Introducción a la Ingienería del software

Características del software

- El software es un **elemento lógico**, no físico.
- El software **se desarrolla**, no se fabrica.
- El software **no se estropea**, pero se deteriora.

Problemática detectada

- Proyectos **fuera de plazo y de presupuesto**.
- **Excesiva dependencia** de los desarrolladores.
- **Falta de control** del desarollo del proyecto.
- **Escasa integración** de las diferentes fases del desarrollo.
- **Escaso control de calidad** del producto.
- **Escasa documentación** actualizada de los proyectos.
- No utilizar una metodología formal.

Principios de la Ingienería del software

- Abstracción
    - Permite parcelar la complejidad. Por ello se olvidan aspectos irrelevantes del sistema y se potencias los fundamentales.
- Encapsulamiento u Ocultación de la información
    - Esconder todos los detalles que no afecten a otros módulos, definiendo interfaces estrictos que sirvan de interacción entre los distintos modelos.
- Modularidad
    - Sirve para parcelar la solución en módulos independientes con fuerte cohesión interna.
- Localización
    - Deben estar agrupados todos aquellos elementos que están afectados por un mismo hecho.
- Uniformidad
    - Todos los módulos deben tener una notación similar.
- Completitud
    - Deben estar desarrollados todos los aspectos del sistema.
- Validación y Verificabilidad
    - El producto debe ser fácilmente validable y verificable.
- Uniformidad

    - Todos los módulos deben tener una notación similar.

- Completitud
    - Deben estar desarrollados todos los aspectos del sistema.
- Validación y Verificabilidad
    - El producto debe ser fácilmente validable y verificable.

Fases

Fase de Definición

- Análisis de requisitos
- Extraer los requisitos de un producto software es la primera fase para crearlo.
- El resultado del análisis de requisitos con el cliente se plasma en el documento de Especificación de Requisitos.
- La captura, análisis y especificación de requisitos, es una parte crucial de la que depende en gran medida el logro de los objetivos finales.
- Factor clave en el éxito o fracaso de los proyectos

Fase de desarrollo

- Se diseñan las estructuras de los datos y los programas.
- Se escriben y documentan los programas,
- y se prueba el software construido.

Fase de mantenimiento

- Comienza una vez construido el sistema, cuando se empieza a utilizar.
- Se centra en el **cambio**.
- El software es sometido a reparaciones y modificaciones cada vez que se detecta un fallo o se necesita cubrir una nueva necesidad de los usuarios.
- En esta fase recae el mayor porcentaje del coste de un sistema.

Tipos de mantenimiento

- **Correctivo**: un programa no realiza correctamente la aplicación para la que ha sido diseñado, y, por tanto, debe ser modificado.
- **Perfectivo**: modificaciones a los programas para conseguir mayor adecuación a los requisitos, mayor eficiencia, o simplemente recoger nuevas funcionalidades no expresadas en la fase de definición del sistema.
- **Adaptativo**: Adaptar los programas para acomodarlos a los cambios de su entorno externo (modificacioens en la legislación, CPU, SO, las reglas de negocio, etc.)
- **Preventivo**: El softwarese deteriora con los cambios, y este tipo de mantenimiento hace cambios en los programas más facilmente **(Reingeniería del software).**

Capas

Procesos

- El funcionamiento de la IS es la capa de proceso.
- Define un marco de trabajo:
    - Identifica todas las actividades y tareas de la IS
    - Define el flujo de trabajo entre las actividades y tareas
    - Identifica los productos de trabajo que se producen
    - Especifia los puntos de control de calidad requeridos

Métodos

- Proporciona el "cómo" y cubre las actividades de la ingienería fundamentales.
- Se centra en las actividades técnicas que deben realizar para conseguir las tareas de ingienería.

Herramientas

- Proporciona soporte a las capas de proceso y métodos
- Automatización de algunas de las actividades manuales:
    - Actividades de gestión de proyectos
    - Métodos técnicos usados en la ingienería del software
    - soporte de sistemas general
    - Marcos de trabajo para otras herramientas.
- La automatización ayuda a eliminar el tedio del trabajo, reduce las posibilidades de errores y hace más fácil usar buenas prácticas de IS

Proyecto de software

Se puede definir un proyecto de software como una actividad humana que transforma unos requisitos de un cliente/usuario en un producto software entregable. Para dicha transformación se aplica un proceso de desarrollo o ciclo de desarrollo establecido.

Proceso de software

Conjunto coherente de políticas, estructuras organizacionales, tecnologías, procedimientos y artefactos que son necesarios para concebir, desarrollar, instalar y mantener un producto de software.

Ciclos de Vida

Definiciones

- "Aproximación lógica a la adquisición, suministro, desarrollo, explotación y mantenimiento del software"
- "Marco de referencia que contiene los procesos, las actividades y las tareas involucradas en el desarrollo, la explotación y el mantenimiento de un producto de software, abarcando la vida del sistema desde la definición de los requisitos hasta la finalización de su uso"

Características

Un ciclo de vida debe determinar el **orden de las fases** del Proceso de Software y establecer **criterios de transición** para pasar de una fase a la siguiente.

Ciclo de Vida vs ciclo de Desarrollo

Ciclo de vida ≠ Ciclo de vida de desarrollo

- Ciclo de vida: Toda la vida del sistema, desde la concepción hasta el fin de uso.
- Ciclo de vida de desarollo: Desde el análisis hasta la entrega al usuario.

Codificación directa

- También llamado **Code and Fix**
- Sinónimo de **arte** en el Desarrollo de Software
- Antónimo de ingienería en el Desarrollo de Software
- **No es recomendable** para proyectos poco más grande que "enanos"

Cascada

También conocido como **Waterfall**, nace en los años 70 como alternativa al Code and Fix, que provocó la crisis del software. Sigue un gran número de metodologías y es **el más conocido, estudiado, difundido y empleado**. Pone especial énfasis en la **realización temprana de actividades** de definición de los requisitos y de la docimentación de análisis y diseño como paso previo a la Codificación. **Secuencia lineal** de las siguientes actividades generales: Analizar, Diseñar, Codificar, Probar e Implementar. Cada fase genera productos de salida que servirán a la siguiente fase para desarrollar la actividad de la misma como productos de entrada.

Ventajas

- Conforma un **marco de referencia para asignar todas las actividades de desarrollo software.
- Es un método muy estructurado que establece pautas de trabajo muy claras.
- Facilita mucho la **coordinación** de los recursos implicados.
- Facilita la **disposición de hitos de seguimiento / control** en el desarrollo de proyectos.
- Facilita la **estimación y el seguimiento / control** del progreso de las actividades.
- Facilita la **detección de desviaciones** y la realización de acciones correctivas.
- Proporciona productos entregables intermedios que conforman el producto final.

Inconvenientes

- Comienza estableciendo **todos** los requisitos del sistema a desarrollar, aunque muchas veces esto no es viable.
- **Gran rigidez**: cada actividad es prerrequisito de las que le siguen.
- Los posibles **problemas se detectan tarde**, aunque se incluyan actividades de verificación y validación.
- Los únicos productos parciales aprovechables están en forma de documentos: **nada está hecho hasta que todo está hecho.**

Otros modelos de Ciclo de Vida

Ciclo de vida en V

Es un modelo similar al de Cascada, aunque "mejorado". Tiene dos tiempos claramente marcados: **rama descendente**, actividades de desarrollo; **rama ascendente**, actividades de prueba. Ambas ramas se comunican en materia de pruebas, cada fase de la rama descendente tiene su homóloga en la rama ascendente para las actividades de prueba correspondientes.

Ventajas

- Conforma un marco de referencia para asignar todas las actividades de desarrollo software, incluyendo las actividades V&V de lo que se hace en las sucesicas etapas.
- Favorece la consideración de las pruebas lo antes posible (comunicación horizontal entre las dos ramas de la V)

Inconvenientes

- Al igual que en Cascada, se comienza estableciendo todos los requisitos del sistema a desarrollar.
- Posee una gran rigidez, aunque menos que el de Cascada por haber comunicación horizontal.
- Los únicos productos (parciales) aprovechables están en forma de documentos.

Prototipado

El proceso básico del prototipado involucra las siguientes tres fases en bucle:

- Escuchar al usuario/cliente.
- Construir y revisar el prototipo.
- El usuario/cliente prueba el prototipo Su sentido es análogo al que tienen otras ingienerías, el prototipado o construcción de un prototipo es un proceso que facilita al desarrollador la creación de un modelo del software a desarrollar.

