Ejemplos preguntas cortas

Proceso Software (Universidade da Coruña)

Principales problemas del desarrollo software

• Proyectos fuera de plazo y de presupuesto.

• Excesiva dependencia de los desarrolladores. Malos desarrolladores harán malos Softwares.

• Falta de control del desarrollo del proyecto. Carece de información detallada y seguimiento.

• Escasa integración de las diferentes fases del desarrollo. No hay componente que una las fases de desarrollo.

• Escaso control de calidad del producto. El cliente busca calidad de producto.

• Escasa documentación actualizada de los proyectos. Cuando un proyecto Software es desarrollado, lo primero que se pide es documentación y pruebas realizadas.

• No utilizar una metodología formal.

Principios de la ingenieria software

Abstracción, encapsulamiento, Modularidad, Localización, Uniformidad, Completitud, Validación y Verificación

Fases genericas del desarollo software

Definición: análisis del sistema. Se extraen los requisitos y se plasman en al Especificación de Requisitos.

Desarrollo: Diseño, codificación, prueba. Se diseñan las estructuras y cómo se caracterizan las interfaces, se pasa al lenguaje de programación, ha de realizarse la prueba, se escriben y documentan los programas y se prueba el software construido.

Mantenimiento: Cuando el Software se empieza a utilizar y se centra en el cambio. Reparaciones y modificaciones cada vez que hay un fallo o nueva necesidad. Mayor coste.

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. Ej. antes de enviar la versión final nos manda una nueva función.

• Adaptativo: Adaptar los programas para acomodarlos a los cambios de su entorno externo (modificaciones en la legislación, CPU, SO, las reglas de negocio, etc.)

• Preventivo: El software se deteriora con los cambios, y este tipo de mantenimiento hace cambios en los programas para que se puedan corregir, adaptar y mejorar más fácilmente (Reingeniería del software).

Capas de la tecnologia multicapa

Procesos: Marco de trabajo que ayuda al jefe de proyecto a controlar la gestión del proyecto y las actividades de la ingeniería. El fundamento de la IS es la capa de proceso. Identifica todas las actividades y tareas, define el flujo, identifica los productos de trabajo y especifica los puntos de control de calidad requeridos.

Métodos: Actividades técnicas requeridas para la utilización de las técnicas de trabajo. Proporciona el "cómo" y cubre las actividades de ingeniería fundamentales. Se centra en las actividades técnicas que se deben realizar para conseguir las tareas de ingeniería.

Herramientas: Ayuda a la automatización de procesos y métodos. Proporciona soporte a las capas de proceso y métodos La automatización ayuda a eliminar el tedio del trabajo, reduce las posibilidades de errores y hace más fácil usar buenas prácticas de IS. Las herramientas ayudan a la automatización de actividades de gestión de proyectos, métodos técnicos usados en la ingeniería del software, soporte de sistemas general y marcos de trabajo para otras herramientas.

Diferencia entre usuario y cliente

Cliente: persona o entidad que encarga y financia el producto software. En un inicio solicita el software que se va a construir. Define los objetivos generales de negocio para el software. Proporciona los requisitos básicos del producto

Usuario: persona o entidad que manejará en su operativa el producto software. En realidad usara el software que se construye para alcanzar un propósito de negocios. Definirá los detalles operativos del software de forma que el propósito del negocio que pueda alcanzarse.

Diferencia entre ciclo de vida y 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 desarrollo: Desde el análisis hasta la entrega al usuario.

Diferencia entre cascada y modelo en v

Modelo en V: Es un modelo similar al de Cascada, aunque "mejorado". Tiene dos tiempos (tipos de actividades) claramente marcados: Rama descendente: actividades de desarrollo.y Rama ascendente: actividades de prueba.

Al ser una variante del de Cascada, hereda todas sus 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 sucesivas etapas. Favorece la consideración de las pruebas lo antes posible (comunicación horizontal entre las dos ramas de laV).

Es menos rígido que cascada y a diferencia de en cascada un cambio debido a un error puede suponer un gran coste.

