.

»Incluye una serie de fases de aplicación:

▪ Descubrir hechos, Producir ideas, Descubrir soluciones

»Clave para resolver la falta de consenso entre usuarios

»Es útil combinarlo con la toma de decisiones

»Ayuda a entender el dominio del problema

»Encara la dificultad del usuario para transmitir

»Ayuda a entender: al usuario y al analista

CLASE 2

¿QUÉ ES UN PROCESO DE SOFTWARE?

»Es un conjunto de actividades y resultados asociados que producen un producto de software

»Actividades fundamentales de los procesos

▪ Especificación del software

▪ Desarrollo del software

▪ Validación del software

▪ Evolución del software

¿QUÉ ES UN MODELO DE PROCESO DE SOFTWARE?

»Es una descripción simplificada de un proceso de software que presenta una visión de ese proceso.

»Estos modelos pueden incluir actividades que son partes de los procesos y productos de software, y el papel de las personas involucradas.

»La mayoría de los modelos de proceso de software se basan en uno de los siguientes modelos generales o paradigmas

▪ Modelo en cascada: Representa las actividades anteriores y las representa como fases de proceso separadas.

▪ Especificación de requerimientos, diseño, implantación, etc.

▪ Desarrollo iterativo : Un sistema inicial se desarrolla rápidamente a partir de una especificación abstracta. Éste se refina basándose en las peticiones del cliente.

▪ IS basada en componentes: Esta técnica supone que las partes ya existen. El proceso se enfoca en la integración de las partes.

¿CUÁLES SON LOS ATRIBUTOS DE UN BUEN SOFTWARE?

»Los productos de software tiene un cierto número de atributos asociados que reflejan su calidad.

▪ Estos atributos reflejan su comportamiento durante su ejecución y la estructura y organización de los programas fuentes en la documentación asociada

»Los atributos básicos son

▪ Mantenibilidad ▪ Posibilidad de modificaciones ante los cambios del negocio

▪ Confiabilidad ▪ Fiabilidad, seguridad, no debe causar daños físico o económicos ante fallas

▪ Eficiencia ▪ Hacer un uso apropiado de los recursos

▪ Usabilidad ▪ Fácil de usar sin esfuerzo adicional

REQUERIMIENTOS

»Un requerimiento (o requisito) es una característica del sistema o una descripción de algo que el sistema es capaz de hacer con el objeto de satisfacer el propósito del sistema

»Tipos de requerimientos

▪ Requerimientos funcionales

▪ Describen una interacción entre el sistema y su ambiente. Cómo debe comportarse el sistema ante determinado estímulo.

▪ Describen lo que el sistema debe hacer, o incluso cómo NO debe comportarse.

▪ Describen con detalle la funcionalidad del mismo.

▪ Son independientes de la implementación de la solución.

▪ Se pueden expresar de distintas formas

▪ Requerimientos no funcionales

▪ Describen una restricción sobre el sistema que limita nuestras elecciones en la construcción de una solución al problema.

Tipos de requerimientos no funcionales

▪ Requerimientos del producto

▪ Especifican el comportamiento del producto (usabilidad, eficiencia, rendimiento, espacio, fiabilidad, portabilidad).

▪ Requerimientos organizacionales

▪ Se derivan de las políticas y procedimientos existentes en la organización del cliente y en la del desarrollador (entrega, implementación, estándares).

▪ Requerimientos externos

▪ Interoperabilidad, legales, privacidad, seguridad, éticos.

INGENIERÍA DE REQUERIMIENTOS

»La ingeniería de requerimientos es la disciplina para desarrollar una especificación completa, consistente y no ambigua, la cual servirá como base para acuerdos comunes entre todas las partes involucradas y en donde se describen las funciones que realizará el sistema.

Ingeniería de requerimientos es el proceso por el cual se transforman los requerimientos declarados por los clientes, ya sean hablados o escritos, a especificaciones precisas, no ambiguas, consistentes y completas del comportamiento del sistema, incluyendo funciones, interfaces, rendimiento y limitaciones”

»También es el proceso mediante el cual se intercambian diferentes puntos de vista para recopilar y modelar lo que el sistema va a realizar. Este proceso utiliza una combinación de métodos, herramientas y actores, cuyo producto es un modelo del cual se genera un documento de requerimientos.”

»“Ingeniería de requerimientos” es un enfoque sistémico para recolectar, organizar y documentar los requerimientos del sistema; es también el proceso que establece y mantiene acuerdos sobre

los cambios de requerimientos, entre los clientes y el equipo del proyecto”

»Importancia

▪ Permite gestionar las necesidades del proyecto en forma estructurada

▪ Mejora la capacidad de predecir cronogramas de proyectos

▪ Disminuye los costos y retrasos del proyecto

▪ Mejora la calidad del software

▪ Mejora la comunicación entre equipos

▪ Evita rechazos de usuarios finales.

ESTUDIO DE VIABILIDAD

»Principalmente para sistemas nuevos

»A partir de una descripción resumida del sistema se elabora un informe que recomienda la conveniencia o no de realizar el proceso de desarrollo

»Responde a las siguientes preguntas:

▪ ¿El sistema contribuye a los objetivos generales de la organización?(Si no contribuye, entonces no tiene un valor real en el negocio)

▪ ¿El sistema se puede implementar con la tecnología actual?

▪ ¿El sistema se puede implementar con las restricciones de costo y tiempo?

▪ ¿El sistema puede integrarse a otros que existen en la organización?

▪Una vez que se ha recopilado toda la información necesaria para contestar las preguntas anteriores se debería hablar con las fuentes de información para responder nuevas preguntas y luego se redacta el informe, donde debería hacerse una recomendación sobre si debe continuar o no el desarrollo.

OBTENCIÓN Y ANÁLISIS DE REQUERIMIENTOS

»Propiedades de los Requerimientos

▪ Necesario: Su omisión provoca una deficiencia.

▪ Conciso: Fácil de leer y entender

▪ Completo: No necesita ampliarse

▪ Consistente: No contradictorio con otro

▪ No ambiguo: Tiene una sola implementación

▪ Verificable: Puede testearse a través de inspecciones, pruebas, etc.

ESPECIFICACIÓN DE REQUERIMIENTOS

»Objetivos

▪ Permitir que los desarrolladores expliquen cómo han entendido lo que el cliente pretende del sistema

▪ Indicar a los diseñadores qué funcionalidad y características va a tener el sistema resultante

▪ Indicar al equipo de pruebas qué demostraciones llevar a cabo para convencer al cliente de que el sistema que se le entrega es lo que había pedido.

▪ Rastreable ▪ Independiente del diseño ▪ Anotada ▪ Concisa ▪ Organizada ▪ Utilizable en operación y mantenimiento ▪ Correcta ▪ No ambigua ▪ Completa ▪ Verificable ▪ Consistente ▪ Comprensible por los consumidores ▪ Modificable

»Documento de definición de requerimientos

▪ Listado completo de todas las cosas que el cliente espera que haga el sistema propuesto
»Documento de especificación de requerimientos

▪ Definición en términos técnicos »Documento de especificación de requerimientos de Software IEEE Std. 830-1998 (SRS)

▪ Objetivo:

▪ Brindar una colección de buenas prácticas para escribir especificaciones de requerimientos de software (SRS).

▪ Se describen los contenidos y las cualidades de una buena especificación de requerimientos.

