Hoja 1 de Ejercicios

Ciclo de Vida del Software

Inicio: Semana del 31 de Enero.
Duración: 1 semana.
Entrega: Opcional. La Semana del 7 de Febrero.

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?

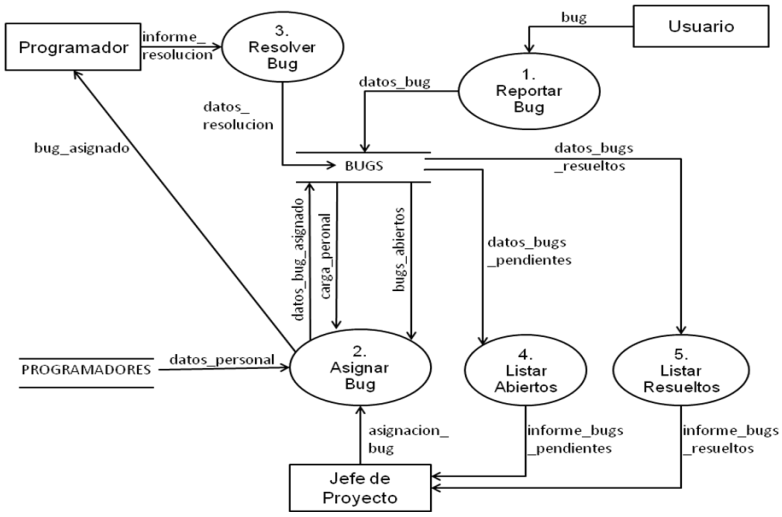
3. "Los usuarios deben escribir su contraseña en un tiempo menor de 15 segundos desde que escribieron su identificación."
   OK, aunque incompleto ¿si no la escriben qué sucede?

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



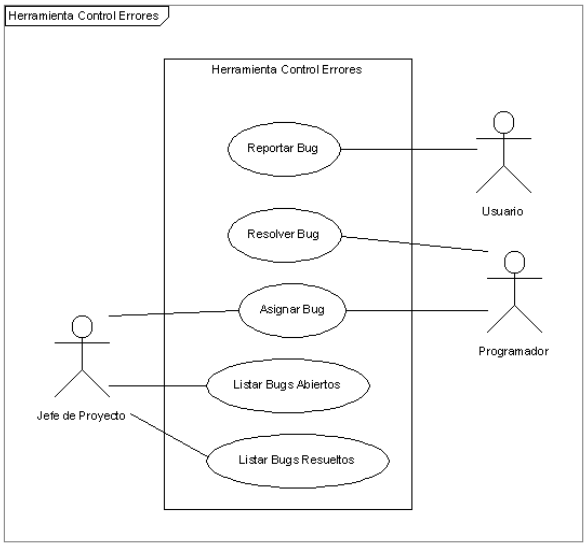
Nivel 1

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) Modela mediante un *Diagrama de Casos de Uso* la aplicación del apartado 4. Describe el escenario "Asignar Bug" en el que el Jefe de Proyecto asigna un bug pendiente a un Programador.

Solución:



CASO DE USO: Asignar Bug

Actor Primario:

Jefe de Proyecto

Interesados y objetivos:

Jefe de Proyecto: Quiere asignar bugs pendientes, para que se resuelvan lo antes posible, para ello, quiere mantener una carga de trabajo adecuada entre los distintos programadores.

Programador: Quiere ser informado de los bugs que necesita resolver de manera completa y lo antes posible.

Precondiciones:

El Jefe de Proyecto se ha identificado y autenticado

Garantía de éxito (Postcondiciones):

El Jefe de Proyecto asigna un bug a un programador, y esto queda registrado en el sistema. El programador recibe una notificación sobre el bug a resolver.

Escenario principal de éxito:

1. El Jefe de Proyecto obtiene la lista de bugs no asignados.
2. El Jefe de Proyecto selecciona un programador de proyecto.
3. El Jefe de Proyecto obtiene la carga de trabajo actual de la persona seleccionada

Se repiten los pasos 2 y 3.

