TECNICAS INTELIGENTES PARA PROCESOS DE DESARROLLO

CONTENIDO

1 INTRODUCCIÓN A LAS FASES DEL DESARROLLO DE SISTEMAS

Las cuatro actividades básicas del proceso de especificación, desarrollo, validación y evolución se organizan de forma distinta en diferentes procesos del desarrollo. En el enfoque en cascada, están organizadas en secuencia, mientras que en el desarrollo evolutivo se entrelazan. Cómo se llevan a cabo estas actividades depende del tipo de software, de las personas y de la estructura organizacional implicada. No hay una forma correcta o incorrecta de organizar estas actividades.

1.1 ESPECIFICACIÓN DEL SOFTWARE

La especificación del software o ingeniería de requerimientos es el proceso de comprensión y definición de qué servicios se requieren del sistema y de identificación de las restricciones de funcionamiento y desarrollo del mismo. La ingeniería de requerimientos es una etapa particularmente crítica en el proceso del software ya que los errores en esta etapa originan inevitablemente problemas posteriores en el diseño e implementación del sistema.

En la Ilustración 1 se muestra el proceso de ingeniería de requerimientos. Éste conduce a la producción de un documento de requerimientos, que es la especificación del sistema. Normalmente en este documento los requerimientos se presentan en dos niveles de detalle. Los usuarios finales y los clientes necesitan una declaración de alto nivel de los requerimientos, mientras que los desarrolladores del sistema necesitan una especificación más detallada de éste.

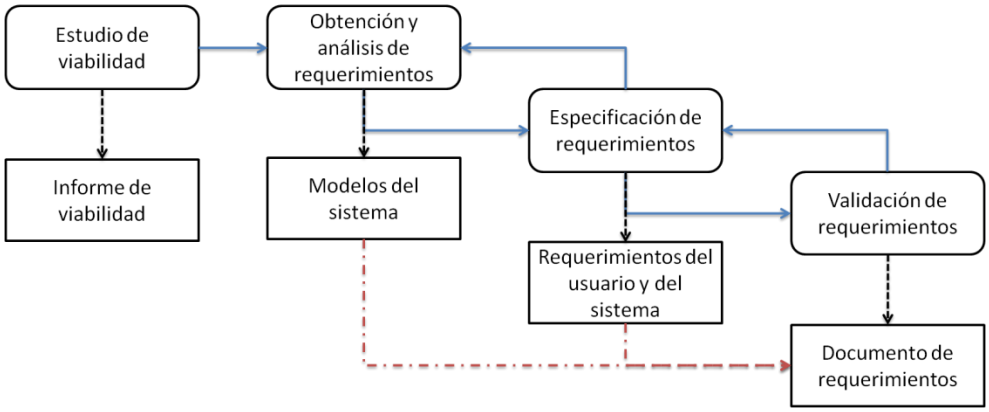

Ilustración 1. El proceso de la ingeniería de requerimientos.

Existen cuatro fases principales en el proceso de ingeniería de requerimientos:

1) *Estudio de viabilidad.* Se estima si las necesidades del usuario se pueden satisfacer con las tecnologías actuales de software y hardware.
2) *Obtención y análisis de requerimientos.* Es el proceso de obtener los requerimientos del sistema por medio de la observación de los sistemas existentes, discusiones con los usuarios potenciales y proveedores, el análisis de tareas, etc.
3) *Especificación de requerimientos.* Es la actividad de traducir la información recopilada durante la actividad de análisis en un documento que define un conjunto de requerimientos.
4) *Validación de requerimientos.* Esta actividad comprueba la veracidad, consistencia y completitud de los requerimientos.

Por supuesto, las actividades en el proceso de requerimientos no se llevan a cabo de forma estrictamente secuencial. El análisis de requerimientos continúa durante la definición y especificación, y a lo largo del proceso surgen nuevos requerimientos.

1.1.1 REQUERIMIENTOS

Los requerimientos se distinguen utilizando la denominación de:

1) *Requerimientos de usuario* que son declaraciones, en lenguaje natural y en diagramas, de los servicios que se espera que el sistema proporcione y de las restricciones bajo las cuales debe funcionar el sistema. El requerimiento del usuario es más abstracto.
   Un requerimiento de usuario se expande en varios requerimientos del sistema.