Tipos de prototipos

Desechables: nos sirve para eliminar dudas sobre lo que realmente quiere el cliente, además de desarrollar la interfáz que más le convenga al cliente. Evolutivos: es un modelo parcialmente construido que puede pasar de ser prototipo a ser software , pero no tiene una buena documentación y calidad, hay que conseguir que tenga los criterios mínimos de calidad.

Ventajas

El prototipo es un mecanismo ideal para extraer requisitos cuando no están claros, incluso por parte del usuario.

Inconvenientes

Existe una clara tendencia del usuario/cliente a creer que el trabajo ya está hecho y que en breve dispondrá de un sistema funcional. El desarrollador toma necesariamente decisiones de implementación simplemente porque le son conocidas y le permiten desarrollar rápidamente el prototipo.

Incremental

Es el proceso de construir una implementación parcial del sistema global y posteriormente ir aumentando la funcionabilidad del sistema. Es la aplicación reiterada de varias secuencias basadas en el modelo Cascada. Cada aplicación del ciclo constituye un incremento del software, cada incremento resulta en un producto operativo, cada incremento puede ser: el refinamiento de un incremento anterior o el desarrollo de una nueva funcionalidad no incorporada en incrementos anteriores; en el primer incremento de un sistema se debe abordar el núcleo esencial del sistema, en incrementos posteriores se abordarán funciones suplementarias. El usuario/cliente evalúa el resultado de un incremento y se elabora un plan para el incremento siguiente. El proceso se repite hasta que se elabore un producto completo. Al seguir un desarrollo incremental, el software debe construirse de tal forma que facilite la incorporación de nuevos requisitos.

Ventajas

La dotación de personal no tiene que estar disponible para una implementación completa. Permite la obtención de incrementos operativos a lo largo del proceso de desarrollo, frente a modelos tradicionales basados en el de Cascada. Reduce la posibilidad de que los requisitos del usuario/cliente cambien durante el desarrollo. Mayor flexibilidad ante más funcionalidades. La calidad y estabilidad del software progresan con las iteraciones, y hay oportunidad para la mejora. Ayuda en proyectos que utilizan tecnologías nuevas o en fase de aprendizaje por parte de la organización.

Incremental

Puede ser muy complejo definir el núcleo operativo para lograr el primer incremento: un software con capacidades suficientes como para ser operativo y que facilite la incorporación de nuevos requisitos. Puede resultar complicado establecer y priorizar las funcionalidades que se mejoran o incorporan a los sucesivos incrementos. Las soluciones de incrementos anteriores pueden no ser válidas para incrementos posteriores. No es útil en proyectos cortos en duración o ámbito ni en los que la funcionalidad a implementar esté perfectamente definida, ya que el coste extra de planifiación no merece la pena.

Espiral

Propuesto por Barry W. Boehm en 1988, el modelo Espiral de proceso evolutivo permite combinar la naturaleza interactiva de construcción de propositos con los aspectos controlados y sistemáticos del modelo en Cascada, además pone especial énfasis en el análisis de los riesgos para reducir su impacto. El software se desarrolla en una serie de versiones incrementales. Durante las primeras iteraciones, la versión incremental será un modelo en papel o un prototipo. Durante las últimas iteraciones, se producen versiones cada vez más completas del sistema. La idea básica es que cada fase (vuelta de espiral) establece los siguientes pasos:

- Determinación de objetivos, alternativas y restricciones.
- Evaluación de alternativas.
- Desarrollo del siguiente nivel de producto.
- Planificación de la siguiente fase.

Ventajas

Permite adaptar el proceso de desarrollo a las necesidades cambiantes del proyecto y al conocimiento que se va adquiriendo. Permite el manejo de prototipos , enlazándolo con el análisis de riesgos. Gestiona

explícitamente los riesgos, permitirá reducirlos antes de que se conviertan en problemas.

Inconvenientes

Requiere de una considerable habilidad para la consideración del riesgo.

Ventajas

Permite adaptar el proceso de desarrollo a las necesidades cambiantes del proyecto y al conocimiento que se va adquiriendo. Permite el manejo de prototipos , enlazándolo con el análisis de riesgos. Gestiona explícitamente los riesgos, permitirá reducirlos antes de que se conviertan en problemas.

Inconvenientes

Requiere de una considerable habilidad para la consideración del riesgo.

Análisis de requisitos

Qué es un Requisito

Un **requisito de software puede ser** definido como una condición o necesidad de un usuario para resolver un problema o alcanzar un objetivo. Un **requisito** es una característica que el sistema **debe** tener o es una restricción que el sistema **debe** satisfacer para ser aceptada por el cliente.

Especificación de Requisitos

El documento de Especificación de Requisitos utiliza estándares de estructuración de especificaciones de requisitos, aclara el objetivo globar a cumplir por el sistema, emplea descripciones textuales y gráficas, ordena y agrupa los requisitos de forma lógica, relaciona unos requisitos con otros para facilitar su entendimiento y relaciona los requistos con otros elementos. Los pasos para identificar y definir los requisitos de un producto son:

- Observar y entender el trabajo desde **el punto de vista del usuario**.
- Interpretar el trabajo del usuario y la forma en la que él lo describe.
- "Inventar" mejores formas de hacer el trabajo.
- Plasmar estos resultados en forma de especificación de requisitos.

Problemas al definir los requisitos

Los requisitos no son obvios y vienen de muchas fuentes, pueden ser difíciles de expresar en palabras, la cantidad de requisitos en un proyecto puede ser difícil de manejar, un requisito puede cambiar a lo largo del ciclo de desarrollo. Se tiende a recordar lo excepcional y olvidar lo rutinario, los usuarios tienen distinto vocabulario que los desarrolladores.

Evolución de los Requisitos

Porque al analizar el problema no se hacen las preguntas correctas a las personas correctas, porque cambió el problema que se estaba resolviendo, porque los usuarios cambiaros su forma de pensar o sus percepciones, porque cambió el mercado en el que se desenvuelve el negocio.

Tipos de Requisitos

- Requisitos de Negocio
- Requisitos de Usuario
- Requisitos del Sistema/Software

Requisitos de Negocio

Representan los objetivos generales que quieren alcanzar con el desarrollo del software. Descpción de alto nievel de lo que el sistema debe de hacer. No describen detalles técnicos ni funcionalidades específicas. Se relacionan con las metas estratégicas de la empresa. Son la base de los requisitos de usuario.

Requisitos de Usuario

Representan las necesidades o metas que tienen los usuarios acerca del sistema. Son más específicos que los de negocio. Se enfocan en las tareas y expectativas del usuario.

Requisitos del sistema

Detallan las especificaciones técnicas que el software debe cumplir. Son de bajo nivel y específicos. Incluyen detalles técnicos como arquitectura, rendimiento, seguridad y estándares a seguir. Se derivan directamente los requisitos de usuario y de negocio.

Requisitos funcionales

Describen lo que el sistema debe hacer. Describen la interacción entre el sistema y su entorno independientemente de su implementación. El entorno incluye al usuario y cualquier otro sistema externo que interactúa con el sistema.

Requisitos no funcionales

Son aquellos que no hacen referencia directamente a las funciones específicas que proporciona el sistema, sino a las propiedades emergentes del mismo como son rendimineto, usabilidad, seguridad o escalabilidad. Los requisitos no funcionales son las **cualidades que debe tener el producto**. EStps requisitos hacen que el producto sea atractivo, útil, rápido, fiable o seguro. Se pueden clasificar de diferentes formas:

- Requisitos de producto
- Requisitos organizacionales
- Requisitos externos

Características de la descripción de un requisito

- Completo
- Correcto
- Realizable
- Necesario
- Priorizable
- No ambiguo
- Verificable

- Consistente
- Modificable
- Trazable

Técnicas de recogida de requisitos

- Reuniones / Entrevistas
- Cuestionarios y encuestas
- Braimstorming
- Casos de uso
- Prototipos
- Escenarios

Modelo de casos de uso

Introducción

Un diagrama de casos de uso muestra la relación entre los actores y los casos de uso del sistema, representa la funcionalidad que ofrece el sistema en lo que se refiere a su interacción externa, centrándose en lo que debe hacerse y no en la manera de hacerlo. Deben evitarse expresiones imprecisas, se busca la sencilled y la claridad. Elementos que puden participar en un diagrama de casos de uso:

- Actores
- Casos de uso
- Relaciones

Actores

Un actor es una entidad externa al sistema que realiza algún tipo de interacción con el mismo, se representa mediante una figura humana; esta representación sirve tanto para actores que son personas como para otro tipo de actores no humanos. Se deben distinguir entre actores:

- Principales: aquellos para los que se construye el sistema
- Secundarios: aquellos que dan soporte al sistema.