4. El Jefe de Proyecto asigna uno de los bugs no asignados al programador seleccionado.
5. El sistema marca como asignado el bug, se actualiza la carga de trabajo del programador seleccionado y se le manda una notificación.

Se repite el paso 1 hasta que el Jefe de Proyecto sale de esta opción.

Extensiones/Flujos alternativos:

a*. En cualquier momento, el sistema falla.

1a. No hay bugs pendientes de asignar.

1a.1. El Jefe de proyecto no puede asignar ningún bug.

Requisitos especiales:

- Posibilidad de internacionalización del texto de aviso al programador, debido a equipos de trabajo multiculturales.
- Posibilidad de navegar eficientemente por listas del orden de cientos de bugs.

Lista de variaciones de tecnología y datos:

La notificación al programador podría ser por e-mail, SMS, u otras tecnologías de mensajería instantánea.

Frecuencia de ocurrencia:

Del orden de decenas de veces al día.

Temas abiertos:

¿Cómo se realizará la recuperación en caso de fallos?

¿Se podría configurar una carga de trabajo/número de bugs asignados máxima por programador?

¿Se podría considerar un sistema semi-automático de asignación?

6) La cadena de restauración "*las 1001 tostas*" te encarga la realización de una aplicación informática para la automatización del servicio a los clientes de sus distintos locales. El funcionamiento es el siguiente:

Cuando un cliente entra en el establecimiento, un camarero le asigna una mesa. Una vez en la mesa, el cliente rellena un formulario con las tostas y bebidas que desea. Ahora esta selección se hace con lápiz y papel, pero se pretende que el cliente lo haga mediante un dispositivo con pantalla táctil. El cliente puede seleccionar bien tostas específicas de entre una lista de 100, o bien un número de entre "las más populares", o de "la selección del chef". El sistema debe pues presentar cuáles son las 20 tostas más pedidas de ese día, así como las 12 tostas seleccionadas por el chef. Si un cliente quiere más comida o bebida realiza una nueva selección y el proceso se repite. Cuando el cliente realiza la orden, se le presenta el precio actual de su consumición.

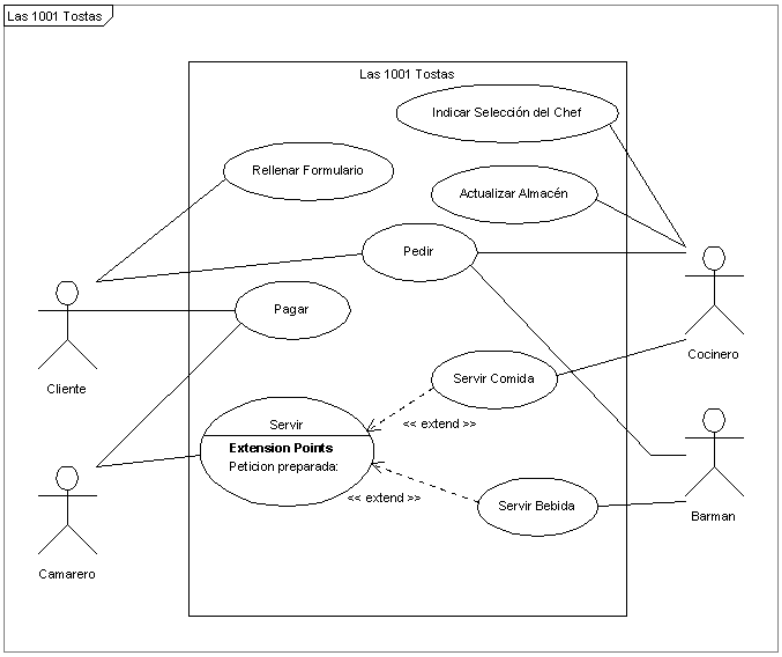
La información sobre la comida seleccionada se por cada cliente se presenta inmediatamente al cocinero (en un dispositivo instalado en la cocina), y el de la bebida al barman (en un dispositivo instalado en la barra). Cuando la comida o la bebida están listas, el cocinero o el barman lo indican en el sistema. Este le presenta un aviso al camarero asignado a la mesa correspondiente (que está equipado de un dispositivo móvil), que se encargan de llevarlas a la mesa.