Modelos evolutivos

Modelo Incremental: Proceso de construir una implementación parcial del sistema global y posteriormente ir aumentando la funcionalidad del sistema. Es la aplicación reiterada de varias secuencias basadas en el modelo en Cascada.

Modelo en Espiral: Fue propuesto por BarryW. Boehm (1988). Es un modelo de proceso evolutivo que permite combinar la naturaleza interactiva de construcción de prototipos con los aspectos controlados y sistemáticos del modelo en Cascada. Además, pone especial énfasis en el análisis de los riesgos para reducir su impacto.

Organización de los requisitos

- Utiliza estándares de estructuración de especificaciones de requisitos (como IEEE830)
- Aclara el objetivo global 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
- Relaciona los requisitos con otros elementos.

Pasos para identificar y definir requisitos

1. Observar y entender el trabajo desde el punto de vista del usuario
2. Interpretar el trabajo del usuario y la forma en la que él lo describe
3. Inventar mejores formas de hacer el trabajo.
4. 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.

Actividades de la Ingeniería de requisitos

- Obtención (Análisis del problema): Comienzo de cada ciclo. Los analistas deben trabajar junto al cliente para obtener el problema a resolver por el sistema.
- Análisis (Evaluación y negociación de los requisitos): Se centra en descubrir problemas con los requisitos identificados hasta este momento. Se leen los requisitos, se informa sobre ellos, se comentan con compañeros…
- Especificación. (documentar los requisitos): Se va realizando conjuntamente con el análisis (“pasar a limpio” el análisis),

- Validación (que sean consistentes y estén completos): Etapa final. Su objetivo es ratificar los requisitos (todos cumplen su descripción). Los procesos son únicos para cada compañía. No debe confundirse con verificación de requisitos (etapa posterior que no pertenece a este proceso).

¿Cuándo evolucionan 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. Se deben establecer políticas, guardar cambios…

Porque los usuarios cambiaron 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 a más alto nivel. Representan los objetivos, base de negocio, estrategia, alcance…

Requisitos de Usuario: Los requisitos de cliente o usuario, deben describir los requisitos del producto de tal forma que sean comprensibles por los propios usuarios del sistema sin conocimiento técnico previo

Requisitos del Sistema/Software: Utilizados por los ingenieros de software como punto de partida para el diseño del producto.

Técnicas de recogida de requisitos

Reuniones/Entrevistas, Cuestionarios y Encuestas, Brainstorming, Casos de uso, Prototipos, Escenarios

Elementos de los diagramas de casos de uso y su representación

Un **actor** es una entidad externa al sistema que realiza algún tipo de interacción con el mismo.

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

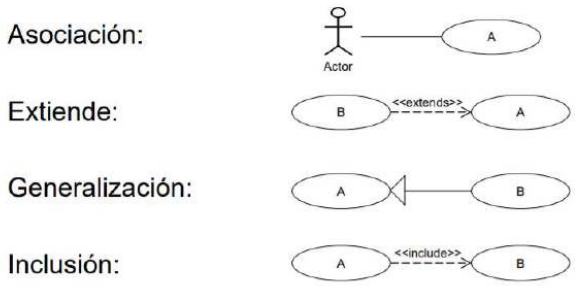
Relaciones entre actores y casos de uso se representan uniéndolos

Tipos de caso de uso

Trazo grueso: No entramos en detalle sobre las acciones que realiza el actor en él (ej. no es necesario explicar la política que aplica, basta con saber que existe).

Trazo fino: Aquellos que se especifican una vez se ha tomado la decisión de implementarlos, incluyendo detalles sobre la forma de la interfaz (ej. cómo se pasa de una ventana a otra). Se especifica con más detalle el comportamiento interno del sistema.

Tipos de relaciones en los diagramas de casos de uso



Proceso de análisis de requisitos con casos de uso

1. Identificación de actores
2. Identificación de principales casos de uso
3. Identificar nuevos casos de uso a partir de los existentes
4. Crear descripciones de caso de uso de trazo grueso
5. Definir prioridades y seleccionar casos del primer incremento
6. Escribir los casos de trazo fino y crear prototipos de interfaces

¿Por qué es necesario hacer modelos para el análisis?