Identificación de Actores

¿Qué usuarios están soportados por el sistema para desarollar su trabajo? ¿Qué usuarios ejecutan las funciones principales del sistema? ¿Qué usuarios desempeñan funciones secundarias, como mantenimiento y administración? ¿El sistema interactúa con hardware externo o software?

Casos de uso

Un caso de uso es una descripción de la secuencia de interacciones que se producen entre un actor y el sistema, cuando el actor usa el sistema para llevar a cabo una tarea determinada. Expresa una unidad coherente de funcionalidad, y se representa mediante una elipse con el nombre del caso de uso en su interior. El nombre del caso de uso debe reflejar la tarea específica que el actor desea llevar a cabo usando el sistema.

Identificación de casos de uso

Capturar el comportamiento deseado del sistema en desarrollo, sin tener que especificar cómo se implementa este comportamineto. Los casos de uso proporcionan un medio para que los desarrolladores, los usuarios finales del sistema y los expertos del dominio lleguen a una comprensión común del sistema. Un escenario de la instancia de un caso de uso. Los casos de uso no deben ser excesivamente genéricos ni demsaisado específicos. Requisitos funcionales del sistema.

Relaciones

En un diagrama de casos de uso pueden aparecer las siguientes relaciones:

- Asociación (----------)
- Extiende (- - - - - ->) <<extiende>>
- Generalización (----------|>)
- Inclusión (- - - - - ->) <<incluye>>

Identificar relaciones entre Casos de uso

Inclusión <<incluye>>

- Indica que el flujo de eventos del caso de uso base se incluye en el comportamiento del otro caso de uso.
- Factoriza comportamiento común.
- Solamente se hace cuando la parte común es utilizada por otro caso de uso o cuando es utilizada por otro actor.

Extensión <<extiende>>

- Un caso de uso extiende otro caso de uso, si el caso de uso extendido incluye el comportamiento del otro bajo ciertas condiciones.
- Se utiliza para modelar la parte de un caso de uso que el usuario puede ver como comportamineto opcional del sistema.
- Se separa el comportamiento opcional del obligatirio

Generalización

- Cuando algunos casos de uso tienen algo en común y puede ser abstraído a otro, mucho más general.
- El caso de uso hijo hereda el comportamiento y el significado del caso de uso padre.
- El hijo puede añadir o redefinir el comportamineto y el significado del padre.
- El hijo puede ser colocado en cualquier lugar donde aparezca el padre. algunos casos de uso tienen algo en común y puede ser abstraído a otro, mucho más general.
- El caso de uso hijo hereda el comportamiento y el significado del caso de uso padre.
- El hijo puede añadir o redefinir el comportamineto y el significado del padre.
- El hijo puede ser colocado en cualquier lugar donde aparezca el padre.

Construcción

Identificación de casos de uso:

- Proceso iterativo en el que se van descubriendo escenarios desde el punto de vista del usuario.
- Pueden utilizarse diversas técnicas: observación, entrevista estructurada, etc.

Descripción de los casos de uso:

- Descripción inicial: se describe en un par de frases las principales etapas de cada caso de uso; se distingue un escenario principal y se identifican los escenarios alternativos y excepcioens.
- Se establece un proceso iterativo en el cual los casos de uso se amplían, profundizándose en su descripción, buscando etapas comunes y alternativas.

El proceso de Análisis de Requisitos con Casos de Uso

1. Elaborar el diagrama de casos de uso de trazo grueso: 1.1. Identificar los actores 1.2. Identificar los principales casos de uso de cada actor. 1.3. Identificar los nuevos casos de uso a partir de los existentes.
2. Crear descripciones de casos de uso de trazo grueso.
3. Definir prioridades y seleccionar casos de uso de cada incremento
4. Se abordan los diferentes incrementos: 4.1. Se profundiza en el conocimiento de los casos de uso involucrados. 4.2. Se refina el diagrama de casos de uso. 4.3. Se escriben los casos de trazo fino.

Análisis de Sistemas Software

Objetivos del análisis

- Especificar las funciones del software
- Especificar la información que se maneja.
- Especificar las interfaces con otros elementos del sistema.
- Establecer restricciones que debe cumplir el software.

Resultados del análisis

El análisis ha de plasmar los requisitos del usuario en modelos del sistema, estos modelos dependerán del enfoque seguido (estructurado o objetual).

Modelos de análisis

¿Por qué es necesario hacer modelos?

Principalmente por las siguientes 3 razones:

- Permiten concentrarse en las propiedades importantes del sistema.
- Favorecen la comunicación con el usuario.
- Permiten verificar que el analista ha comprendido el problema / entorno a abordar.

¿Qué aportan los modelos?

Entre sus aportaciones destacan:

- Reflejar lo que requiere el usuario del sistema; normalmente de forma gráfica.

- Establecer la base para la creación de un diseño del software.
- Definir un modelo contra el que validar una vez construido el software.

Análisis Estructurado

Origen

- Crisis del software.
- Aparecen los primeros lenguajes estructurados.
- Aparecen técnicas de análisis y diseño estructurado.

Productos a obtener

El análisis proporciona modelos del sistema derivados de los requisitos del usuario:

- Modelos de Datos (E/R)
- Modelos de Procesos (DFD)
- Diccionario de Datos (DD)
- Especificación de Procesos (EP) Estos modelos se obtienen a partir de las Técnicas de Análisis.

Objetivos

- Dejar constancia de los límites del sistema considerado.
- Construir un modelo lógico de procesos del sistema basándose en explosiones por nivel.
- Reflejar los procesos que transforman la información.
- Simplificar la complejidad del sistema y su compresión.
- Facilitar el mantenimiento del sistema

Conceptos

Entidad Externa:

- Refleja una interfaz entre el sistema considerado y su entorno.
- Son entes ajenos al sistema considerado (personas, unidades organizativas, sistemas o cualquier otro elemento que no forme parte del sistema).
- Aportan o reciben información desde el punto de vista del sistema tratado. Proceso:
- Refleja una actividad que transforma datos de entrada en datos de salida.
- Sólo es lugar de transformación y no es origen ni final de datos. Almacén de datos:
- Representa un depósito de información en el sistema. Es un contenedor de paquetes de información cuyo nombre lo suele identificar.
- Se emplea si es necesario por los requisitos o si los procesos que lo manejan actúan con diferente temporalidad.
- No puede crear, destruir ni transformar datos. Sólo los almacena estáticamente. Flujo de datos:
- Transportan la información desde/hasta los otros componentes de un DFD (procesos, almacenes y entidades externas) estableciendo la comunicación entre ellos.
- Siempre tendrán un proceso como origen y/o destino.
- Pueden transportar datos (elementos lógicos) o materiales (elementos físicos).
- Si están claros y/o empleamos el DD, se pueden consolidar flujos por claridad.

- El mismo contenido puede tener diferentes significados en diferentes partes del sistema.
- Tipos:
    - Divergentes: el flujo fluye por diferentes caminos.
    - Convergentes: se combinan flujos para formar uno de mayor complejidad.

Aplicación

Un sistema puede llegar a tener numerosos procesos, almacenes, flujos y entidades externas. Por ello, el DFD es necesario organizarlo en niveles:

- TOP-DOWN
- BOTTOM-UP
- MIDDLE-UP La aproximación más común es TOP-DOWN, ya que facilita la aproximación y compresión del sistema no sólo por el usuario, sino también por el analista. Lo que se obtiene es una jerarquía de niveles de explosión: Diagrama de Contexto, Diagrama de Sistema, Nivel 2, Nivel 3... El primer nivel (nivel 0 o Diagrama de Contexto), mostrando:
- Las interfaces de éste con las entidades externas.
- Los almacenes externos (si los hubiera). El segundo nivel (Diagrama de Sistema) consiste en tantos procesos (burbujas) como subsistemas (funciones más importantes del sistema), por lo tanto, representa una visión a alto nivel del sistema global. Los siguientes niveles de explosión refinan/explican/detallan, si hiciese falta, los procesos identificados en el nivel anterior.
  No todas las partes del sistema deben tener el mismo nivel de explosión; es decir, la "altura del árbol de explosión" no tiene que ser homogénea. Si la especificación de un proceso primitivo (burbuja de último nivel) no puede hacerse en una sola página, generalmente indicará la necesidad de explosionarlo. Un nivel de DFD no debería contener más de media docena de procesos (burbujas) en un caso general. Regla genérica: 6±1, que si no se cumple puede indicar la necesidad de un nivel intermedio. Los almacenes se muestran por primera vez en el nivel más alto donde actúan como interfaz entre dos o más procesos. Los niveles deben ser necesariamente consistentes entre sí; esto es, deben estar balanceados: Principio de Conservación de Flujo. Los flujos de datos de salida y entrada de una burbuja (proceso) en un nivel dado deben corresponderse con los que salen y entran en el DFD que refleja la explosión de ese proceso (siguiente nivel).