VALIDACIÓN DE REQUERIMIENTOS

»Es el proceso de certificar la corrección del modelo de requerimientos contra las intenciones del usuario.

»Trata de mostrar que los requerimientos definidos son los que estipula el sistema. Se describe el ambiente en el que debe operar el sistema.

»Es importante, porque los errores en los requerimientos pueden conducir a grandes costos si se descubren más tarde

Definición de la IEEE

▪ Validación ▪ Al final del desarrollo evaluar el software para asegurar que el software cumple los requerimientos

▪ Verificación ▪ Determinar si un producto de software de una fase cumple los requerimientos de la fase anterior

»Sobre estas definiciones:

▪ la validación sólo se puede hacer con la activa participación del usuario

▪ Validación: hacer el software correcto ▪

Verificación: hacer el software correctamente

»Comprenden

▪ Verificaciones de validez (para todos los usuarios)

▪ Verificaciones de consistencia (sin contradicciones)

▪ Verificaciones de completitud (todos los requerimientos)

▪ Verificaciones de realismo (se pueden implementar)

▪ Verificabilidad (se puede diseñar conjunto de pruebas)

»Técnicas de validación

▪ Pueden ser manuales o automatizadas

▪ Revisiones de requerimientos (formales o informales)

▪ Informales ▪ Los desarrolladores deben tratar los requerimientos con tantos stakeholders como sea posible.

▪ Formal ▪ El equipo de desarrollo debe conducir al cliente, explicándole las implicaciones de cada requerimiento

▪ Antes de una revisión formal, es conveniente realizar una revisión informal.

▪ Construcción de prototipos

▪ Generación de casos de prueba

TÉCNICAS DE ESPECIFICACIÓN DE REQUERIMIENTOS

»Estáticas

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

▪ Cuando el tiempo no es un factor mayor en la operación del sistema, es una descripción útil y adecuada.

▪ Referencia indirecta

▪ Relaciones de recurrencia

▪ Definición axiomática

▪ Expresiones regulares

▪ Abstracciones de datos

»Dinámicas

▪ Se considera un sistema en función de los cambios que ocurren a lo largo del tiempo.

▪ Se considera que el sistema está en un estado particular hasta que un estímulo lo obliga a cambiar su estado.

▪ Tablas de decisión

▪ Diagramas de transición de estados

▪ Tablas de transición de estados

▪ Diagramas de persianas

▪ Diagramas de transición extendidos,

▪ Redes de Petri ▪ Casos de Uso ▪ Historias de Usuario ▪ DFD/DFC

»Tablas de Decisión

▪ Es una herramienta que permite presentar de forma concisa las reglas lógicas que hay que utilizar para decidir acciones a ejecutar en función de las condiciones y la lógica de decisión de un problema específico.

»Describe el sistema como un conjunto de:

▪ Posibles CONDICIONES satisfechas por el sistema en un momento dado

▪ REGLAS para reaccionar ante los estímulos que ocurren cuando se reúnen determinados conjuntos de condiciones y

▪ ACCIONES a ser tomadas como un resultado.

HISTORIAS DE USUARIO

»Una historia de usuario es una representación de un requisito de software escrito en una o dos frases utilizando el lenguaje común del usuario.

»Son utilizadas en las metodologías de desarrollo ágiles para la especificación de requisitos
»Acompañadas de las discusiones con los usuarios y las pruebas de validación

»Debe ser limitada, esta debería poderse escribir sobre una nota adhesiva pequeña.

»Son una forma rápida de administrar los requisitos de los usuarios sin tener que elaborar gran cantidad de documentos formales y sin requerir de mucho tiempo para administrarlos.

»Permiten responder rápidamente a los requisitos cambiantes.

»Generalmente se espera que la estimación de tiempo de cada historia de usuario se sitúe entre unas 10 horas y un par de semanas

▪ Estimaciones mayores a dos semanas son indicativo de que la historia es muy compleja y debe ser dividida en varias historias.

»Al momento de implementar las historias, los desarrolladores deben tener la posibilidad de discutirlas con los clientes.

»Si bien el estilo puede ser libre, la historia de usuario debe responder a tres preguntas: ¿Quién se beneficia?, ¿qué se quiere? y ¿cuál es el beneficio?

▪ Como (rol) quiero (algo) para poder (beneficio).

▪ Como usuario registrado deseo loguearme para poder poder empezar a utilizar la aplicación.

»Características

▪ Independientes unas de otras

▪ De ser necesario, combinar las historias dependientes o buscar otra forma de dividir las historias de manera que resulten independientes.

▪ Negociables ▪ La historia en si misma no es lo suficientemente explícita como para considerarse un contrato, la discusión con los usuarios debe permitir esclarecer su alcance y éste debe dejarse explícito bajo la forma de pruebas de validación.

▪ Valoradas por los clientes o usuarios

▪ Los intereses de los clientes y de los usuarios no siempre coinciden, pero en todo caso, cada historia debe ser importante para alguno de ellos más que para el desarrollador.

▪ Estimables

▪ Un resultado de la discusión de una historia de usuario es la estimación del tiempo que tomará completarla. Esto permite estimar el tiempo total del proyecto.

▪ Pequeñas

▪ Las historias muy largas son difíciles de estimar e imponen restricciones sobre la planificación de un desarrollo iterativo. Generalmente se recomienda la consolidación de historias muy cortas en una sola historia.

▪ Verificables

▪ Las historias de usuario cubren requerimientos funcionales, por lo que generalmente son verificables. Cuando sea posible, la verificación debe automatizarse, de manera que pueda ser verificada en cada entrega del proyecto.

»Beneficios

▪ Al ser muy corta, ésta representa requisitos del modelo de negocio que pueden implementarse rápidamente (días o semanas)

▪ Necesitan poco mantenimiento

▪ Mantienen una relación cercana con el cliente

▪ Permite dividir los proyectos en pequeñas entregas

▪ Permite estimar fácilmente el esfuerzo de desarrollo

▪ Es ideal para proyectos con requisitos volátiles o no muy claros

»Limitaciones

▪ Sin pruebas de validación pueden quedar abiertas a distintas interpretaciones haciendo difícil utilizarlas como base para un contrato

▪ Se requiere un contacto permanente con el cliente durante el proyecto lo cual puede ser difícil o costoso

▪ Podría resultar difícil escalar a proyectos grandes

▪ Requiere desarrolladores muy competentes

CLASE 4

MODELO DE PROCESO

»Proceso
▪ Actividades que involucran, restricciones y recursos que producen una determinada salida
▪ 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
▪ Existen principios que orientan sobre las metas de cada actividad
▪ Las restricciones pueden aplicarse a una actividad, recurso o producto.

»Ciclo de vida
▪ Proceso que implica la construcción de un producto

»Ciclo de vida del Software
▪ Describe la vida del producto de software desde su concepción hasta su implementación, entrega utilización y mantenimiento

»Modelos de proceso de software
▪ Es una representación abstracta de un proceso del software.

Términos Equivalentes

▪ Modelo de proceso
▪ Paradigma de software
▪ Ciclo de vida del software

»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.
▪ Cada modelo de proceso prescribe también un “flujo de trabajo”, es decir de que forma los elementos del proceso se interrelacionan entre sí.