Permiten concentrase en las propiedades importantes del sistema.

Favorecen la **comunicación con el usuario**.

Permiten verificar que el **analista ha comprendido** el problema/entorno a abordar.

Principios de los modelos

Objetivo elaborar software, viajar ligero, reproducir el modelo más sencillo, susceptible a cambios, propósito explícito, adaptar los modelos, construir modelos útiles, no ser dogmático respecto de la sintaxis del modelo, hacer caso al instinto, obtener retroalimentación

Principios de los modelos de análisis

Debe entenderse y representarse el **dominio** de información de un problema

Deben entenderse y representarse las **funciones** que debe realizar el software

Deben **dividirse** los modelos de manera que descubran los detalles por capas (o jerárquicamente).

El proceso de análisis debe avanzar desde la **información esencial hacia la implementación en detalle**

Flujos entre un almacén y un proceso y un proceso y un almacén

Un flujo que vaya de un almacén a un proceso representa la lectura de:

• Un único paquete de información.

• Parte de un paquete de información.

• Varios paquetes de información.

• Partes de más de un paquete de información.

Un flujo que vaya de un proceso a un almacén puede representar:

• Adición de uno o más paquetes de información.

• Borrado de uno o más paquetes de información.

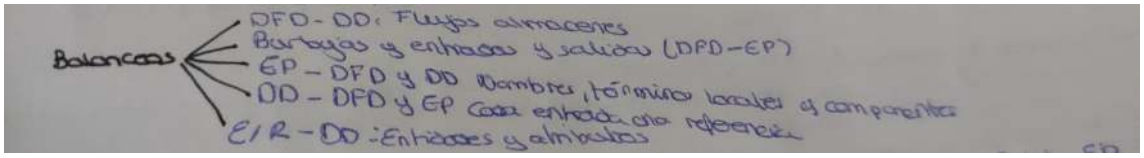
• Actualización de:

- Un único paquete de información.
- Parte de un paquete de información.
- Varios paquetes de información.
- Partes de más de un paquete de información.

Regla genérica de los DFD

6±1 (entre 5 y 7 burbujas), que si no se cumple puede indicar la necesidad de un nivel "intermedio".

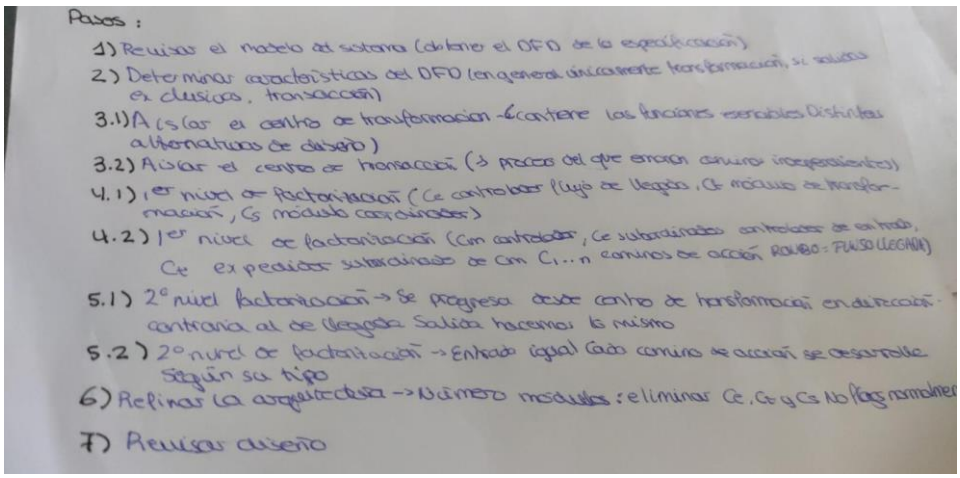
Balanceos



Tipos de 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 global
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.

Pasos del análisis de transformación y el análisis de transacción



Opciones estratégicas de la prueba de humo

1. aborda muchos requerimientos de software,
2. tiene un alto nivel de control
3. es complejo o proclive al error o
4. tiene requerimientos de rendimiento definidos.