Características de un buen DFD

- ¿Hay consistencia entre niveles? Principio de conservación de flujo.
- ¿Faltan procesos en el DFD obtenido?
- ¿Faltan flujos de datos que un proceso requiere para "hacer su trabajo"?
- ¿Sobran flujos de datos que para el proceso no son necesarios?
- ¿Hay redundancias en el DFD obtenido?
- ¿Hay cruces que dificultan la legibilidad del DFD?
- Por último, no deben reflejarse "subsistemas" que sólo se vayan a emplear una vez; esto es, al poner en marcha el sistema. Esto será objeto de otro sistema.

Temporalidad

Un DFD es una red de procesos asíncronos que se comunican; esto es, no se especifica temporalidad. En un DFD no se refleja o modela ninguna secuencia de ejecución ni alternativas o bifurcaciones. Tan solo la ausencia o existencia de información puede implícitamente establecer una secuencia entre procesos.

Recomendaciones

- Evitar la complejidad, un DFD debería caber cómodamente en un DIN A4 apaisado.
- Escoger nombres significativos para todos los elementos: entidades, procesos, almacenes y flujos.
- Numerar correctamente los procesos: así se muestra la jerarquía de explosión.
- El DFD ha de ser consistente.

Consistencia

Evitar sumideros infinitos o agujeros negros; no puede haber procesos que tengan flujos de entrada pero no de salida. Evitar procesos de "generación espontánea", generalmente no son correctos los procesos que tienen salidas pero no entradas. Evitar flujos y procesos no etiquetados; permite que uno no se pierda al explosionar y asegurarse de que las cosas son coherentes y razonables. Evitar almacenes de "sólo lectura" o "sólo escritura"; el único caso en que es posible almacenes con sólo entradas o sólo salidas es el de los almacenes externos.

Análisis Estructurado

Notación

= está compuesto de
+ y
() optativo
{} iteración (0 o más veces) Se puede poner límite inferior, superior o ambos.
[] seleccionar una de varias alternativas. A escoger sólo una.
| separa opciones alternativas
** comentario
@ identificador (campo clave) para un almacén.
Alias Se referencia en el DD para que sea completo y se referencia al nombre "oficial" del dato: computador = *alias de cliente*. Sólo si los usuarios no se ponen de acuerdo en un solo nombre. Mejor no usar alias.

Definición de un dato

Para definir por completo un dato, se debe incluir:

- El significado del dato dentro del contexto de la aplicación considerada (normalmente se ofrece como comentario).
- La composición del dato, si se compone de partes elementales con significado.
- Los valores que puede tomar el dato, si es un dato elemental que no se puede descomponer más.

Balanceo del DFD y DD

Cada flujo y almacén de datos del DFD debe estar incluido en DD. Cada dato y almacén del DD deben estar en alguna parte del DFD. Cada burbuja del DFD tiene que estar asociada con un DFD de nivel inferior o en una EP, pero no con ambos. Cada EP debe tener una burbuja de nivel inferior asociada en el DFD. Las entradas y salidas deben de coincidir en la EP y en el DFD. Cada referencia de un dato en la EP tiene que cumplir una de las siguientes reglas:

- Coincidir con el nombre de un flujo o almacén de datos conectado a la burbuja descrita por la EP.
- Es un término local, definido explícitamente en la EP.
- Aparece como componente en una entrada del DD para un flujo o almacén de datos conectado a la burbuja. Cada entrada de DD tiene que tener una referencia en una EP, un DFD o en el propio DD. El E/R debe ser consistente con el DFD:
- Cada almacén del DFD debe corresponderse con una entidad, una relación del E/R o una combinación de entidades y relaciones.
- Análogamente, toda entidad y relación del E/R tiene que tener su reflejo en el DFD
- Los nombres de entidades y relaciones en el E/R deben coincidir con los nombres de los almacenes de datos en el DFD. El E/R debe ser consistente con las EP:
- Las EP deben crear y eliminar instancias de cada entidad y relación que aparece en el E/R.
- Algún proceso del DFD tiene que usar o leer los valores de cada dato. En general, las entidades se nombran en singula y los almacenes en plural. Todos los atributos de una entidad aparecerán en el DD.

Análisis Estructurado

Pasos recomendados

1. Obtener el conocimiento del entorno:

- Breve descripción iniciañ deñ entorno, preferiblemente escrita por el usuario. Inicialmente llegarán unas pocas páginas
- Mantener entrevistas estructuradas con el usuario / grupo de usuarios.
- Revisar/Refinar las entrevistas hasta que no existan puntos conflictivos.
- Recoger los requisitos por escrito en un documento.
- Obtener el consenso con respecto al documento elaborado.

2. Obtener los modelos de análisis:

- Obtención en paralelo de modelos de procesos, modelos de datos, diccionario de datos para recoger los elementos de los DFD y el E/R
- Continuar explosionando el DFD hasta el nivel adecuado sin "matar moscas a cañonazos".
- Blancear el DFD, el E/R y el DD a lo largo de todo el proceso.

3. Obtener las EP:

- Obtención de las EP de al menos las burbujas primitivas.
- Evaluar la necesidad de elaborar las EP de las burbujas del resto de niveles.
- Balancear la EP con el resto de modelos.

Diseño estructurado

Es diseño estructurado es una técnica de diseño que permite el paso entre el análisis del problema y la instrumentación del mismo. Consiste en un conjunto de reglas que aplicadas a una especificación, permite derivar una arquitectura del software.

Características

- Permite obtener una solución a partir de la definición del problema, ya que la solución tiene los mismos componentes e interrelaciones que en el problema original.
- Simplifica el problema, lo subdivide y cre una división del problema en cajas negras, organizadas en jerarquías.
- Ofrece un conjunto de estrategias para desarrollar una solución de diseño a partir de análisis. Permite, mediante una serie de estrategias, derivar una arquitectura del software a partir del DFD que define la solución propuesta al problema.
- Utiliza técnicas gráficas que facilitan la especificación del diseño y la comunicación de éste. La diagramación es imprescindible en cualquier ingienería. En el diseño estructurado, se utiliza el **diagrama de estructuras**, también pueden utilizarse las **tablas de interfaz**.
- Criterios para evaluar la calidad de la solución respector al problema.

Diagrama de Estructuras (DEC)

Caracteríasticas

Puede observarse directamente la jerarquía de control, la comunicación entre módulos, los datos de entrada/salida, los datos de control de entrada/salida. Para que un diagrama de estructuras sea fácil de entender, los nombres de los módulos, datos y flags deben ser lo más significativos posibles. Los flags no acostumbran a representarse, excepto en el análisis de transacción. Una diapositiva del diagrama de estructuras, al igual que en cualquier proceso de ingienería, es la posibilidad de abstracción. Cuando la comunicación entre módulos es muy compleja, se pueden abstraer en el diagrama los componentes de la comunicación. No obstante, hay que tener en cuenta que se debe conseguir la máxima cohesión y el mínimo acoplamiento.

Módulos

AECC: Parte lógica separable de un programa. Yourdon: Es una secuencia contigua de sentencias de programa, limitada por delimitadores y que tiene un identificador globar. Fenton: Cualquier objeto que, en un nivel de abstracción dado, queramos considerar como un concepto simple. D.E: Es aquella parte del código que se puede llamar.

Visiones del módulo

Externa

- Entrada
- Salida
- Función Interna
- Mecanismos
- Datos internos

Conexión y comunicación

Conexión: líneas que conectan el módulo llamador con el llamado. Comunicación: Datos de procesamiento, la relación de ellos con el problema, datos importantes del mundo exterior Flags: valores de condición, importantes internamente.