»Modelos descriptivos
▪ Descripción en la forma en que se realizan en la realidad

»Ambos modelos deberían ser iguales

»Modelo en Cascada
»Modelo en V
»Modelo de Prototipos
»Desarrollo por fases
▪ Incremental
▪ Iterativo
»El modelo espiral

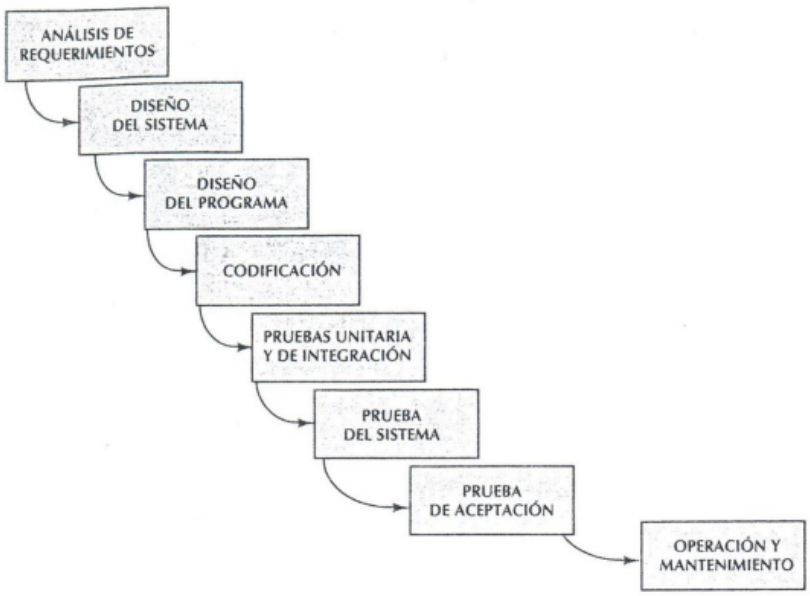
»Modelo en Cascada

▪ Las etapas se representan cayendo en cascada
▪ Cada etapa de desarrollo se debe completar antes que comience la siguiente

▪ Útil para diagramar lo que se necesita hacer
▪ Su simplicidad hace que sea fácil explicarlo a los clientes

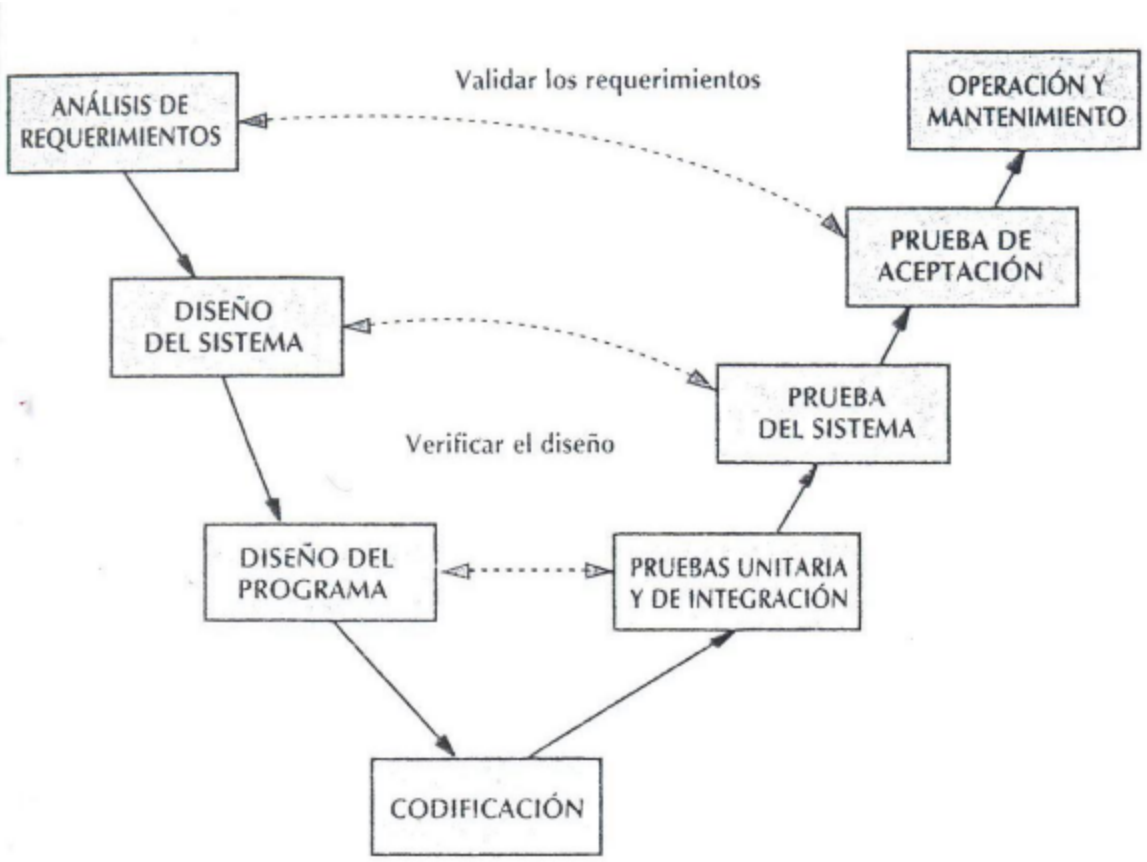
- Dificultades:
  ▪ No existen resultados concretos hasta que todo este terminado.
  ▪ Las fallas más triviales se encuentran al comienzo del período de prueba y las más graves al final.
  ▪ La eliminación de fallas suele ser extremadamente difícil durante las últimas etapas de prueba del sistema.
  ▪ Deriva del mundo del hardware y presenta una visión de manufactura sobre el desarrollo de software.
  ▪ El software no se desarrolla de la misma manera
  ▪ La necesidad de prueba con la computadora aumenta exponencialmente durante las etapas finales de prueba.
  ▪ "CONGELAR" una fase es poco realista.
  ▪ Existen errores, cambios de parecer, cambios en el ambiente.

»Modelo en Cascada



»Modelo en V

▪ Demuestra como 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
▪ 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

▪ Un prototipo es un producto parcialmente desarrollado que permite que clientes y desarrolladores examinen algunos aspectos del sistema propuesto, y decidan si éste es adecuado o correcto para el producto terminado.
▪ Esta es una alternativa de especificación para tratar mejor la incertidumbre, la ambigüedad y la volubilidad de los proyectos reales

»Modelo de Prototipos
Tipos
▪ Evolutivos
▪ El objetivo es obtener el sistema a entregar.
▪ Permite que todo el sistema o alguna de sus partes se construyan rápidamente para comprender o aclarar aspectos y asegurar que el desarrollador, el usuario y el cliente tengan una comprensión unificada tanto de lo que se necesita como de lo que se propone como solución
▪ Descartables
▪ No tiene funcionalidad
▪ Se utilizan herramientas de modelado

▪ Proyectos candidatos
▪ 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

▪ Para asegurar el éxito
▪ Debe ser un sistema con el que se pueda experimentar

▪ Debe ser comparativamente barato (< 10%)
▪ Debe desarrollarse rápidamente
▪ Énfasis en la interfaz de usuario
▪ Equipo de desarrollo reducido
▪ Herramientas y lenguajes adecuados

»Desarrollo por fases

▪ Se desarrolla el sistema de tal manera que puede ser entregado en piezas. Esto implica que existen dos sistemas funcionando en paralelo: el sistema operacional y el sistema en desarrollo.