Depuración y sus opciones estratégicas

La **depuración** ocurre como consecuencia de las pruebas "exitosas". Es decir, cuando un caso de prueba descubre un error. Es el proceso que se da como resultado de la corrección del error. Las estrategias son fuerza bruta, vuelta atrás, eliminación de causas.

Clases de equivalencia según la condición de entrada

1. Si una condición de entrada especifica un rango, se define una clase de equivalencia válida y dos inválidas.
2. Si una condición de entrada requiere un valor específico, se define una clase de equivalencia válida y dos inválidas.
3. Si una condición de entrada especifica un miembro de un conjunto, se define una clase de equivalencia válida y una inválida.
4. Si una condición de entrada es booleana, se define una clase válida y una inválida.

Patrones de prueba

PairTesting: Patrón orientado a procesos, dos examinadores trabajan en conjunto para diseñar y ejecutar una serie de pruebas.

SeparateTestInterface: Probar cada clase en un sistema orientado a objetos, incluidas "clases internas".

ScenarioTesting: Revisar el software desde el punto de vista del usuario

Pasos de las pruebas OO

1. Las pruebas se inician con la revisión de dichos modelos
2. Una vez generado el código se comenzan con las pruebas de clase que observan una clase colaboradora con otras clases
3. A medida que las clases se juntan con otras formando subsistemas, se aplican pruebas en grupo y en conjunto con enfoque basados en fallo a fin de probar en conjunto las clases colaboradora
4. Finalmente, se hacen casos de uso desarrollados como parte del modelo de requerimiento para probar la validación del Software.

Métodos de prueba OO

1. Cada caso de prueba debe identificarse de manera única y explícita asociado con la clase que se va a probar.
2. Debe establecerse el propósito de la prueba.
3. Debe desarrollarse una lista de pasos de prueba para cada una de ellas, que debe contener:
3.1 Una lista de estados especificados para la clase que se probará

3.2 Una lista de mensajes y operaciones que se probarán como consecuencia de la prueba
3.3 Una lista de excepciones que pueden ocurrir conforme se prueba la clase
3.4 Una lista de condiciones externas
3.5 Información complementaria que ayudará a comprender o a implementar la prueba

Pruebas de estructura superficial y profunda

Cuando se habla de **estructura superficial** se hace referencia a la estructura observable externamente de un programa OO. En lugar de realizar funciones, a los usuarios de muchos sistemas OO se les pueden dar objetos para manipular en alguna forma. Pero las pruebas se basan todavía en tareas de usuario. Capturar estas tareas involucra comprensión, observación y hablar con usuarios representativo.
Cuando se habla de **estructura profunda**, se hace referencia a los detalles técnicos internos de un programa OO. La prueba de estructura profunda se diseña para ejercitar dependencias, comportamientos y mecanismos de comunicación que se establezcan como parte del modelo de diseño para el software OO.

Prueba interclase/de clase múltiple

1. Para cada clase cliente, use la lista de operaciones de clase a fin de generar una serie de secuencias de prueba aleatorias. Las operaciones enviarán mensajes a otras clases servidor.
2. Para cada mensaje generado, determine la clase colaborador y la correspondiente operación en el objeto servidor.
3. Para cada operación en el objeto servidor determine los mensajes que transmite.
4. Para cada uno de los mensajes, determine el siguiente nivel de operaciones que se invocan e incorpore esto en la secuencia de prueba.

5 puntos de vista de la calidad

1. cliente ve calidad cuando se cumplen sus objetivos o metas
2. desde el fabricante que cumpla las especificaciones originales
3. desde el producto con las características propias
4. de acuerdo con el valor
5. desde el precio que tenga un producto