Tabla de interfaz

Especifica los parámetros, datos o flags, que se pasan entre módulos. La tabla de interfaz especifica, por lo tanto, la interfaz de los módulos.

Componentes

- Módulo llamado
- Parámetros formales
- Parámetros de entrada
- Parámetros de salida
- Uso del parámetro
- Significado del parámetro

Letra	Significado
P	El parámetro es procesado
M	El parámetro es modificado
T	El parámetro es transferido a un módulo subordinado
C	El parámetro es usado como una variable de control
I	El parámetro es transferido a un módulo subordinado y modificado en éste

Módulo	Parámetro Formal	Entrada	Salida	Uso	Significado
Factorial	Número	S	N	P	Factorial a calcular
Factorial	Resultado	N	S	M	Factorial calculado

Uso

Las tablas de interfaz no son independientes de los diagramas de estructura, sino que los complementa. Permiten al diseñador mejorar la claridad de los diagramas de estructuras mediante una especificación más rigurosa de las interfaces. Permite desglosar una comunicación de datos definida en un diagrama de estructuras.

Estrategias de Diseño

Permite una transformación del DFD incluido en la especificación de requisitos a una descripción de diseño de la arquitectura del software. Dos técnicas distintas: Análisis de transformación y análisis de transacción

Análisis de transformación

La información llega al sistema mediante caminos que transforman los datos externos (flujos de llegada). En el centro del sistema, en el núcleo, ocurre una transformación. Los datos de llegada transformados circulan por caminos que conducen a la salida.

Análisis de transacción

La información llega al sistema mediante caminos que transforman los datos externos (flujos de llegada). En el centro del sistema, el núcleo, ocurre un encaminamiento del flujo de entrada. Los datos de llegada encaminados circulan alguno de los distintos caminos de acción (flujo de salida).

Técnica

En un DFD real, los procesos formarán combinacioens de transformación-transacción. Deberán realizarse ambos análisis. Las zonas de transformación y transacción no están definidas por la forma. Se debe analizar el flujo de datos en el DFD para saber ante qué estamos.

Análisis de transformación

1. Revisar el modelo del sistema

Se debe obtener el DFD de la especificación del sistema. Si no existiese, debería crearse a partir de las especificaciones funcionales del sistema. El DFD debe tener el suficiente nivel de detalle, típicamente una profundidad de tres o más niveles. Es necesario para poder aplicar el análisis de transformación con el detalle necesario.

2. Determinar las características del DFD

En general, cualquier DFD puede realizarse únicamente con caminos de transformación. Sin embargo, para la aplicación de esta técnica conviene escoger aquellos DFD con características claras de transformación. Si existe algún proceso con salidas exclusivas, probablemente estamos ante un DFD de transacción.

3. Aislar el centro de transformación

El centro de transformación contiene las funciones esenciales del DFD. Los límites de flujo de llegada y del flujo de salida están abiertos a interpretación, se pueden derivar distintas alternativas de diseño.

4. Realizar un "Primer Nivel de Factorización"

A todos los módulos se les debe asignar un nombre significativo que defina lo que van a realizar sus módulos subordinados.

5. Realizar un "Segundo Nivel de Factorización"

Se progresa desde el centro de transformación en dirección contraria al flujo de llegada, colocando los procesos del DFD como módulos subordinados. Con el flujo de salida debe realizarse el mismo proceso. Al final de la jerarquía, se introducen módulos predefinidos que proporcionen las entradas y salidas necesarias.

6. Refinar la arquitectura del sistema

Aumentar o disminuir los módulos. Se deben especificar los datos comunicados entre módulos de la arquitectura del sistema. Las flags no acostumbran a representarse, aunque en algún caso, por claridad, pueden ser necesarios. Estos datos se derivan por refinamiento de los flujos del DFD.

7. Revisar el diseño

Se debe revisar el diagrama de estructura para comprobar que el diseño derivado es correcto. Por ejemplo, se puede realizar una simulación de la ejecución de la jerarquía obtenida, para comprobar si es consistente con el DFD. De este forma, podrá comprobarse que el orden de ejecución es correcto.

Análisis de transacción

1. Revisar el modelo del sistema

2. Determinar las características del DFD

3. Identificar el centro de transacción

Se debe determinar que proceso contituye el centro de transacción. Este proceso es, normalmente, identificable a simple vista, ya que de él emanan varios caminos independientes de tratamiento. Por último, debe aislarse el flujo de llegada y evaluar cada camino de acción. Los caminos de acción pueden pertenecer a un camino tipo transformación o a un camino tipo transacción. Cada camino debe identificarse individualmente para realizar el segundo nivel de factorización.

4. Realizar un "Primer Nivel de Factorización"

5. Realizar un "Segundo Nivel de Factorización"

Se desarrolla el camino de entrada al igual que en el análisis de transformación. Cada camino de acción se desarrolla según su tipo. Transformación o Transacción.

Metodología de desarrollo software

Conjunto de métodos, procedimientos, técnicas, herramientas y soportes documentales que definen las reglas para realizar las transformaciones internas de las actividades de un modelo de Ciclo de Vida, y que permiten a los desarrolladores implementar nuevos productos de software.

Ciclo de vida

IEEE 1074: "Una aproximación lógica a la adquisición, el suministro, el desarrollo, la explotación y el mantenimiento del software" ISO 12207: "Un marco de referencia que contiene los procesos, las actividades y las tareas involucradas en el desarrollo, la explotación y el mantenimiento de un producto software, abarcando la vida del sistema desde la definición de los requisitos hasta la finalización de su uso"

Proceso

Conjutno de pasos realizados para un fin determinado. (IEEE) Un conjunto de actividades interrelacionadas que transforman entradas en salidas (ISO 12207 / UNE 77104) Un proceso, entendido de manera general, es una serie de pasos que incluyen actividades, restricciones y recursos que resultan en un producto determinado con ciertas características.

Diferencias

Ciclo de Vida: indica qué es lo que hay que obtener a lo largo de desarrollo del proyecto, pero no cómo. **Metodología**: dota de contenido a los Ciclos de Vida. Una metodología puede seguir uno o varios modelos de Ciclo de Vida. **Proceso**: pasos de ingienería necesarios para alcanzar un producto. Una metodología incluye varios procesos, como por ejemplo, proceso de gestión, proceso de análisis, etc.

Metodología de desarrollo software

Conjunto de métodos que se utilizan en una determinada actividad con el fin de formalizar y optimizarla. Determina los pasos a seguir y cómo realizarlos. Optimiza el proceso y el producto software. Métodos que guían en la planificación en el desarrollo del software. Define qué hacer, cómo y cuándo durante todo el desarrollo y mantenimiento de un proyecto. Define una estrategia global para enfrentarse con el proyecto. Entre los elementos que forman parte de una metodología:

- **Fases**: tareas a realizar.
- **Productos**: E/S de cada fase.
- **Procedimientos y herramientas**: apoyo a la realización de cada tarea.
- **Criterios de evaluación**: del proceso y del producto. Saber si se han logrados los objetivos.

Ventajas uso Metodologías

Gestión:

- Facilita la tarea de planificación.
- Facilita la tarea de control y seguimiento de un proyecto.
- Mejorar la relación coste/beneficio.
- Optimizar el uso de recursos disponibles.
- Facilitar la evaluación de resultados y cumplimiento de los objetivos
- Facilitar la comunicación efectiva entre usuarios y desarrolladores. Ingenieros de software:
- Ayudar a la compresión del problema.
- Optimizar el proceso de desarrollo.
- Facilitar el mantenimiento del producto.
- Permitir la reutilización de partes del producto. Cliente o usuario.
- Garantía de calidad en el producto final.
- Confianza en los plazos fijados en la definición del proyecto.

Tipos de Metodologías

Elegir la metodología adecuada para un determinado proyecto es trascendental para el éxito del producto. Según la filosofiías del desarrollo:

- Metodologías tradicionales.
- Metodologías ágiles. Según el paradigma:
- Metodologías estructuradas.
- Metodologías orientadas a objetos.

Metodologías tradicionales

Cumplir con un plan de proyecto definido en la fase inicla del desarrollo. Llevar una documentación exhaustiva de todo el proyecto. Altos costes ante cambios y falta de flexibilidad en proyectos donde el

entorno es volátil. Se focalizan en la documentación, planificación y procesos.