Un ejemplo de un requerimiento de usuario es el siguiente:

1. LIBSYS controlará todos los datos requeridos por las agencias que licencian los derechos de autor en el Reino Unido y en otras partes.

2) *Requerimientos de sistema* que establecen con detalle las funciones, servicios y restricciones operativas del sistema. El documento de requerimientos del sistema debe ser preciso. Los requerimientos del sistema añaden detalle. Debe definir exactamente qué es lo que se va a implementar. Puede ser parte del contrato entre el comprador del sistema y los desarrolladores del software. Un ejemplo de una especificación de un requerimiento del sistema es:

1.1 Al hacer una petición de un documento del LIBSYS, el solicitante se presentará con un formulario que registre los detalles del usuario y de la petición hecha.
1.2 El formulario de petición del LIBSYS será almacenado en el sistema durante cinco años desde la fecha de la petición.
1.3 Todos los formularios de peticiones del LIBSYS se deben indexar por usuario, por el nombre del material solicitado y por el proveedor de la petición.
1.4 El LIBSYS mantendrá un fichero en el que se registrarán todas las peticiones que se han hecho al sistema.
1.5 Para el material donde se aplican los derechos de préstamo del autor, los detalles del préstamo serán enviados mensualmente a las agencias que licencian los derechos del autor que se han registrado con LIBSYS.

Los requerimientos de usuario son empleados por:

- Administradores clientes
- Usuarios finales del sistema
- Ingenieros clientes
- Administradores contratistas
- Arquitectos del sistema

Los requerimientos del sistema son usados por:

- Usuarios finales del sistema
- Ingenieros clientes
- Arquitectos del sistema
- Desarrolladores del software

Los requerimientos del sistema, se clasifican en Funcionales y No funcionales

1.1.2 REQUERIMIENTOS FUNCIONALES

Los requerimientos funcionales de un sistema describen lo que el sistema debe hacer. Estos requerimientos dependen del tipo de software que se desarrolla, de los posibles usuarios del software y del enfoque general tomado por la organización al redactar requerimientos.

Los requerimientos funcionales del sistema describen con detalle la función de éste, sus entradas y salidas, excepciones, etc.

Un ejemplo de requerimientos funcionales para el sistema bibliotecario universitario LIBSYS son los siguientes:

1. El usuario deberá tener la posibilidad de buscar en el conjunto inicial de la base de datos o seleccionar un subconjunto de ella.
2. El sistema deberá proporcionar visores adecuados para que el usuario lea documentos en el almacén de documentos.
3. A cada pedido se le deberá asignar un identificador único (ID_PEDIDO), que el usuario podrá copiar al área de almacenamiento permanente de la cuenta.

1.1.3 REQUERIMIENTOS NO FUNCIONALES

Los requerimientos no funcionales son aquellos requerimientos que no se refieren directamente a las funciones específicas que proporciona el sistema, sino a las propiedades emergentes (limitaciones y restricciones) de éste, que pueden referirse a atributos del sistema como:

1. El desempeño: la confiabilidad, el tiempo de respuesta, la capacidad de almacenamiento, la seguridad.
2. La escalabilidad: que es la capacidad de permitir en el futuro el desarrollo de nuevas funcionalidades, modificarlas o eliminarlas.
3. Facilidad de uso: Fácil uso de entrenamiento, mensajes de error, no podrá cerrarse una operación hasta no concluir las demás.
4. Seguridad: El sistema debe estar restringido, rechazar accesos o modificaciones no autorizadas.
5. Validación: Obligatoriedad de campos, manejo de tipos de datos, validación de contraseñas.
6. Otros como: Flexibilidad, Mantenibilidad, Instalación, Operatividad, Hardware, Interfaz e Interoperabilidad.

Ejemplos de requerimientos no funcionales para el sistema bibliotecario universitario LIBSYS son los siguientes:

1. *La interfaz de usuario de LIBSYS, se implementará como HTML simple, sin marcos o applets java.*
2. *En el proceso del desarrollo del sistema los documentos a entregar deberán ajustarse al proceso y a los productos a entregar, definidos en XYZCo-SP-STAN-95.*
3. *El sistema no deberá revelar al personal de la biblioteca que lo utilice, ninguna información de los usuarios del sistema aparte de su nombre y número de referencia de la biblioteca.*

1.2 DISEÑO E IMPLEMENTACIÓN DEL SOFTWARE

La etapa de implementación del desarrollo de software es el proceso de convertir una especificación del sistema en un sistema ejecutable. Implica los procesos de diseño y programación de software.