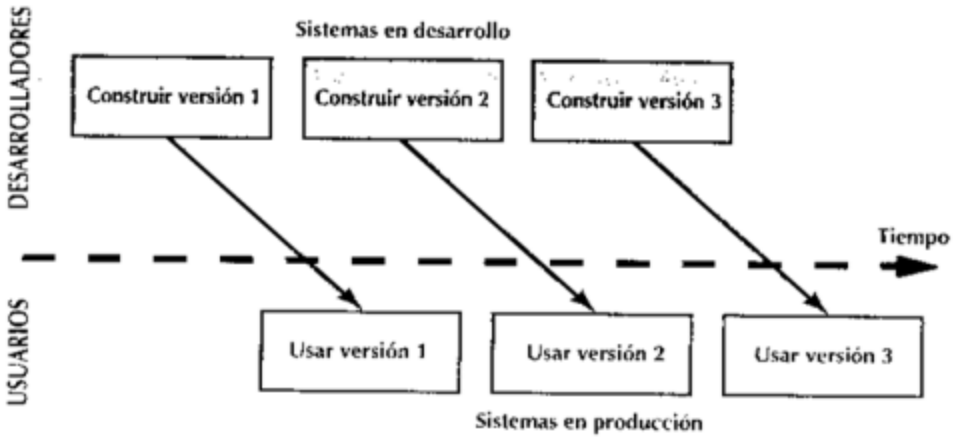
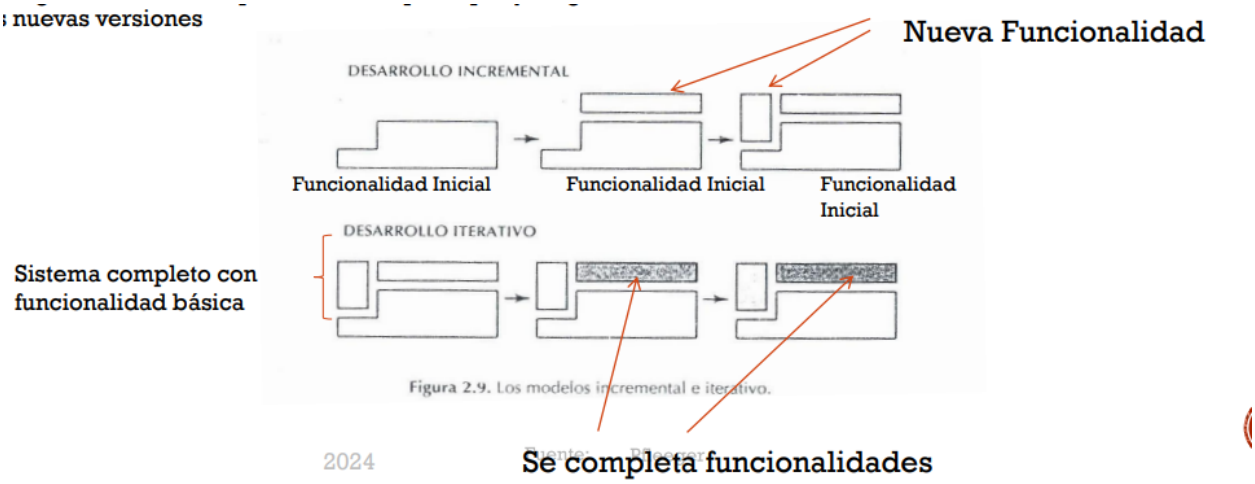


Figura 2.8. El modelo de desarrollo escalonado.

»Desarrollo por fases
▪ Incremental
▪ El sistema es particionado en subsistemas de acuerdo con su funcionalidad. Cada entrega agrega un subsistema.

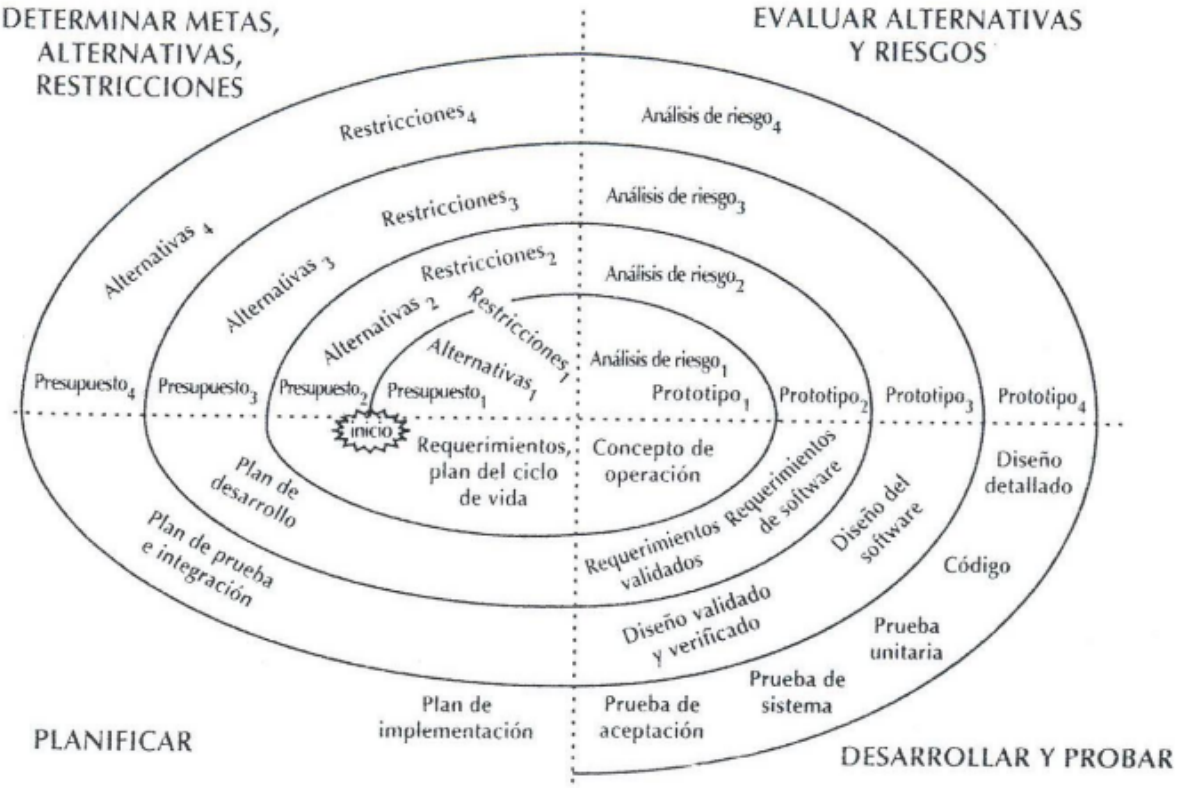
▪ Iterativo

▪ Entrega un sistema completo desde el principio y luego aumenta la funcionalidad de cada subsistema con las nuevas versiones



