ENTORNOS DE DESARROLLO

UT3: Diseño y realización de pruebas I

CFGS Desarrollo de Aplicaciones Multiplataforma
Curso 2024-25

Índice de contenidos

1. Introducción

Como programadores, sabemos que escribir código funcional es solo el primer paso; necesitamos verificar que ese código realmente funcione como esperamos y que lo siga haciendo en el tiempo, incluso al hacer cambios. Se trata de garantizar su **calidad**, **fiabilidad** y **funcionalidad** para cumplir con los objetivos del sistema y las expectativas de los usuarios. Las pruebas en el software son importantes para:

- **Prevenir de errores en producción**: Uno de los mayores costos en desarrollo de software es corregir errores cuando ya se encuentran en producción. Detectar estos problemas en una fase temprana puede ahorrar tiempo, dinero y mejorar la satisfacción del usuario final.
- **Mejorar la calidad del software**: Las pruebas aseguran que el sistema cumpla con los requisitos funcionales (qué debe hacer) y no funcionales (rendimiento, seguridad, usabilidad). Un software de alta calidad cumple con ambas expectativas, minimizando errores y siendo más intuitivo y estable.
- **Facilitar el mantenimiento y la evolución**: En proyectos a largo plazo, el software suele evolucionar con cambios y adiciones de funcionalidades. Las pruebas automatizadas facilitan la modificación del código, proporcionando una "red de seguridad" que alerta si algo que antes funcionaba se rompe con los nuevos cambios.
- **Aumentar la confianza en el sistema**: Cuando un equipo de desarrollo sabe que un sistema ha pasado por una batería de pruebas, existe una mayor confianza en que la aplicación funcionará como se espera. Esto es importante no solo para los desarrolladores, sino también para los clientes y usuarios finales.

Si bien la teoría de pruebas suena ideal, en la práctica enfrentamos varias limitaciones: **tiempo**, **costos** y **complejidad**. No siempre es viable hacer pruebas exhaustivas en todos los aspectos, pero sí es posible establecer un proceso de pruebas eficiente y efectivo que nos permita entregar software de calidad y confianza.

1.1 Procedimientos y casos de prueba

Un procedimiento de prueba incluye los elementos clave para asegurar que la prueba se ejecute de manera correcta y reproducible. Los **casos de prueba** son **descripciones detalladas de escenarios específicos** que se utilizan para verificar que una funcionalidad del software funciona como se espera. Los casos de prueba sirven como guías prácticas y detalladas para los *testers* al ejecutar las pruebas. Algunos de sus componentes más comunes son:

1. **ID del caso de prueba**: Un identificador único para cada caso de prueba, que permite referirse a él en documentación y reportes.
2. **Título o descripción breve**: Un nombre descriptivo o breve resumen que indique de qué trata el caso de prueba, como "Verificar inicio de sesión con credenciales válidas".
3. **Objetivo de la prueba**: Explica el propósito específico de la prueba, es decir, qué se espera verificar o validar en el sistema. Puede estar relacionado con la funcionalidad, rendimiento, seguridad, usabilidad, etc.
4. **Requisitos previos o precondiciones**: Son las condiciones o configuraciones que deben cumplirse antes de iniciar la prueba, como el estado del sistema, datos específicos en la base de datos, configuraciones de red, usuarios con permisos particulares, etc.
5. **Entradas y datos de prueba**: Especifica los datos necesarios para la prueba, como valores de entrada, parámetros, o archivos de configuración. Estos datos deben estar claramente definidos para que la prueba pueda ejecutarse en las mismas condiciones cada vez.
6. **Pasos detallados de ejecución**: Se proporciona una lista clara y detallada de los pasos que el *tester* debe seguir para realizar la prueba. Estos pasos deben ser específicos para evitar ambigüedades. Cada paso puede incluir acciones a realizar en la interfaz, comandos a ejecutar, o interacciones con el sistema.
7. **Resultados esperados**: Para cada paso o al final del procedimiento, se indica el resultado que debería observarse si el sistema funciona correctamente. Esto permite al tester verificar si la prueba ha sido exitosa o si existe una discrepancia (es decir, un defecto).
8. **Criterios de éxito o fracaso**: Se establecen criterios que indican cuándo una prueba se considera pasada o fallida. Esto es importante para documentar el resultado final de cada prueba y evaluar si el sistema cumple con los requisitos.
9. **Postcondiciones o limpieza**: Define cualquier acción necesaria después de la ejecución de la prueba, como restablecer el sistema a su estado inicial o limpiar datos temporales. Esto garantiza que el sistema esté listo para futuras pruebas.