Un diseño de software es una descripción de la estructura del software que se va a implementar, los datos que son parte del sistema, las interfaces entre los componentes del sistema y los algoritmos utilizados.

En la Ilustración 1 se muestra un modelo del proceso de diseño, donde las etapas son secuenciales, pero normalmente se entrelazan en la práctica.

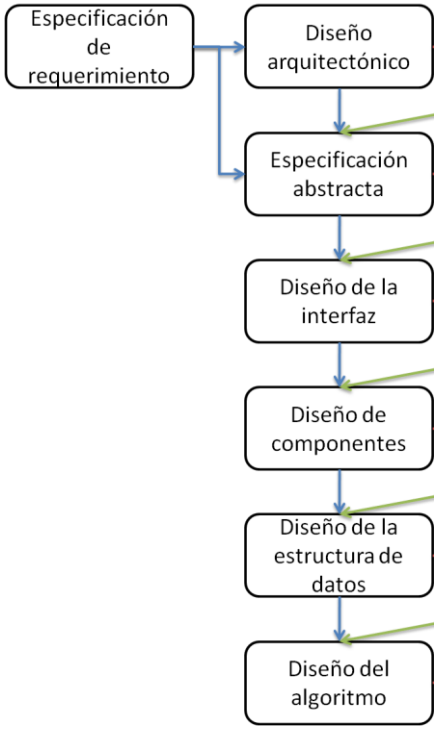

Ilustración 1. Modelo general del proceso de diseño.

Las actividades específicas del proceso de diseño son:

1. *Diseño arquitectónico.* Los subsistemas que forman el sistema y sus relaciones se identifican y documentan.
2. *Especificación abstracta.* Para cada subsistema se produce una especificación abstracta de sus servicios (¿qué hace?) y las restricciones bajo las cuales debe funcionar.
3. *Diseño de la interfaz.* Para cada subsistema se diseña y documenta su interfaz con otros subsistemas.
4. *Diseño de componentes.* Se asignan servicios a los componentes y se diseñan sus interfaces.
5. *Diseño de la estructura de datos.* Se diseña en detalle y especifica la estructura de datos utilizada en la implementación del sistema.
6. *Diseño de algoritmos.* Se diseñan en detalle y especifican los algoritmos utilizados para proporcionar los servicios.

Éste es un modelo general del proceso de diseño, y los procesos reales y prácticos pueden adaptarlo de diversas maneras.

Cuando se utilizan métodos ágiles de desarrollo, las salidas del proceso de diseño no serán documentos de especificación separados, sino que estarán representadas en el código del programa.

Un enfoque opuesto es dado por los *métodos estructurados* que se basan en la producción de modelos gráficos del sistema y código automáticamente generado desde estos modelos.

Un método estructurado incluye un modelo del proceso de diseño, notaciones para representar el diseño, formatos de informes, reglas y pautas de diseño. En la práctica los métodos estructurados son realmente notaciones estándar que comprenden prácticas aceptables.

El desarrollo de un programa para implementar el sistema se sigue de forma natural del proceso de diseño. Aunque algunos programas, como los de los sistemas críticos de seguridad, se diseñan en detalle antes de que se inicie cualquier implementación.

Las herramientas CASE se pueden utilizar para generar un programa esqueleto a partir de un diseño. Esto incluye código para definir e implementar las interfaces, y en muchos casos el desarrollador sólo necesita agregar detalles del funcionamiento de cada componente del programa.

Normalmente, los programadores llevan a cabo algunas pruebas del código que han desarrollado. Esto muestra defectos en el programa que se deben eliminar del mismo. Esto se denomina *depuración*. Las pruebas y la depuración de defectos son procesos diferentes. Las pruebas establecen la existencia de defectos. La depuración comprende la localización y corrección de estos defectos (Ilustración 2).

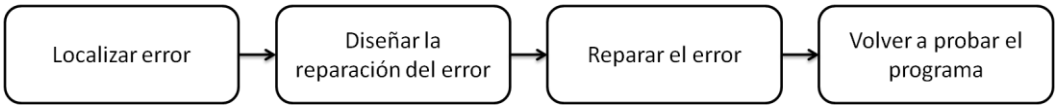

Ilustración 2. El proceso de depuración.