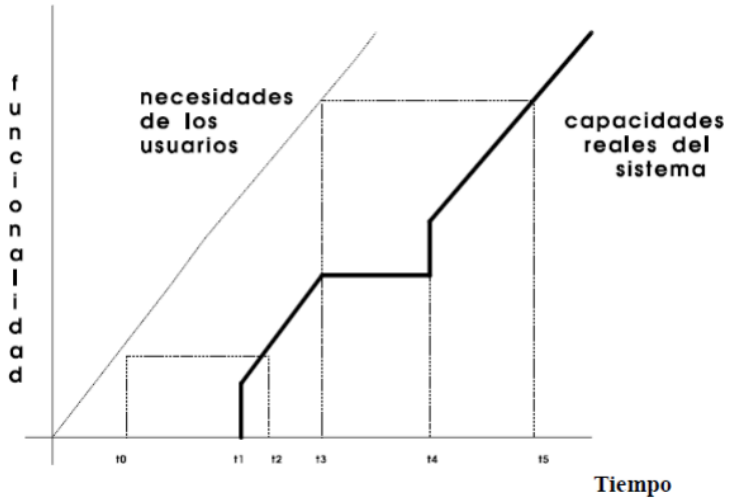
»El modelo espiral (Boehm)

▪ Combina las actividades de desarrollo con la gestión del riesgo
▪ Trata de mejorar los ciclos de vida clásicos y prototipos.
▪ Incorpora objetivos de calidad y gestión de riesgos
▪ Elimina errores y alternativas no atractivas al comienzo
▪ Permite iteraciones, vuelta atrás y finalizaciones rápidas
▪ Cada ciclo empieza identificando:
    ▪ Los objetivos de la porción correspondiente
    ▪ Las alternativas
    ▪ Restricciones
▪ Cada ciclo se completa con una revisión que incluye todo el ciclo anterior y el plan para el siguiente


DETERMINAR METAS, ALTERNATIVAS, RESTRICCIONES
EVALUAR ALTERNATIVAS Y RIESGOS
Restricciones 4
Análisis de riesgo 4
Restricciones 3
Análisis de riesgo 3
Restricciones 2
Análisis de riesgo 2
Restricciones 1
Análisis de riesgo 1
Alternativas 4
Alternativas 3
Alternativas 2
Alternativas 1
Presupuesto 4
Presupuesto 3
Presupuesto 2
Presupuesto 1
Prototipo 1
Prototipo 2
Prototipo 3
Prototipo 4
inicio
Requerimientos, plan del ciclo de vida
Concepto de operación
Plan de desarrollo
Plan de prueba e integración
Requerimientos validados
Requerimientos de software
Diseño del software
Diseño detallado
Código
Diseño validado y verificado
Prueba unitaria
Prueba de sistema
Plan de implementación
Prueba de aceptación
PLANIFICAR
DESARROLLAR Y PROBAR

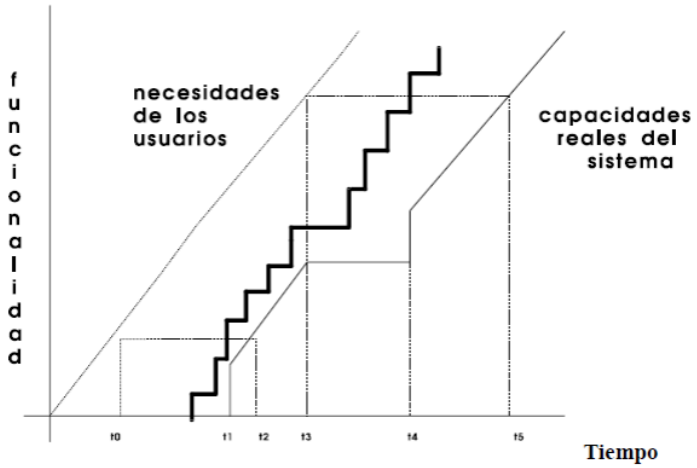
»Análisis Comparativo

Clásico



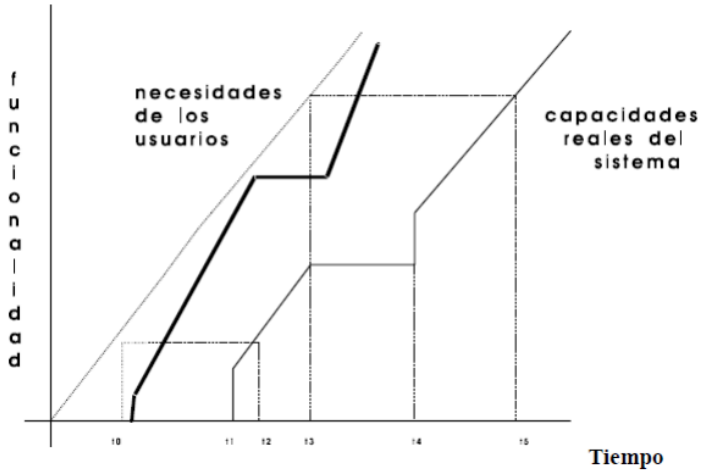
»Análisis Comparativo

Incremental



»Análisis Comparativo

Prototipo Evolutivo



METODOLOGÍAS AGILES

»La ingeniería de software ágil combina una filosofía y un conjunto de directrices de desarrollo.

-La filosofía busca

- La satisfacción del cliente y la entrega temprana del software incremental
- Equipos de proyectos pequeños y con alta motivación
- Métodos informales un mínimo de productos de trabajo de la ingeniería de software
- Una simplicidad en general

-Las directrices resaltan

- La entrega sobre el análisis y el diseño, aunque estas actividades no se descartan
- La comunicación activa y continua entre los desarrolladores y el cliente

»El desarrollo ágil e software son métodos de ingeniería del software basados en el desarrollo iterativo e incremental, donde los requerimientos y soluciones evolucionan mediante la colaboración de grupo auto organizados y multidisciplinares.

»Existen muchos métodos de desarrollo ágil
▪ la mayoría minimiza riesgos desarrollando software en cortos lapsos de tiempo.
▪ El software desarrollado en una unidad de tiempo es llamado una iteración, la cual debe durar de una a cuatro semanas. Cada iteración del ciclo de vida incluye:
▪ Planificación, análisis de requerimientos, diseño, codificación, revisión y documentación.
▪ Una iteración no debe agregar demasiada funcionalidad para justificar el lanzamiento del producto al mercado, pero la meta es tener una «demo» (sin errores) al final de cada iteración.
▪ Al final de cada iteración el equipo vuelve a evaluar las prioridades del proyecto.

»Manifiesto
▪ Estamos descubriendo formas mejores de desarrollar software tanto por nuestra propia experiencia como ayudando a terceros. A través de este trabajo hemos aprendido a valorar:
▪ Individuos e interacciones sobre procesos y herramientas
▪ Software funcionando sobre documentación extensiva
▪ Colaboración con el cliente sobre negociación contractual
▪ Respuesta ante el cambio sobre seguir un plan
▪ Esto es, aunque valoramos los elementos de la derecha, valoramos más los de la izquierda.

»Manifiesto
1.Nuestra principal prioridad es satisfacer al cliente a través de la entrega temprana y continua de software de valor.
2.Son bienvenidos los requisitos cambiantes, incluso si llegan tarde al desarrollo. Los procesos ágiles se doblegan al cambio como ventaja competitiva para el cliente.
3.Entregar con frecuencia software que funcione, en periodos de un par de semanas hasta un par de meses, con preferencia en los periodos breves.
4.Las personas del negocio y los desarrolladores deben trabajar juntos de forma cotidiana a través del proyecto.
5.Construcción de proyectos en torno a individuos motivados, dándoles la oportunidad y el respaldo que necesitan y procurándoles confianza para que realicen la tarea.

