Hoja 1 de Ejercicios

Ciclo de Vida del Software

Inicio: Semana del 23 de Enero.
Duración: 1 semana.

1) ¿Para qué vale el estudio de viabilidad? ¿Qué aspectos hay que estudiar?. ¿Cuál es más importante?

El estudio de viabilidad consiste en realizar un análisis previo a la realización de un proyecto para determinar si éste es rentable económicamente y estratégicamente. Sirve para fundamentar la toma de decisión sobre si continuar o no con el proyecto, e identificar los riesgos que conlleva la realización del proyecto. Hay que estudiar los siguientes aspectos:

- *Consideraciones técnicas:*
  - *Riesgo de desarrollo.*
    ¿Se puede desarrollar el sistema de tal forma que las funcionalidades y el rendimiento esperado se consigan dentro de las restricciones impuestas?.
  - *Disponibilidad de recursos humanos.*
    ¿Tenemos el suficiente número de personas, con cualificación adecuada?.
  - *Disponibilidad de recursos informáticos.*
    ¿Tenemos los recursos hardware y software necesarios?.
  - *Tecnología actual.*
    ¿Estamos usando tecnología lo suficientemente probada?.
    ¿Qué tecnologías se requieren para conseguir los requisitos?
    ¿Qué nuevos métodos/algoritmos/técnicas son necesarios? ¿Cuál es el riesgo?
- *Viabilidad económica: Análisis de coste/beneficios. Tener en cuenta los beneficios tangibles (ganancias en dinero) e intangibles (mejor posicionamiento estratégico de la empresa, mejor calidad de diseño, etc.).*
- *Viabilidad legal: Infracción, violación o ilegalidad resultado del desarrollo del sistema*
- *Estudio de diversas alternativas: Evaluación de posibles alternativas al desarrollo del sistema.*

2) Clasifica los siguientes requisitos en funcionales, y no funcionales. Dentro de estos últimos indica el tipo.

1. La aplicación debe incluir una opción de menú para la impresión de todos los listados.
   Funcional.
2. La Base de Datos de la aplicación debe soportar hasta un máximo de 5 millones de registros.
   No Funcional. De recursos.
3. La respuesta de autorización de crédito debe ser en menos de 30 secs, el 90% de las veces.
   No Funcional. De rendimiento.
4. El dispositivo de punto de venta tendrá una pantalla táctil en panel grande y plano. El texto debe ser visible desde un metro.
   No Funcional. De usabilidad/operación.
5. El sistema debe tener una recuperación robusta cuando el acceso a sistemas externos (tales como el inventario, impuestos, etc.) falla.
   No Funcional. De Fiabilidad/Operacional.
6. El sistema deberá incluir versiones de la ayuda en español e inglés.
   Funcional.

3) De los siguientes requisitos, indica cuál de ellos no cumple alguno de los atributos de corrección:

1. "... hasta 15 autobuses se dibujarán dentro de la misma ventana. Si excede el número se utilizará otra ventana diferente."
   OK.

2. "El sistema tendrá una interfaz de usuario sencilla de utilizar."
   ¿Qué es una interfaz sencilla?

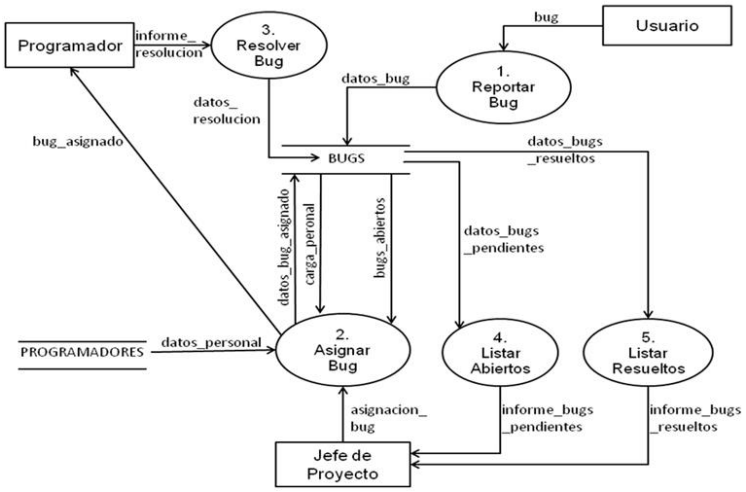
3. "Los usuarios deben escribir su contraseña en un tiempo menor de 15 segundos desde que escribieron su identificación."
   OK, aunque incompleto ¿si no la escriben qué sucede?