El sistema debe además controlar el stock de alimentos en almacén. El cocinero se encarga de introducir dos veces al día la información sobre los nuevos alimentos adquiridos o los utilizados. Cuando un ingrediente se agota en el almacén, las correspondientes tostas no deben poder ser seleccionadas por los clientes.

Finalmente, cuando un cliente se va, lo indica mediante el dispositivo de la mesa. Esto hace que se imprima su recibo en la caja central, y se le presenta un aviso al camarero asignada a esas mesas, que le cobra (sólo se acepta dinero en efectivo). En cualquier momento antes de abandonar su mesa, de manera opcional el cliente puede rellenar un formulario para evaluar el servicio recibido. Este formulario ha de ser personalizable.

a) Realiza un diagrama de casos de uso.
b) Describe el escenario correspondiente a realizar una petición de comida/bebida por parte del cliente. Incluye tanto el flujo principal y los alternativos si los hubiera.

Solución:



CASO DE USO: Pedir

Actor Primario:

- Cliente.
- Barman.
- Cocinero.

Interesados y objetivos:

- ***Cliente***: quiere realizar una petición de manera rápida, quiere obtener información sobre tostas disponibles, recomendaciones, y poder ver el precio actual de la petición.
- ***Barman***: quiere obtener de manera rápida descripciones precisas de las bebidas solicitadas.
- ***Cocinero***: quiere obtener de manera rápida descripciones precisas de las tostas solicitadas.
- ***Cadena de restauración***: quiere que el cliente quede satisfecho, y que se anote el precio a pagar.

Pre-condiciones:

- El cliente ha entrado en el local, un camarero le ha asignado una mesa.

Post-condiciones:

- Se le presenta el total de la consumición, se actualiza la cuenta de tostas solicitadas en el día (si fuera necesario). El cocinero recibe la petición de tostas en caso necesario, el barman recibe la petición de bebidas en caso necesario.

Escenario Principal:

1. El cliente activa el dispositivo de la mesa.
2. El sistema calcula las tostas disponibles de las 100 totales.
3. El sistema calcula las 20 tostas más pedidas.
4. El sistema presenta la información de tostas y bebidas disponibles.
5. El cliente realiza una selección (y el sistema le va presentando el precio actual según se va seleccionando).
6. El cliente efectúa la petición.
7. El sistema calcula el precio de su consumición y se lo presenta al usuario.
8. El sistema actualiza la contabilidad de tostas pedidas en el día.
9. El sistema manda un aviso a la barra con la información de las bebidas.
10. El sistema manda un aviso a cocina con la información de las tostas.

Escenarios Alternativos:

a* En cualquier momento el sistema falla:

1. Se manda un aviso al camarero asignado a la mesa.

2. Se presenta un aviso al cliente y el dispositivo se desactiva

6a. El cliente cancela la selección y el pedido.

1. El sistema presenta la pantalla de bienvenida.

6b. El cliente no efectúa su petición y se marcha, o permanece inactivo más de 15 minutos.

1. El sistema se desactiva a los 15min. de inactividad.

Requisitos Especiales:

- Dispositivos en mesas con pantalla táctil y cierta protección contra golpes, derrames de bebidas, manchas de alimentos, etc.
- Control de mal-uso por parte de clientes.

Listas de variaciones de tecnología y datos:

1. Dispositivo manejable con señalador.

Frecuencia de ocurrencia:

Puede ser casi continua.

Temas abiertos:

7) 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, y que se permite el reparto del trabajo entre distintos desarrolladores.

8) ¿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.

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

Falso.

10) 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.