1. DISEÑO Y REALIZACIÓN DE PRUEBAS.

1 INTRODUCCIÓN

Las pruebas de software consisten en verificar y validar un producto software antes de su puesta en marcha. Constituyen una de las etapas del desarrollo de software, y básicamente consiste en probar la aplicación construida. Se integran dentro de las diferentes fases del ciclo de vida del software dentro de la ingeniería del software.

Durante la prueba del software se realizarán tareas de verificación y validación del software:

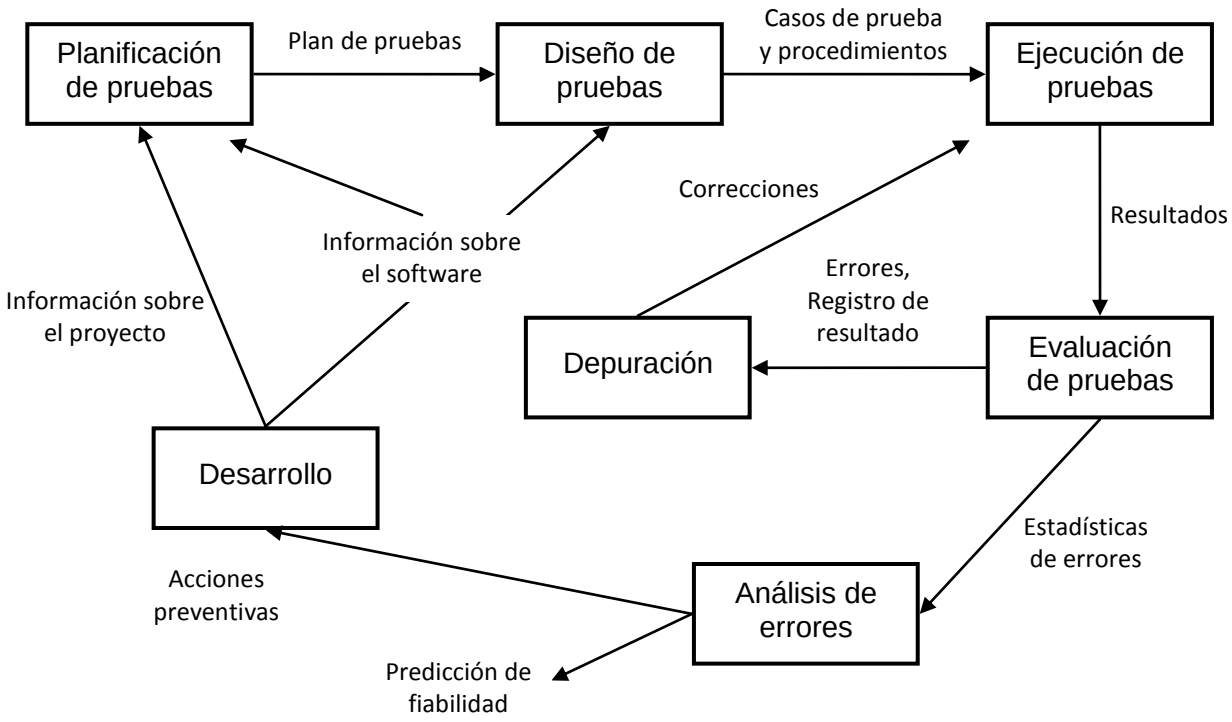
- La **verificación**: Se refiere al conjunto de actividades que tratan de comprobar si se está construyendo el producto correctamente, es decir, si el software implementa correctamente una función específica.

- La **validación**: Se refiere al conjunto de actividades que tratan de comprobar si el producto es correcto, es decir, si el software construido se ajusta a los requisitos del cliente.

Las recomendaciones para las pruebas son las siguientes:

- Cada prueba debe definir los resultados de salida esperados.
- Un programador o una organización debe evitar probar sus propios programas.
- Es necesario revisar los resultados de cada prueba en profundidad.
- Las pruebas deben incluir datos de entrada válidos y esperados, así como no válidos e inesperados.
- Centrar las pruebas en dos objetivos: 1) comprobar si el software no hace lo que debe hacer y 2) comprobar si el software hace lo que no debe hacer.
- Evitar hacer pruebas que no estén documentadas ni diseñadas con cuidado.
- No planear pruebas asumiendo que no se encontrarán errores.
- La probabilidad de encontrar errores en una parte del software es proporcional al número de errores ya encontrados.
- Las pruebas son una tarea creativa.