6.La forma más eficiente y efectiva de comunicar información de ida y vuelta dentro de un equipo de desarrollo es mediante la conversación cara a cara.

7.El software que funciona es la principal medida del progreso.
8.Los procesos ágiles promueven el desarrollo sostenido. Los patrocinadores, desarrolladores y usuarios deben mantener un ritmo constante de forma indefinida.

9.La atención continua a la excelencia técnica enaltece la agilidad.

10.La simplicidad como arte de maximizar la cantidad de trabajo que no se hace, es esencial.
11.Las mejores arquitecturas, requisitos y diseños emergen de equipos que se autoorganizan.
12.En intervalos regulares, el equipo reflexiona sobre la forma de ser más efectivo y ajusta su conducta en consecuencia.

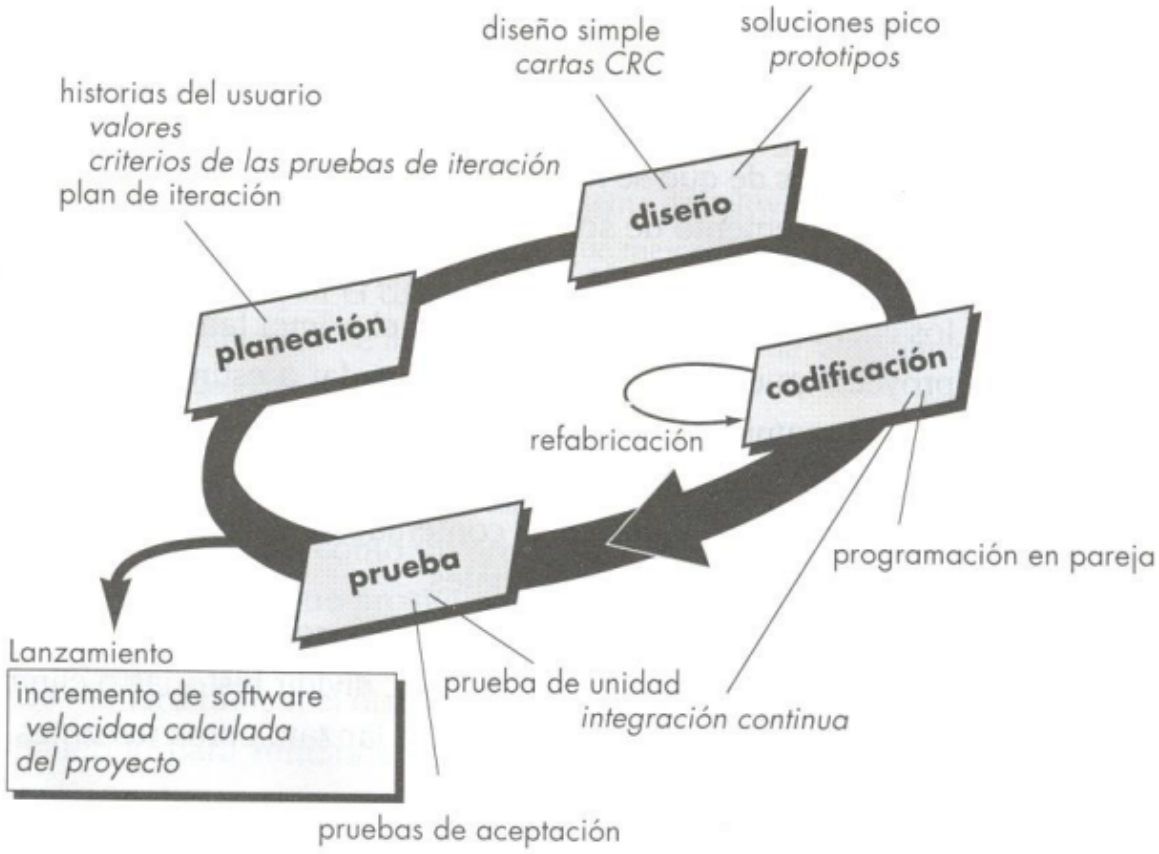
Metodologías Ágiles	Metodologías Tradicionales
Basadas en heurísticas provenientes de prácticas de producción de código	Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo
Especialmente preparados para cambios durante el proyecto	Cierta resistencia a los cambios
Impuestas internamente (por el equipo)	Impuestas externamente
Proceso menos controlado, con pocos principios	Proceso mucho más controlado, con numerosas políticas/normas
No existe contrato tradicional o al menos es bastante flexible	Existe un contrato prefijado
El cliente es parte del equipo de desarrollo	El cliente interactúa con el equipo de desarrollo mediante reuniones
Grupos pequeños (<10 integrantes) y trabajando en el mismo sitio	Grupos grandes y posiblemente distribuidos
Pocos artefactos	Más artefactos
Pocos roles	Más roles
Menos énfasis en la arquitectura del software	La arquitectura del software es esencial y se expresa mediante modelos

Tabla 1. Diferencias entre metodologías ágiles y no ágiles

EXTREME PROGRAMMING

»Es una disciplina de desarrollo de software basado en los valores de la sencillez, la comunicación, la retroalimentación, la valentía y el respeto
»Su acción consiste en llevar a todo el equipo reunido en la presencia de prácticas simples, con suficiente información para que el equipo para ver dónde están y para ajustar las prácticas a su situación particular.
»En la programación extrema, cada colaborador del proyecto es una parte integral del " Equipo".
▪ Las formas del equipo en torno a un representante de la empresa llama "el Cliente", que se sienta con el equipo y trabaja con ellos todos los días.

▪ Desarrollo iterativo e incremental: pequeñas mejoras, unas tras otras.
▪ Pruebas unitarias continuas, frecuentemente repetidas y automatizadas, incluyendo pruebas de regresión.
▪ Programación en parejas
▪ Frecuente integración del equipo de programación con el cliente o usuario.
▪ Corrección de todos los errores antes de añadir nueva funcionalidad.
▪ Refactorización del código
▪ Propiedad del código compartida
▪ Simplicidad en el código



SCRUM

»Scrum es un proceso en el que se aplican, de manera regular, un conjunto de mejores prácticas para trabajar en equipo y obtener el mejor resultado posible de un proyecto.
»Estas prácticas se apoyan unas a otras y su selección tiene origen en un estudio de la manera de trabajar de equipos altamente productivos.
»En Scrum se realizan entregas parciales y regulares del resultado final del proyecto, priorizadas por el beneficio que aportan al receptor del proyecto

»Eliminar el desperdicio
▪ No generar artefactos, ni perder el tiempo haciendo cosas que no le suman valor al cliente.
»Construir la calidad con el producto
▪ La idea es 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, esperar hasta ese momento, ya que uno tiene mas
información a medida que va pasando el tiempo. Si se puede esperar, mejor.
»Entregar rápido
▪ Debe ser una de las ventajas competitivas más importantes.
»Respetar a las personas
▪ La gente trabaja mejor cuando se encuentra en un ambiente que la motive y se sienta respetada.
»Optimizar el todo
▪ Optimizar todo el proceso, ya que el proceso es una unidad, y para lograr tener éxito y avanzar, hay que
tratarlo como tal.