Metodologías ágiles

Nacen como respuesta a los problemas que puedan ocasionar las metodologías tradicionales. Se basan e la adaptabilidad de los procesos de desarrollo: retrasar las decisiones y planificación adaptativa. La capacidad de respuesta a un cambio es más importante que el seguimiento estricto de un plan. Es un proceso Incremental, Cooperativo y Adaptativo.

Principios

1. Se encarga de valorar al individuo y las interacciones del equipo más que a las herramientas o los procesos utilizados.
2. El cliente está en todo momento colaborando en el proyecto.
3. Es más importante crear un producto software que funcione, que escribir mucha documentación.
4. Es más importante la capacidad de respuesta ante un cambio realizado que el seguimiento escrito de un plan.

¿Tradicionales o Ágiles?

Metodologías Tradicionales	Metodologías Ágiles
Especialización	Equipo multidisciplinar
Fases	Solapamiento
Requisitos detallados	Visión del producto
Seguimiento del plan	Adaptación a los cambios

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

Metodologías tradicionales	Metodologías ágiles
Más roles	Pocos roles
La arquitectura del software es esencial y se expresa mediante modelos	Menos énfasis en la arquitectura del software

Críticas al desarrollo ágil

Falta de estructura y documentación. Enorme coste para los clientes, encuentros a intervalos muy frecuentes. Muy complicado obtener estimaciones realistas del esfuerzo para proporcionar un presupuesto. Puede ser muy ineficiente si los requisitos de un área cambian durante varias iteraciones, misma programación varias veces.

¿Tradicionales o Ágiles?

características relevantes para decidir la metodología más adecuada:

1. Principal prioridad de negocio.
2. Estabilidad de los requisitos.
3. Rigidez del producto.
4. Coste de prototipado.
5. Criticidad del sistema.
6. Tamaño del sistema.

Metodología tradicional Métrica versión 3

MÉTRICA es una metodología para la simplificación, desarrollo y mantenimiento de sistemas de información (SI). Promovida por el antiguo Ministerio de Administraciones Públicas del Gobierno de España. Sistematización de actividades del ciclo de vida de los proyectos software en el ámbito de las administraciones públicas. Cubre el Proceso de Desarrollo y de Mantenimiento de SI. Abarca el desarrollo completo de SI sea cual sea su complejidad y magnitud. Debe adaptarse y dimensionarse de acuerdo a las características particulares de cada proyecto. Enfoque orientado al proceso. Descompone cada uno de los procesos en actividades, y éstas en tareas. Establece un conjunto de técnicas a utilizar y productos a obtener. Desarrollar SI de mayor calidad, productividad y satisfacción de los usuarios. Facilitar el mantenimiento posterior.

Métrica V3. Procesos

Planificación de Sistemas de información (PSI): Proporcionar un marco estratégico de referencia para los SI de un determinado ámbito de la Organización. **Desarrollo de Sistemas de Información**: Contiene todas las actividades y tareas que se debe llevar a cabo para desarrollar un sistema, cubriendo desde el análisis de requisitos hasta la instalación del software. **Mantenimiento de Sistemas de Información (MSI)**: Obtener una nueva versión de un SI a partir de las peticiones de mantenimiento. **Estudio de Viabilidad del Sistema (EVS)**: Analizar un conjunto concreto de necesidades, con la idea de proponer una solución a corto plazo. **Análisis del Sistema de Información (ASI)**: Especializacióndetallada del SI, a través de un catálogo de requisitos y una serie de modelos. **Diseño del Sistema de Información (DSI)**: Obtener la definición de la arquitectura del sistema y del entorno tecnológico que le va a dar soporte, junto con la especificación

detallada de los componentes del SI. **Construcción del Sistema de Información (CSI)**: Construcción y prueba de los distintos componentes del SI. **Implantación y Aceptación del Sistema de Información (IAS)**: Entrega y aceptación del sistema, incluyendo actividades para el paso a producción del sistema. Cada proceso e Interfaz está compuesto por **Actividades**, a su vez, las Actividades están compuestas por **Tareas**.

Proceso o Interfaz	**# Actividades**	**# Tareas**
Planificación de SI (PSI)	9	24
Estudio de Viabilidad del Sistema (EVS)	6	18
Análisis del SI (ASI)	11	34
Diseño del SI (DSI)	12	44
Construcción del SI (CSI)	9	19
Implantación y Aceptación del SI (IAS)	10	24
Mantenimiento de SI (MSI)	4	10
Gestión de Proyectos	16	29
Aseguramiento de la Calidad	25	38
Seguridad	31	34
Gestión de Configuración	5	8
Total	**138**	**282**

Métrica V3. Técnicas y Prácticas

Se utilizan para la elaboración de productos. Mejoran la productividad y aseguran la calidad de los productos. Una **técnica** es un conjunto de heurísticas y/o procedimientos que utiliza una notaciń específica. Una **práctica** representa un medio para la consecución de unos objetivos específicos de manera rápida, segura y precisa. Se distinguen 3 tipos:

- Técnicas de desarrollo.
- Técnicas de gestión de proyectos.
- Prácticas.

Métrica V3. Participantes

- Directivo
- Jefe de Proyecto
- Consultor
- Analista
- Programados

Metodología ágil Scrum

Scrum: Formación de rugby. Scrum es el término que deine la formación más reconocible del rugby, en la que ambos equipos, agazapados y atenazados entre sí empujan para obtener el balón y sacarlo de la formación sin tocarlo con la mano. A finales de los 80, Nonaka y Takeuchi compararon la nueva forma de trabajo en equipo que estaba identificando en las empresas tecnológicas más innovadoras con el avance en formación de Scrum. A finales de los 90, Ken Schwaber y Jeff Sutherland presentaron de forma conjunta un artículo describiendo el método Scrum. Características del método Scrum:

- Adoptar una estrategia de desarrollo incremental, en lugar de la planificación u ejecución completa del producto.
- Basar la calidad del resultado más en el conocimiento táctico de las personas en equipos autoorganizados, que en la calidad de los procesos empleados.
- Solapamiento de las diferentes fases del desarrollo, en lugar de realizarlas una tras otra en un ciclo secuencial o de cascada. **Sprint**: Iteración, núcleo central que proporciona la base de desarrollo iterativo e incremental. Cada sprint finaliza con la entrega de una parte operativa del producto: **incremento**. Se monitoriza la evolución de cada sprint en reuniones breves diarias.

Roles

Propietario del producto: decide en última instancia cómo será el resultado final y el orden en el que se van construyendo los sucesivos incrementos: qué se pone y qué se quita de la pila del producto y su prioridad. **Equipo de trabajo**: Equipo **multifuncional**, en el que todos los miembros trabajan de forma solidaria con responsabilidad compartida. **Scrum Master**: Es el responsable del cumplimiento de las reglas de un marco de scrum. Proporciona la asesoría y formación necesaria al propietario del producot y al equipo. **Interesados**: Usuarios, expertos, etc. Cualquiera que esté involucrado e interesado en el proyecto.

Introducción a la Calidad

Definiciones de calidad

La calidad es la totalidad de las **características de un producto o servicio** que tienen la capacidad de satisfacer las **necesidades explícitas o implícitas**. ANSI Conjunto de **propiedades** de un **producto o servicio** que le confieren su aptitud para **satisfacer unas necesidadese explícitas o implícitas**. ISO 8402 Calidad es el nivel al que una serie de **características inherentes satisfacen los requisitos**. ISO 9000:2000

Definiciones de calidad software

Concordancia con los **requisitos funcionales y de rendimiento** explícitamente establecidos con los estándares de desarrollo explícitamente documentados y con características implícitas que se espera de todo software desarrollado profesionalmente. R.O.Pressman La capacidad de un conjunto de **características** inherentes de un **producto, o componente del producto, o proceso,** de satisfacer por completo los requisitos del **cliente**. CMMI

Datos - Información relevante

Cuesta entre 4 y 6 veces más obtener un cliente nuevo que retener a uno antiguo. Un porcentaje muy pequeño de clientes descontentos se quejan, menos del 5%. Un cliente descontento "transmite" su

descontento al menos a 9 personas. Solo el 10% de los clientes con una mala experiencia vuelve a repetir la compra. Un 70% de los clientes que dejan la compañía, lo hace por mala calidad de servicio. Un 50% de los clientes eligen una compañía en base a recomendaciones de conocidos. Solo la mitad de los servicios prestados satisfacen de forma efectiva al cliente. El 30% de los servicios plantean alguna incidencia, pero los clientes no la materializan en una reclamación. El 15% de los servicios prestados motivan reclamaciones verbales o escritas que no suelen llegar a los equipos de gestión. Solo el 3%de los servicios prestados motivan una queja por escrito correctamente dirigida. Las razones de cambio de una entidad finacniera por lo s clientes particulares son: 68% por aspectos de calidad de servicio y atención y; el 19% por oferta de productos o precios más ventajosos y el 13% restante por motivos ajenos a la entidad. De una inversión en proyectos de $255 billones (americanos), se desperdician $55 billones. De cada 100 proyectos, 94 se reinician. Al liberar un producto, tan sólo están incluidas el 52% de las funciones y propiedades requeridas. De media los costes de los proyectos suponen el 143% de lo estimado, y el 82% se pasa de plazos.