El flujo de proceso para probar el software se muestra en la figura y es el siguiente:



1. En primer lugar, hay que generar un plan de pruebas partiendo de la documentación del proyecto y de la documentación sobre el proyecto a probar.

2. A partir del plan se diseñan las pruebas. Se identifican las técnicas a utilizar para probar el software.

3. Generación de los casos de prueba. Se han de confeccionar los distintos casos de prueba según la técnica o técnicas identificadas previamente.

4. Definición de los procedimientos de la prueba. Hay que especificar cómo se va a llevar a cabo el proceso, quién lo va a realizar, cuándo, etc.

5. Ejecución de las pruebas aplicando los casos de prueba generados previamente.

6. Evaluación. Se identifican los posibles errores producidos al comparar los resultados obtenidos en la ejecución con los esperados. Es necesario realizar un informe con el resultado de ejecución de las pruebas, qué casos de prueba pasaron satisfactoriamente, cuáles no, y qué fallos se detectaron.

7. Depuración. Trata de localizar y corregir los errores. Si se corrige un error, se debe volver a probar el software para ver que se ha resuelto el problema. Si no se consigue localizar un error puede ser necesario realizar más pruebas para obtener más información.

8. El análisis de errores puede servir para predecir la fiabilidad del software y mejorar los procedimientos de desarrollo.

2 TÉCNICAS DE DISEÑO DE CASOS DE PRUEBA

Un caso de prueba es un conjunto de entradas, condiciones de ejecución y resultados esperados, desarrollado para conseguir un objetivo particular o condición de prueba. Para llevar a cabo un caso de prueba, es necesario definir las precondiciones y postcondiciones, identificar unos valores de entrada y conocer el comportamiento que debería tener el sistema ante dichos valores. Tras realizar ese análisis e introducir dichos datos en el sistema, se observa si su comportamiento es el previsto o no y por qué. De este forma, se determina si el sistema ha pasado o no la prueba.

Para llevar a cabo el diseño de casos de prueba se utilizan dos técnicas o enfoques: prueba de caja blanca y prueba de caja negra. Estas pruebas no son excluyentes y se pueden combinar para descubrir diferentes tipos de errores.

2.1 Pruebas de caja blanca

También se las conoce como pruebas **estructurales** o de caja de cristal. Se basan en el minucioso examen de los detalles procedimentales del código de la aplicación. Mediante esta técnica se pueden obtener casos de prueba que:

- Garanticen que se ejecutan al menos una vez todos los caminos independientes de cada módulo.
- Ejecuten todas las sentencias al menos una vez.
- Ejecuten todas las decisiones lógicas en su parte verdadera y en su parte falsa.
- Ejecuten todos los bucles en sus límites.
- Utilicen todas las estructuras de datos internas para asegurar su validez.

Una de las técnicas utilizadas para desarrollar los casos de prueba de caja blanca es la prueba del camino básico.

2.2 Pruebas de caja negra

Estas pruebas se llevan a cabo sobre la interfaz del software, no hace falta conocer la estructura interna del programa ni su funcionamiento. Se pretende obtener casos de prueba que demuestren que las funciones del software son operativas, es decir, que las salidas que devuelve la aplicación son las esperadas en función de las entradas que se proporcionen.

A este tipo de pruebas también se les llama pruebas de **comportamiento**. El sistema se considera como una caja negra cuyo comportamiento solo se puede determinar estudiando las entradas y las salidas que devuelve en función de las entradas suministradas.

Con este tipo de pruebas se intenta encontrar errores de las siguientes categorías:

- Funcionalidades incorrectas o ausentes.
- Errores de interfaz.
- Errores en estructuras de datos o en accesos a bases de datos externas.
- Errores de rendimiento.
- Errores de inicialización y finalización.

Existen diferentes técnicas para confeccionar los casos de prueba de caja negra, algunos son: clases de equivalencia, análisis de los valores límite, métodos basados en grafos, pruebas de comparación, etc.

3 ESTRATEGIAS DE PRUEBAS DEL SOFTWARE

La estrategia de prueba del software se puede ver en el contexto de una espiral:

- En el vértice de la espiral comienza la **prueba de unidad**. Se centra en la unidad más pequeña de software, el módulo tal como está implementado en código fuente.
- La prueba avanza para llegar a la **prueba de integración**. Se toman los módulos probados mediante la prueba de unidad y se construye una estructura de programa que esté de acuerdo con lo que dicta el diseño. El foco de atención es el diseño.
- La espiral avanza llegando a la **prueba de validación** (o de aceptación). Prueba del software en el entorno real de trabajo con intervención del usuario final. Se validan los requisitos establecidos como parte del análisis de requisitos del software, comparándolos con el sistema que ha sido construido.
- Finalmente se llega a la **prueba del sistema**. Verifica que cada elemento encaja de forma adecuada y se alcanza la funcionalidad y rendimiento total. Se prueba como un todo el software y otros elementos del sistema.