1.3 VALIDACIÓN DEL SOFTWARE

La validación o, la verificación y validación (V&V) se utiliza para mostrar que el sistema se ajusta a su especificación y que cumple con las expectativas del usuario que lo comprará. Implica procesos de comprobación, como las inspecciones y revisiones, en cada etapa del proceso del software desde la definición de requerimientos hasta el desarrollo del programa.

Los sistemas no se deben probar como una simple unidad monolítica. Un proceso de pruebas se integra por tres etapas en el cual se prueban los componentes del sistema, la integración del sistema y el sistema con los datos del cliente. Cuando se descubren defectos el programa debe depurarse y esto puede requerir la repetición de otras etapas del proceso de pruebas. Por lo tanto el proceso es iterativo y se retroalimenta tanto de las últimas etapas como de la primera parte del proceso.

Las etapas del proceso de pruebas son:

1. *Prueba de componentes (o unidades).* Se prueban los componentes individuales para asegurarse de que funcionan correctamente.
2. *Prueba del sistema.* Los componentes se integran para formar el sistema. Este proceso comprende encontrar errores que son el resultado de interacciones no previstas entre los componentes y su interfaz.
3. *Prueba de aceptación.* El sistema se prueba con los datos proporcionados por el cliente más que con datos de prueba simulados.

Un equipo independiente de probadores debe trabajar a partir de planes de prueba que se desarrollan desde la especificación y diseño del sistema. La Ilustración 1 muestra cómo los planes de prueba son el vínculo entre las actividades de prueba y de desarrollo.

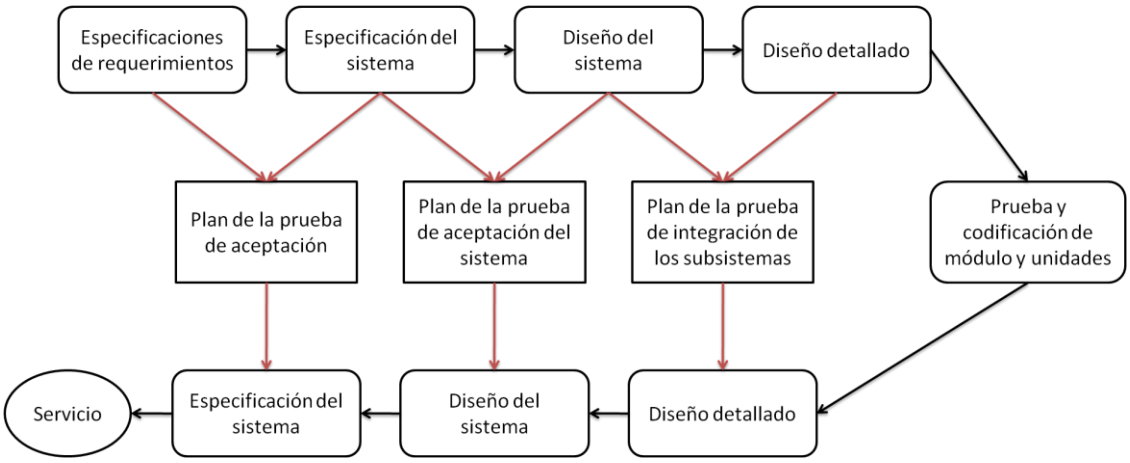

Ilustración 1. Las fases de prueba en el proceso de software.

La prueba de aceptación algunas veces se denomina prueba **alfa**. El proceso de prueba alfa continúa hasta que el desarrollador del sistema y el cliente acuerdan que el sistema que se va a entregar es una implementación aceptable de los requerimientos del sistema.

Cuando un sistema se va a comercializar como un producto de software, a menudo se utiliza un proceso de prueba denominado prueba **beta**. Ésta comprende la entrega de un sistema a un número potencial de clientes que acuerdan utilizarlo, los cuales informan de los problemas a los desarrolladores del sistema. Esto expone el producto a un uso real y detecta los errores no identificados por los constructores del sistema.