Prespectiva de costes de la calidad

Es lo que le cuesta a la empresa prevenir los fallos, detectarlos y también lo que le cuestan los fallos producidos. **Costes de prevención**: Son los derivados de controlar que el proceso de producción se atengan a los criterios de calidad establecidos y prevenir la aparición de defectos. Revisiones de diseño formación, evaluación de proveedores, mantenimiento preventivo. **Costes de evaluación**: Asociados a la producción terminada y auditorías de funcionamiento: Inspecciones, pruebas, aceptación, etc. **Costes de la no calidad**: Relacionados con la producción, antes o después de la entrega al cliente; a) internos (reclamaciones, retrabajo por reparaciones, materiales, etc.) y b) externos (gastos de garantía, retirada de productos del mercado, juicios, etc.)

Prespectiva de costes

Relación coste/beneficio: **A mayor calidad del producto los costes de la no calidad disminuyen, pero al mismo tiempo el coste del sistema de calidad aumenta. El perfeccionismo también resulta caro.

Ventajas de la calidad

- Mejora la calidad.
- Mejora la productividad.
- Reduce costes.
- Reduce precios.
- Crecimiento del nogocio.
- Excepcional reporno de la inversión.

Prespectiva histórica

Inspección

Asociado al concepto taylorista de la producción nace el control de la calidad como **supervisión de los productos terminados** según el concepto de "aceptado" o "no aceptado". No se aceptaban los que no superaban la inspección:

- El objetivo principal es la **detección de errores**.
- Énfasis en la uniformidad de servicio.
- Responsabilidad centrada en el departamento de inspección.

- La calidad se comprueba: **inspeccionar, clasificar, contar y medir**. Henry Ford fue de los primeros en aplicar técnicas de inspección en 1918. En 1924 Bell creó un Departamente de Inspección del que formaron parte muchos precursores de la calidad actual, como Shewhart.

Control de calidad

Hacia 1924-1931 el matemático **Walter A. Shewhart** introdujo el **Control de la Calidad Estadístico**, el cual fue aplicado en la segundo guerra mundial. El control de calidad se puede definir como el conjunto de técnicas y actividades, utilizadas para verificar los requisitos relativos a la calidad del producto o servicio.

- El objetivo principal es el control para detectar errores en los productos finales.
- Énfasis en la uniformidad de servicio.
- Introduce herramientas y técnicas estadísticas.
- Responsabilidad centrada en departamentos de calidad y de inspección.
- La calidad se obtiene por comparación de las especificaciones con el resultado del control del producto terminado.

Aseguramiento de la calidad

Conjunto de **actividades planificadas y sistemáticas** definidas en un sistema de calidad con el objetivo de **garantizar que se satisfacen los requisitos de calidad de los productos y servicios**. Sus principales características son:

- El objetivo principal es la coordinación de los diferentes procesos de producción del producto o servicio. **El foco se pone en los procesos**.
- Énfasis en la totalidad de la cadena, incluidas las fuentes de I+D y las áreas de apoyo.
- Se tiene que apoyar en programas y sistemas de calidad.
- La responsabilidad de la calidad es de todos los departamentos implicados.
- La calidad se obtiene realizando las tareas según las normas y el número de desviaciones. En **1946** se crea la Amercian Society for Quality Control. En **1956** nace la Organización europea para el Control de Calidad. En **1950** nacen los primeros estándares de calidad en el ámbito militar. En **1979** IEEE emitión su primera norma sobre la calidad del software. su primera norma sobre la calidad del software.

Gestión de la calidad total y excelencia

Surgió en la década de los ochenta para impulsar la calidad entendida como sinónimo de **"satisfacción del cliente". Nacen los modelos y premios Malcom Balbridge, Premio Deming, y el Modelo europeo EFQM. Sus características principales son:

- El objetivo principal es el **impacto estratégico**.
- Se percibe la calidad como una ventaja competitiva.
- Énfasis en el mercado y necesidades de los clientes.
- **Alcanza todos los procesos de la empresa**.
- La responsabilidad de la calidad es de todos los miembros de la empresa.
- La calidad se obtiene con actividades de gestión, control y aseguramiento de la calidad. **A medidados de los 80** nacen las primeras normas internacionales: ISO 9000. **En 1987** se crea el Malcolm Balbridge Quality Award. En 1992 se establece el Premio Europeo a la calidad. EFQM.

Características principales	CONTROL DE CALIDAD	ASEGURAMIENTO DE LA CALIDAD	GESTIÓN DE LA CALIDAD
Concierne	Depto. de control de calidad	Auditores de calidad	Todas las personas
Se aplica	Al producto: inicial, intermedio o final	A los procesos operativos	A todos los procesos
Se actúa para	Encontrar defectos	Encontrar no conformidades	Conseguir objetivos
Se orienta	Al efecto (producto)	A las causas (procesos)	A las causas (procesos)
Participación del personal	No necesaria	Conveniente	Imprescindibles
Actitud	Arreglo / reacción	Prevención	Mejora

Configuración de la calidad

Política de Calidad. Directrices y objetivos generales de una empresa o parte de una empresa, relativos a la calidad, expresados formalmente por la Dirrección. **Sistemas de Calidad. Conjunto de elementos que se establecen** en una empresa para gestionar la calidad: Estructura organizativa, responsabilidades, procedimientos, procesos y recursos. **Gestión de la Calidad. Conjunto de actividades** que una empresa o parte de la empresa lleva a cabo para conseguir los objetivos definidos en la política de calidad. **Plan de calidad**. Documento que especifica **qué procedimientos y recursos** deben aplicarse, quién debe aplicarlos y cuándo deben aplicarse a un proyecto, producto, proceso, contrato, etc.

Factores de calidad de McCall orientadas al producto

Revisión del producto:

- Facilidad de mantenimiento: ¿Puedo corregirlo?
- Flexibilidad: ¿Puedo cambiarlo?
- Facilidad de prueba: ¿Puedo probarlo? Transición del producto:
- Portabilidad: ¿Podré usarlo en otra máquina?
- Reusabilidad: ¿Podré reusar alguna parte del software?
- Interoperabilidad: ¿Podré hacerlo interactuar con otro sistema? Operaciones del producto:
- Corrección: ¿Hace lo que quier?
- Fiabilidad: ¿Lo hace de forma fiable todo el tiempo?
- Eficiencia: ¿Se ejecutará en mi hardware lo mejor que pueda?
- Integridad: ¿Es seguro?
- Facilidad de uso: ¿Está diseñado para ser usado fácilmente

Atributos de Calidad del Software

Bell 2000 orientados al producto

- Fiable: capacidad de ofrecer los mismos resultados bajo las mismas condicioens.

- Eficiente: Utilización óptima de los recursos de la máquina.
- Robusto: No poseer un comportamiento catastrófico ante situaciones excepcionales.
- Correcto: Se ajusta a las especificaciones dadas por el usuario.
- Portable: capaz de integrarse en entornos distintos con el mismo esfuerzo.
- Adaptable: Modificar alguna función sin que afecte a sus actividades.
- Inteligible: Diseño claro, bien estructurado y documentado.
- No erróneo: No exista diferencia entre los valores reales y los calculados.
- Reutilizable

Sommerville 2002 orientados al producto

- Mantenibilidad
- Coniabiliadad
  - Fiabilidad
  - Seguridad
  - Protección
- Eficiencia
- Usabilidad

Calidad del Proceso

La Mejora de Procesos de Software ayuda a las organizaciones:

- Mejoras en la entrega de los productos en funcionalidad, coste y plazos.
- Coordinación eficaz con proveedores y clientes. Proveer productos y servicios con reconocimiento de calidad (CMMI, ITMark, SPICE,...)
- Definición e implementación de una prespectiva integrada de negocio e ingienería.
- Procesos comunes, integrados y basados en la mejora para el desarrollo de sistemas y software.