»Eliminar el desperdicio
▪ No generar artefactos, ni perder el tiempo haciendo cosas que no le suman valor al cliente.
»Construir la calidad con el producto
▪ La idea es 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, esperar hasta ese momento, ya que uno tiene mas
información a medida que va pasando el tiempo. Si se puede esperar, mejor.
»Entregar rápido
▪ Debe ser una de las ventajas competitivas más importantes.
»Respetar a las personas
▪ La gente trabaja mejor cuando se encuentra en un ambiente que la motive y se sienta respetada.
»Optimizar el todo
▪ Optimizar todo el proceso, ya que el proceso es una unidad, y para lograr tener éxito y avanzar,

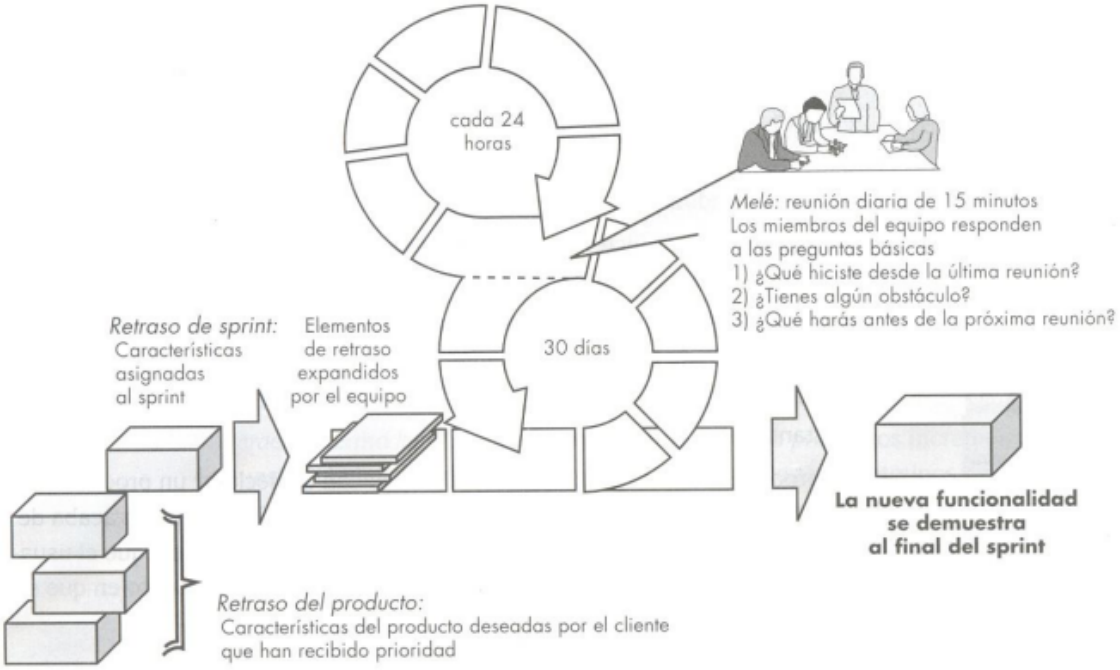
hay que
tratarlo como tal.

»El Product Owner (Propietario)
▪ Conoce y marca las prioridades del proyecto o producto.
»El Scrum Master (Jefe)
▪ Es la persona que asegura el seguimiento de la metodología guiando las reuniones y ayudando al equipo ante cualquier problema que pueda aparecer. Su responsabilidad es entre otras, la de hacer de paraguas ante las presiones externas.
»El Scrum Team (Equipo)
▪ Son las personas responsables de implementar la funcionalidad o funcionalidades elegidas por el Product Owner.
»Los Usuarios o Clientes
▪ Son los beneficiarios finales del producto, y son quienes viendo los progresos, pueden aportar ideas, sugerencias o necesidades

»Product Backlog
▪ Es la lista maestra que contiene toda la funcionalidad deseada en el producto. La característica más importante es que la funcionalidad se encuentra ordenada por un orden de prioridad.
»Sprint Backlog
▪ Es la lista que contiene toda la funcionalidad que el equipo se comprometió a desarrollar durante un Sprint determinado.
»Burndown Chart
▪ Muestra un acumulativo del trabajo hecho, día-a-día.
»Entre otros…

»Scrum es iterativo e incremental
▪ Se busca poder atacar todos los problemas que surgen durante el desarrollo del proyecto
▪ El nombre Scrum se debe a que durante los Sprints, lo que serían las fases de desarrollo, se solapan, de manera que no es un proceso de cascada por cada iteración, si no que tenemos todas éstas etapas juntas que se ejecutan una y otra vez, hasta que se crea suficiente.

▪ Este solapamiento de fases se puede asemejar a un scrum de rugby, en el cual todos los jugadores (o roles, en nuestro caso), trabajan juntos para lograr un objetivo.



»Scrum está pensado para ser aplicado en proyectos en donde el caos es una constante, aquellos proyectos en los que tenemos requerimientos dinámicos, y que tenemos que implementar tecnología de punta.
»Esos proyectos difíciles, que con los enfoques tradicionales se hace imposible llegar a buen puerto.

MODELO DE PROCESOS - DESARROLLO DE SOFTWARE DIRIGIDO POR MODELOS

»Model Driven Development (MDD) promueve enfatizar los siguientes puntos claves:

▪ Mayor nivel de abstracción en la especificación tanto del problema a resolver como 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 como medio 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.

» PIM - Platform Independent Model

▪ “Un modelo de un sistema que no contiene información acerca de la plataforma o la tecnología que es usada para implementarlo”

» PSM - Platform Specific Model

▪ “Un modelo de un sistema que incluye información acerca de la tecnología especifica que se usará para su implementación sobre una plataforma especifica”

» Transformación de modelos

▪ “Especifica el proceso de conversión de un modelo en otro modelo del mismo sistema.”

»Transformación

▪ En general, se puede decir que una definición de transformación consiste en una colección de reglas, las cuales son especificaciones no ambiguas de las formas en que un modelo (o parte de él) puede ser usado para crear otro modelo (o parte de él).

▪ El patrón MDD es normalmente utilizado sucesivas veces para producir una sucesión de transformaciones.

»Beneficios de MDD.

▪ 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.

CLASE 5

¿QUE ES LA CALIDAD?

»Calidad es un concepto manejado con bastante frecuencia en la actualidad, pero a su vez, su significado es percibido de distintas maneras.
»Al hablar de bienes y/o servicios de calidad, la gente se refiere normalmente a bienes de lujo o excelentes con precios elevados.
»Su significado sigue siendo ambiguo y muchas veces su uso depende de lo que cada uno entiende por calidad, por lo cual es importante comenzar a unificar su definición.