4. "El tiempo de respuesta para todos los comandos será menor de 0.1 segundos. El tiempo de respuesta para el comando '*DELETE'* será menor de 5 segundos."
   Inconsistente.

5. "El sistema tendrá un tiempo de respuesta aceptable."
   ¿Qué significa aceptable?

4) Describe el funcionamiento de la siguiente aplicación, modelada mediante un Diagrama de Flujo de Datos. ¿Qué otros aspectos de la aplicación habría que modelar en la fase de requisitos?

Nivel 0



Nivel 1

Solución: El DFD modela una aplicación para el control de errores de una aplicación informática. De tal manera que los usuarios de la aplicación pueden enviar informes de errores. Los errores se guardan en el almacén "BUGS". El Jefe de proyecto asigna la resolución de errores al personal del proyecto, dependiendo de la carga de cada programador. Los programadores reciben notificaciones de los bugs que tienen que resolver e informan cuando los han resuelto. El jefe de proyecto puede además obtener dos listados, con los bugs abiertos y resueltos.

Habría que modelar: La estructura de los datos, mediante un Diagrama Entidad Relación y un Diccionario de datos. También se podría modelar el comportamiento de alguna de las entidades del sistema, como por ejemplo el de un "bug".

5) En el diseño de software, ¿Qué es la modularidad? ¿y la ocultación de información?.

La modularidad consiste la división lógica del programa en secciones lo más independientes posible, que lleven a cabo una funcionalidad cohesiva y definida. Esto hace que el diseño sea más comprensible, manejable, extensible y fácil de probar (se puede probar cada módulo por separado, el ámbito de un error se acota). Además, también conlleva ventajas a nivel de coordinación, ya que se permite el reparto del trabajo entre distintos desarrolladores.

6) ¿Qué son las pruebas de unidad y cuándo hay que hacerlas?

Son pruebas a nivel de función o método. Hay que hacerlas durante la fase de codificación.

7) Si se hacen pruebas de caja blanca ya no hay que hacer pruebas de caja negra. ¿Verdadero o Falso?

Falso.

8) Selecciona el modelo de ciclo de vida más adecuado para los siguientes escenarios.

1. Un software que controle un reactor nuclear con los objetivos de prever posibles fallos, detectar anomalías, recomendar sugerencias de tácticas y realizar un seguimiento de las acciones llevadas a cabo por los operadores. La definición del sistema software es conflictiva y presenta altos riesgos en diferentes etapas del desarrollo. Además, las pruebas son difíciles de realizar, ya que el sistema sólo puede pasar a la fase de explotación (uso) cuando esté completamente probado, por lo que se pretende construir un simulador como ayuda a esta fase.
   Espiral.

2. Un software que realice las gestiones de un departamento de ventas de una editorial donde las funciones están claramente identificadas. El sistema software que se tiene que implementar apenas presenta riesgos, sus fases de desarrollo están bien definidas y tu empresa tiene gran experiencia en el desarrollo de este tipo de aplicaciones
   Cascada.

3. Un software para planificación del stock de vinos que una bodega tiene en distintos almacenes. El sistema utilizará algoritmos de planificación basándose en la demanda del mercado y en el suministro de los proveedores. Esta aplicación no presenta, a priori, grandes riesgos aunque se prevé que puedan surgir requisitos nuevos durante el desarrollo.
   Incremental e Iterativo.

4. Se quiere construir una aplicación que permita acceder a través de Internet a información sobre las obras que hay en un museo. El cliente cree tener muy claros los requisitos de la aplicación. Sin embargo, el Ingeniero de Software no opina lo mismo por lo que quiere verificar los requisitos de usuario lo antes posible, en particular los relacionados con la funcionalidad y la interfaz de usuario. Técnicamente no parece presentar ningún problema.
   Maqueta+cascada, quizá con iteraciones.