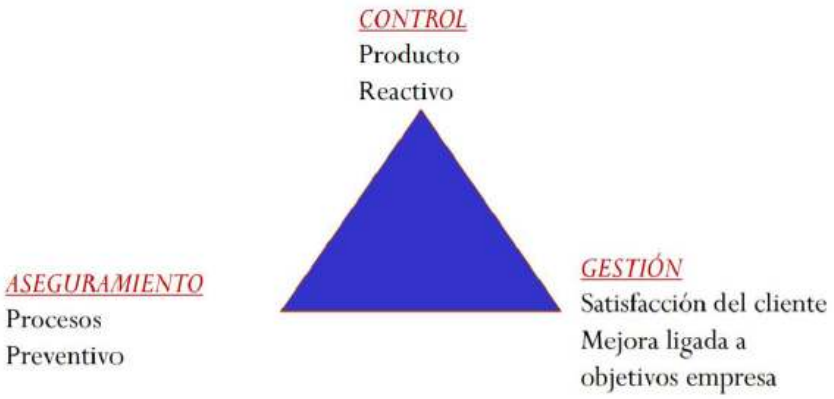
4 principales factores de riesgo en cuanto a la calidad para las empresas

Florecimiento de requisitos, excesiva presión de plazos, baja calidad, costes sobrepasados

Tabla

RESUMEN			
Características principales	**CONTROL DE CALIDAD**	**ASEGURAMIENTO DE LA CALIDAD**	**GESTION 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

Control - aseguramiento - gestión



Control: Se enfoca en encontrar problemas en el producto una vez ocurridos.

Aseguramiento: Se enfoca en asegurar la calidad de los procesos usados para fabricar esos productos, previniendo defectos.

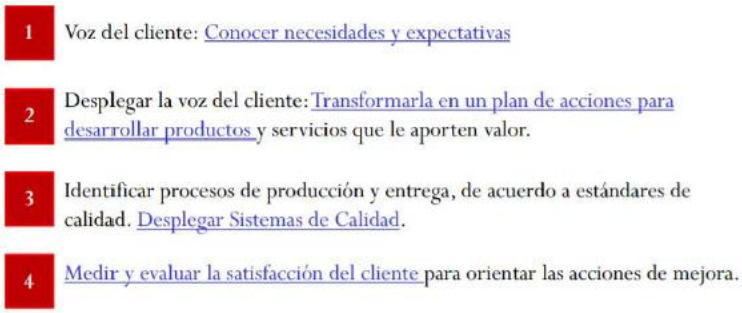
Gestión: Se busca cumplir las expectativas del cliente en base a una mejora continua de los objetivos de la empresa.

Calidad Software de un producto

Áreas de interés de CMMI

Desarrollo, adquisición y servicios

Proceso de creación de valor



Problemas del desarrollo software

• Proyectos fuera de plazo y de 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

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

Actividades del gestor

• Redacción de la propuesta

• Planificación y calendarización del proyecto

• Estimación de costes del proyecto

• Supervisión y revisión del proyecto

• Selección y evaluación del personal

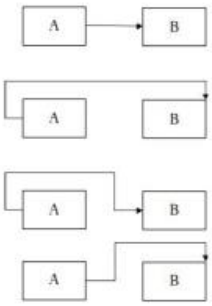
• Redacción y presentación de informes

Definiciones de hito y hamaca

Hito: Tipo de actividad especial que no tiene duración y sirve para indicar un acontecimiento, un momento particular e importante del proyecto. No consume recursos Normalmente se utiliza para describir puntos de control. Se usan mucho para gestionar el trabajo de terceras partes (subcontratas) en el proyecto.
Hamaca: Tipo especial de actividad que mide el tiempo transcurrido entre 2 puntos del proyecto (entre actividades, hitos, etc.). Las restricciones se deben establecer entre las tareas elementales y no entre las hamacas.

Restricciones

- **Restricciones**
  - FC: Fin-Comienzo
  - CF: Comienzo-Fin
  - CC: Comienzo-Comienzo
  - FF: Fin-Fin



Seguimiento del proyecto

A medida que un proyecto avanza es necesario comprobar si todo sucede según lo previsto.

• Comparar los resultados actuales con los planes previstos (línea de base).

• Tomar acciones correctivas cuando existan desviaciones significativas con respecto a los planes previstos.

• Es necesario definir una normativa estándar para el seguimiento de proyectos (mensualmente, trimestralmente, etc.)