3.1 Prueba de unidad

En este nivel se prueba cada unidad o módulo con el objetivo de eliminar errores en la interfaz y en la lógica interna. Esta actividad utiliza técnicas de caja negra y caja blanca, según convengan para que se pueda probar. Se realizan pruebas sobre:

- La interfaz del módulo, para asegurar que la información fluye adecuadamente.
- Las estructuras de datos locales, para asegurar que mantienen su integridad durante todos los pasos de ejecución del programa.

- Las condiciones límite, para asegurar que funciona correctamente en los límites establecidos durante el proceso.

- Todos los caminos independientes de la estructura de control, con el fin de asegurar que todas las sentencias se ejecutan al menos una vez.

- Todos los caminos de manejo de errores.

3.2 Prueba de integración

En este tipo de prueba se observa cómo interaccionan los distintos módulos. Se podría pensar que esta prueba no es necesaria, ya que si todos los módulos funcionan por separado, también deberían funcionar juntos. Realmente el problema está aquí, en comprobar si funcionan juntos.

Existen dos enfoques fundamentales para llevar a cabo las pruebas:

- Integración **no incremental** o big bang. Se prueba cada módulo por separado y luego se combinan todos de una vez y se prueba todo el programa completo. En este tipo de enfoque se encuentran gran cantidad de errores y la corrección se hace difícil.

- Integración **incremental**. El programa completo se va construyendo y probando en pequeños segmentos, en este caso los errores son más fáciles de localizar. Se dan dos estrategias: Ascendente y Descendente. En la integración **Ascendente** la construcción y prueba del programa empieza desde los módulos de niveles más bajos de la estructura del programa. En la **Descendente** la integración comienza en el módulo principal, moviéndose hacia abajo por la jerarquía de control.

3.3 Prueba de validación

La validación se consigue cuando el software funciona de acuerdo con las expectativas razonables del cliente definidas en el documento de especificación de requisitos del software o ERS. Se llevan a cabo una serie de pruebas de caja negra que demuestran la conformidad con los requisitos. Las técnicas a utilizar son:

- **Prueba Alfa**: Se lleva a cabo por el cliente o usuario en el lugar de desarrollo. El cliente utiliza el software de forma natural bajo la observación del desarrollador que irá registrando los errores y problemas de uso.

- **Prueba Beta**: Se lleva a cabo por los usuarios finales del software en su lugar de trabajo. El desarrollador no está presente. El usuario registra todos los problemas que encuentra reales y/o imaginarios, e informa al desarrollador en los intervalos definidos en el plan de pruebas. Como resultado de los problemas informados, el desarrollador lleva a cabo las modificaciones y prepara una nueva versión del producto.

3.4 Prueba del sistema

La prueba del sistema está formada por un conjunto de pruebas cuya misión es ejercitar profundamente el software. Son las siguientes:

- **Prueba de recuperación**. En este tipo de prueba se fuerza el fallo del software y se verifica que la recuperación se lleva a cabo apropiadamente.

- **Prueba de seguridad**. Esta prueba intenta verificar que el sistema está protegido contra accesos ilegales.

- **Prueba de resistencia (stress).** Trata de enfrentar el sistema con situaciones que demandan gran cantidad de recursos, por ejemplo, diseñando casos de prueba que requieran el máximo de memoria, incrementando la frecuencia de datos de entrada, que den problemas en un sistema operativo virtual, etc.

4 DOCUMENTACIÓN PARA LAS PRUEBAS

El estándar IEEE 829-1998 describe el conjunto de documentos que pueden producirse durante el proceso de prueba. Son los siguientes:

- **Plan de pruebas**. Describe el alcance, el enfoque, los recursos y el calendario de las actividades de prueba. Identifica los elementos a probar, las características que se van a probar, las tareas que se van a realizar, el personal responsable de cada tarea y los riesgos asociados al plan.

- **Especificaciones de pruebas**. Están cubiertas por tres tipos de documentos: la especificación del diseño de la prueba, la especificación de los casos de prueba y la especificación de los procedimientos de prueba.

- **Informes de pruebas**. Se definen cuatro tipos de documentos: un informe que identifica los elementos que están siendo probados, un registro de las pruebas, un informe de incidentes de prueba y un informe resumen de las actividades de prueba.