1.4 EVOLUCIÓN DEL SOFTWARE

Después de que los sistemas hayan sido desarrollados, inevitablemente han de sufrir cambios si tiene que seguir siendo útiles. Una vez que el software comienza a utilizarse, surgen nuevos requerimientos y los requerimientos existentes cambian. Los cambios en los negocios a menudo generan nuevos requerimientos para el software existente. Algunas partes del software tienen que modificarse para corregir errores encontrados en su funcionamiento, adaptarlo a una nueva plataforma y mejorar su rendimiento u otras características no funcionales. El desarrollo de software, por lo tanto, no se detiene cuando un sistema es entregado, sino que continúa durante el tiempo de vida del sistema.

La evolución del software es importante debido a que las organizaciones actualmente son completamente dependientes de sus sistemas de software y han invertido mucho capital en ellos. Los sistemas de software son activos críticos de negocio y las organizaciones deben invertir en los cambios del sistema para mantener el valor de estos activos.

Los cambios posteriores al desarrollo no están relacionados sólo con la reparación de defectos. La mayoría de los cambios son una consecuencia de que se generan nuevos requerimientos como respuesta a cambios en el negocio y en las necesidades de los usuarios.

1.4.1.1 MANTENIMIENTO DEL SOFTWARE

El mantenimiento del software es el proceso general de cambiar un sistema después de que ha sido entregado. El término se aplica normalmente a software desarrollado a la medida. Los cambios realizados al software pueden ser simplemente la corrección de errores de código, o correcciones más extensas como errores de diseño o mejoras significativas de especificación o nuevos requerimientos. Los cambios se implementan modificando los componentes del sistema existente y añadiendo nuevos componentes al sistema donde sea necesario.

Existen tres tipos de mantenimiento de software:

1. *Mantenimiento para reparar defectos de software.* Pueden ser errores de código, de diseño, o de requerimientos.
2. *Mantenimiento para adaptar el software a diferentes entornos operativos.* Indispensable cuando cambia algún aspecto del entorno del sistema, como el hardware o el sistema operativo.
3. *Mantenimiento para añadir o modificar las funcionalidades del sistema.* Es necesario cuando los requerimientos del sistema cambian debido a condiciones organizacionales.

1.4.1.2 EVOLUCIÓN DE SISTEMAS HEREDADOS

Las organizaciones que cuentan con un presupuesto limitado para mantener y actualizar sus sistemas heredados tienen que decidir cómo obtener los mejores beneficios a su inversión. Esto significa que deben evaluar objetivamente sus sistemas heredados y decidir cuál es la estrategia más adecuada para la evolución de esos sistemas. Existen cuatro opciones estratégicas:

1. *Desechar completamente el sistema.* Se elige esta opción cuando el sistema heredado no constituye una contribución efectiva para los procesos de negocio.
2. *Dejar el sistema sin cambios y continuar con un mantenimiento regular.* Esta opción es ideal cuando el sistema aún es necesario, pero es estable y no hay muchas peticiones de cambio.
3. *Hacer reingeniería del sistema para mejorar su mantenibilidad.* Esta opción se elige cuando la calidad del sistema se ha degradado por los cambios continuos y necesarios.

4. *Reemplazar todo o parte del sistema con un nuevo sistema.* Esta opción es viable cuando algún factor como el hardware implica que el sistema antiguo no pueda continuar en funcionamiento o cuando el costo de un nuevo sistema sea razonable.

Por lo tanto, a partir de estas opciones se puede observar que existen cuatro clases de sistemas:

1. *Baja calidad/bajo valor de negocio.* Mantener estos sistemas es caro y no representan beneficios a la empresa.
2. *Baja calidad/alto valor de negocio.* Son importantes para el negocio, pero muy caros de mantener. Se puede aplicar reingeniería para mejorar su calidad o deberían ser reemplazados.
3. *Alta calidad/bajo valor de negocio.* No contribuyen mucho al negocio, pero no son caros de mantener. No es viable reemplazar estos sistemas, y si es necesario realizar cambios costosos se deben de desechar estos sistemas.
4. *Alta calidad/alto valor de negocio.* Estos sistemas se deben mantener en funcionamiento.

Para evaluar el valor de negocio de un sistema, se tiene que identificar a los interesados del sistema, y plantearles las siguientes cuestiones:

1. El uso del sistema
2. Los procesos de negocio que están soportados
3. La confiabilidad del sistema
4. Las salidas del sistema