CMMI

Integración de modelos de madurez de capacidades o Capability Maturity Model Integration, CMMI Es un modelo para la mejora y evaluación de procesos para el desarrollo, mantenimiento y operación de sistemas de software. Las mejors prácticas CMMI se publican en los documentos llamados modelos. En la actualidad hay tres áreas de interés cubiertas por los modelos de CMMI: Desarrollo, Adquisición y Servicios. En la versión 1.3 de CMMI la cual fue liberada el 1 de noviembre de 2010. Hay tres constelaciones:

- CMMI para el Desarrollo (CMMI-DEV p CMMI for Development), Versión 1.2 fue liberado en agosto de 2006. En él se tratan procesos de dearrollo de productos y servicios.
- CMMI para la adquisición (CMMI-ACQ o CMMI for Acquisition), Versión 1.2 fue liberado en noviembre de 2007. En él se tratan la gestión de la cadena de suministro, adquisición y contratación externa en los procesos del gobierno y la industria.
- CMII (CMMI-SVC o CMMI for Services), está diseñado para cubrir todas las actividades que requieren gestionar, establecer y entregar Servicios.

Calidad del proceso: CMMI

Las organizaciones no pueden ser certificadas CMMII. Por el contrario, una organización es evaluada (pro ejemplo, usando un método de evaluación como SCAMPI y recibe una calificación de nivel 1-5 si sigue los

niveles de Madurez, si bien comienza con el nivel 2). Nivel 1. Inicial, Los procesos generalmente ad hoc y caóticos. No hay un entorno estable para dar soporte a los procesos. Nivel 2. Administrado, Se garantiza los procesos, se planifican y ejecutan de acuerdo con las políticas. Asegurar que las buenas prácitcas se mantengan independientes de los vaivenes que afecten a una organización. Nivel 3. Definido, Los procesos ya no sólo se definen de manera independientemente para cada proyecto, sino que toda la organización goza de unas pautas comunes. Nivel 4. Administrado cuantitativamente, La organización y los proyectos establecen objetivos cuantitativos par ala calidad y el rendimiento del proceso, y los utilizan como criterios en la gestión de proyectos. Nivel 5. Optimizar, Mejorar continuamente el rendimiento de los procesos mediante mejoras incrementales e innovadores de proceso y de tecnología.

Enfoques - proceso de creación de valor

1. Voz del cliente: **Conocer necesidades y expectativas**
2. Desplegar la voz del cliente: **Transformarla en un plan de acciones para desarrollar productos** y servicios que le aporten valor.
3. Identificar procesos de producción y entrega, de acuerdo a estándares de calidad. **Desplegar Sistemas de Calidad**.
4. **Medir y evaluar la satisfacción del cliente** para orientar las accioens de mejora.

Introducción a la Planificación de Proyectos Software

Problemas del desarrollo software

- Proyectos fuera de plazo y presupuesto.
- Excesiva dependencia de los desarrolladores.
- Falta de control del desarrollo del proyecto.
- Escasa integración de las diferentes fases del desarrollo.
- Escaso control de calidad del producto.
- Escasa documentación actualizada de los proyectos.
- No utilizar una metodología formal.

Definición

Un **proyecto** es un conjunto de actividades coordinadas y controladas, con fechas de inicio y fin definidas, encaminado a la creación de un producto o servicio único y conforme a unos requisitos específicos, incluyendo limitaciones de tiempo, coste y recursos. La **gestión de proyectos** es la aplicación de un conjunto de conocimientos, habilidades, herramientas y técnicas a las actividades de un proyecto para satisfacer los requisitos del proyecto. Conclusión: Necesidad de **gestionar** el **proyecto** software.

Actividades del gestor

- Redacción de la propuesta.
- Planificación y calendarización del proyecto.
- Estimación de costes del proyecto.
- Supervisión del proyecto.
- Selección y evaluación del personal.
- Redacción y presentación de informes.

Planificación del proyecto

Proceso de planificación

1. Se valoran las restricciones que afectan al proyecto, fecha de entrega requerida, personal disponible, presupuesto, etc.
2. Se prepara un calendario para el proyecto.
3. Se comienza el proyecto.
4. Se revisa el proyecto y se actualiza el plan del proyecto

Plan de proyecto

Un **calendario** debería responder a:

- ¿Qué actividades hay que realizar y cuándo tienen que estar terminadas?
- ¿Existen los recursos necesarios y suficientes para abordar un nuevo proyecto?
- ¿Qué actividades tiene que hacer cada miembro del equipo?
- ¿Cuándo finaliza el analista x su participación en el proyecto y?
- ¿Cuánto durará este proyecto?
- ¿Cuánto costará este proyecto? Muestra la **división** y **organización** del trabajo en **actividades**. Fija los **recursos** disponibles: humanos, materiales, financieros, etc. Crea un **calendario** de trabajo, mostrando la relación entre actividades, recursos y tiempo.

Conceptos básicos

Tarea/Actividad: Unidad elemental del nivel de planificación: un proyecto se organiza en actividades. Cada actividad conduce a la obtención de un resultado a utilizar para el desarrollo de otras actividades del proyecto. **Hito**: Tipo de actividad especial que no tiene duración y sirve para indicar un acontecimiento, un momento particular e importante del proyecto. No consime recurosos. Normalmente se utiliza para describir puntos de control. Se usan mucho para gestionar el trabajo de terceras partes, subcontratadas, en el proyecto. **Hamaca**: Tipo especial de actividad que mide el tiempo transcurrido entre 2 puntos del proyecto. Las restricciones se deben establecer entre las tareas elementales y no entre las hamacas. **Esfuerzo** de una tarea/proyecto: Tiempo que le llevaría **a un sólo recurso** realizar la tarea/proyecto. **Restricciones**: Permiten establecer las dependencias entre las distintas tareas del proyecto para saber de qué manera han de encadenarse dichas tareas en la planificación. **Recursos**: Un elemento se considera recursivo si va a estar sujeto a compartición, posiblemente originando conflictos de uso. Los recursos son los que "hacen" las actividades del proyecto. Los recursos humanos tienen calendarios. **Tiempo/Duración** de una tare/proyecto: Duración de la tarea/proyecto una vez asignados los recursos. **Coste** de una tarea/proyecto: En función de los salarios de los recursos humanos y de los costes adicionales debidos a recursos materiales, maquinaria, etc. **Diagrama de Gantt**: Representa en una escala de tiempos cada una de las actividades del proyecto mediante barras, que representan su duración en fechas de calendario. Representa la organización en el tiempo de las actividades, con sus relaciones. **Nivelación**: Tras asignar recursos concretos a cada actividad a algunos se les puede estar exigiendo que trabajen más de su jornada. Estos recursos están **sobreasignados**. Posibles soluciones a sobrecargas:

- Alargamientos de las actividades al eliminar recursos de las mismas.
- Cambio de recursos.

- Introducción de recursos que compartan el esfuerzo.
- Modificación de la temporabilidad de la actividad.
- Segmentación de actividades. **Camino crítico**
- **Holgura** de una tarea: Tiempo que se puede retrasar una tarea sin que se retrase ninguna otra.
- **Tarea crítica**: Tarea sin holgura.
- **Camino crítico**: Sucesión de actividades sin holgura que marca la duración del proyecto. Si alguna de las actividades se retrasa, se retrasa todo el proyecto.
- Un proyecto puede tener varios caminos críticos diferentes. **Línea de base**: Foto fija de planificación a efectos de comparación en los sucesivos controles de avance y seguimientos del proyecto. Si el proyecto transcurre de forma óptima, transcurrirá como está establecido en la línea base. Cualquier desviación en el proyecto se debe comparar con la línea de base para determinar su gravedad.

Seguimiento del proyecto

A medida que un proyecto avanza, es necesario comprobar si todo sucede según lo previsto. Comparar los resultaods actuales con los planes previstos. Tomar acciones correctivas cuando existan desviacioens significativas con respecto a los planes previstos. Es necesario definir una normativa estándar para el seguimiento de proyectos.