»Las principales normas internacionales definen la calidad como :
»“El grado en el que un conjunto de características inherentes cumple con los requisitos “ (ISO 9000)
»“Conjunto de propiedades o características de un producto o servicio que le confieren aptitud para satisfacer unas necesidades expresadas o implicitas” (ISO 8402)

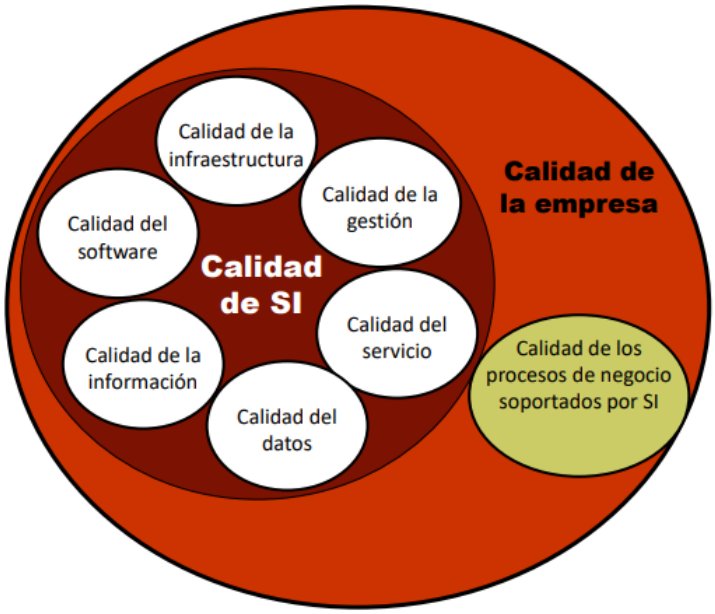
CALIDAD DE LOS SISTEMAS DE INFORMACIÓN

»La importancia de los sistemas de información (SI) en la actualidad hace necesario que las empresas de tecnología hagan mucho hincapié en los estándares de calidad.

» Stylianou y Kumar plantean que se debe apreciar la calidad desde un todo, donde cada parte que la componen debe tener su análisis de calidad.

Visión holística de la calidad

Stylianou y Kumar (2000)



COMPONENTES

»Calidad de la Infraestructura

▪ incluye, por ejemplo, la calidad de las redes, y sistemas de software.

»Calidad de Software

▪ de las aplicaciones de software construidas, o mantenidas, o con el apoyo de IS.

»Calidad de Datos

▪ Que ingresan en el sistema de información.

»Calidad de Información

▪ está relacionada con la calidad de los datos.

»Calidad de gestión

▪ incluye el presupuestó , planificación y programación.

»Calidad de servicio
▪ incluye los procesos de atención al cliente

CALIDAD DE SOFTWARE

»La calidad del software se ha mejorado significativamente en esto últimos años, en particular por una mayor conciencia de la importancia de la gestión de la calidad y la adopción de técnicas de gestión de la calidad para desarrollo en la industria del software

»Se divide en
▪ Calidad del producto obtenido
▪ Calidad del proceso de desarrollo

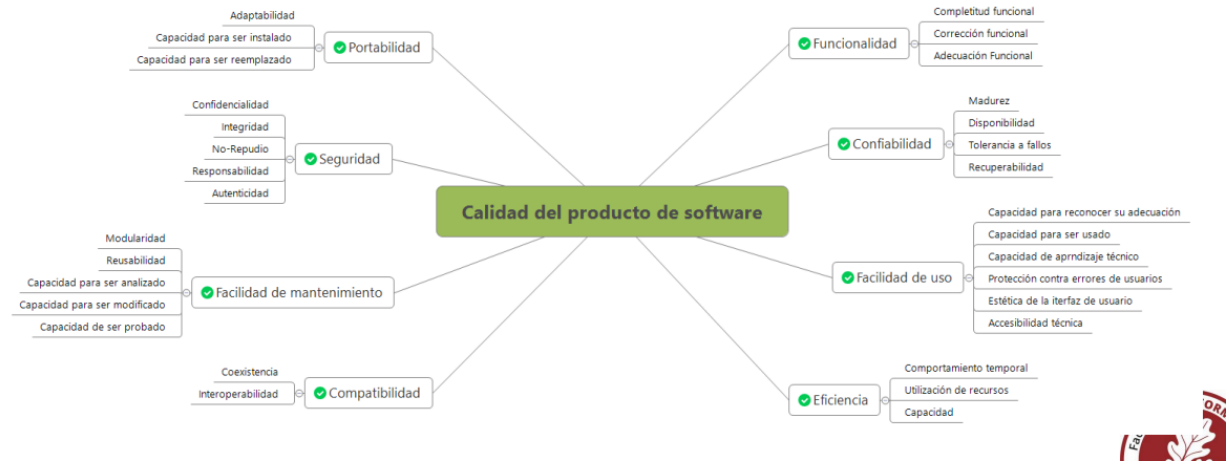
CALIDAD DEL PRODUCTO Y PROCESO

»Producto
▪ Un producto es un bien tangible que es el resultado de un proceso.
▪ Aunque el software tiene aspectos intangibles, un producto software es sin embargo un bien en sí mismo e incluye sus documentos asociados.
▪ La estandarización del producto define las propiedades que debe satisfacer el producto software resultante.

»Proceso
▪ La estandarización del proceso define la manera de desarrollar el producto software.

Sin un buen proceso de desarrollo es casi imposible obtener un buen producto.

MODELO DE CALIDAD SQUARE ISO/IEC 25010



CALIDAD DE LOS DATOS ISO/IEC 25012

» La norma entiende por calidad de datos:

▪ La capacidad de las características de los datos de satisfacer necesidades explícitas e implícitas bajo determinadas condiciones de uso.

»Los clasifica estas características de calidad considerando dos puntos de vista:

▪ Inherente

▪ Capacidad de las características de los datos de tener el potencial intrínseco para satisfacer las necesidades explícitas e implícitas

▪ Este punto de vista está más relacionado con los aspectos del dominio gestionados por los expertos del negocio.

▪ Dependiente del sistema:

▪ Capacidad del sistema informático de alcanzar y preservar la calidad de los datos cuando los datos se utilizan en determinadas condiciones

▪ Este punto de vista suele ser responsabilidad de los técnicos del sistema.

SQUARE - PROCESO DE EVALUACIÓN - ISO/IEC 25040

1. Establecer los requisitos de la evaluación
    a. Definir el rigor de la evaluación
    b. Identificar las partes del producto que se deben evaluar
    c. Obtener los requisitos de calidad del producto
    d. Establecer el propósito de la evaluación
2. Especificar la evaluación
    a. Definir los criterios de decisión de la evaluación
    b. Definir los criterios de decisión para las métricas
    c. Seleccionar los módulos de evaluación
3. Diseñar la evaluación
    a. Planificar las actividades de la evaluación
4. Ejecutar la evaluación

    Realizar las mediciones

    Aplicar los criterios de decisión para las métricas

    Aplicar los criterios de decisión de la evaluación
5. Finalizar la evaluación

    Revisar los resultados de la evaluación

    Crear el informe de evaluación

    Revisar la calidad de la evaluación y obtener feedback

    Tratar los datos de la evaluación

MODELO DE CALIDAD DE PROCESO SOFTWARE
CMM - CMMI

»En diciembre de 2000, el SEI publicó un nuevo modelo, el CMMI o "Modelo de Capacidad y Madurez - Integración", con el objetivo de realizar algunas mejoras respecto al SW-CMM (e integrarlo con el SE-CMM y el IPD-CMM, que pasaron a ser considerados como "obsoletos".

»Incluye cuatro disciplinas, Software, Ingeniería de sistemas , Desarrollo integrado de procesos y productos y Gestión de proveedores .
»A su vez incorpora una nueva representación, "Continua", la que permite evaluar el nivel en cada área independientemente.

»El SEI ha desarrollado también un nuevo método de evaluación de las organizaciones según CMMI denominado SCAMPI.