1.2 Documentación de las incidencias

La **documentación de las pruebas de software** es un proceso fundamental para registrar qué se probó, cómo se probó, y los resultados obtenidos. Esta documentación es clave para mantener un historial claro y detallado de las pruebas realizadas, facilitando la trazabilidad, el análisis de resultados, la repetición de pruebas, y la comunicación dentro del equipo de desarrollo y con otras partes interesadas. Los documentos más importantes son:

1. **Plan de pruebas**: Describe el alcance, enfoque, recursos y cronograma de las actividades de prueba. Contenido:
    - Alcance de las pruebas: qué funcionalidades o módulos serán probados.
    - Estrategia de pruebas: tipos de pruebas (unitarias, de integración, etc.).
    - Criterios de inicio y finalización: condiciones para empezar y terminar las pruebas.
    - Roles y responsabilidades: equipo encargado de cada aspecto de las pruebas.
    - Recursos: herramientas, entornos, y datos de prueba necesarios.
    - Cronograma de pruebas: fechas de inicio y fin para cada fase de prueba.

2. **Informe final de pruebas**: Presenta un resumen de todas las pruebas realizadas y sus resultados. Contenido:
    - Resumen de resultados: porcentaje de pruebas pasadas, fallidas y bloqueadas.
    - Calidad del sistema: evaluación general basada en los resultados de las pruebas.
    - Defectos encontrados y resueltos: lista de errores con sus respectivos estados.
    - Recomendaciones: sugerencias para futuras pruebas o para la liberación del software.
    - Cierre de pruebas: confirmación de que se han cumplido los criterios de finalización establecidos en el plan de pruebas.

2. Planificación de pruebas

Es importante planificar qué pruebas se van a realizar durante el desarrollo de un proyecto software. Normalmente si el proyecto es de envergadura es probable que se realicen las pruebas que se describen a continuación. Si el proyecto es pequeño, serán los responsables técnicos del proyecto los que determinen cómo proceder.

2.1 Pruebas unitarias

Las **pruebas unitarias** son un tipo de prueba de software que verifica el funcionamiento de componentes individuales o "unidades" del código, como funciones o métodos, de forma aislada. Su objetivo es asegurar que cada unidad funcione correctamente según lo esperado, detectando errores en una etapa temprana del desarrollo.

Estas pruebas se realizan generalmente **durante la fase de desarrollo**, a medida que se escriben las unidades de código, permitiendo a los desarrolladores identificar y corregir errores antes de integrar las unidades en el sistema completo.

2.2 Pruebas de integración

Las **pruebas de integración** son un tipo de prueba de software que verifica la interacción y el funcionamiento conjunto de múltiples componentes o módulos del sistema. Su objetivo es detectar problemas que pueden surgir cuando las unidades individuales se combinan, asegurando que se integren correctamente y cumplan con los requisitos.

Estas pruebas se realizan **después de las pruebas unitarias**, generalmente en la fase de desarrollo y antes de las pruebas de validación, una vez que los módulos han sido desarrollados y están listos para ser integrados y probados en conjunto.

2.3 Pruebas de aceptación o validación

Las **pruebas de aceptación o validación** son un tipo de prueba de software que evalúa el sistema completo como un todo para verificar su conformidad con los requisitos especificados. Este tipo de pruebas se realiza en un entorno que simula el entorno de producción y se centra en validar el comportamiento del sistema, incluyendo su funcionalidad, rendimiento, seguridad y usabilidad.

El objetivo principal de las pruebas del sistema es asegurar que todas las partes del software funcionan correctamente en conjunto y que el sistema cumple con las expectativas del usuario final. Se llevan a cabo generalmente después de las pruebas de integración y antes de la liberación del software.

En este estadio, existen las pruebas **alfa**, que se realizan en un entorno controlado y bajo unas especificaciones concretas, y pruebas **beta**, en las que los usuarios prueban el sistema en un entorno con controlado por los desarrolladores.

3. Tipos de pruebas

Existen muchos tipos de pruebas. Veamos las categorías más importantes:

3.1 Funcionales

Las pruebas funcionales se centran en verificar que el software cumple con los requisitos especificados y se comporta como se espera. Estas pruebas se centran en las funciones del sistema y en cómo se comporta ante diversas entradas. Suelen considerarse como pruebas de **caja negra** porque no se centran en la estructura interna del código, sino en lo que el sistema debe hacer.

El objetivo de estas pruebas es asegurar que las funcionalidades del software, como la entrada y salida de datos, se implementen correctamente y que el sistema responde adecuadamente a diferentes condiciones.

3.2 Estructurales

Las pruebas estructurales, también conocidas como pruebas de **caja blanca**, se centran en la estructura interna del software. Esto incluye la lógica, el flujo de control y las interacciones entre componentes.

El objetivo de estas pruebas es evaluar la funcionalidad interna y la calidad del código, asegurando que todas las rutas de ejecución se hayan probado y que no existan errores ocultos en la lógica del sistema. Generalmente, para este tipo de pruebas se utilizan herramientas especializadas.

3.3 Regresión

Las pruebas de regresión son un tipo de prueba que se realiza para verificar que cambios recientes en el código (como correcciones de errores o nuevas funcionalidades) no hayan afectado negativamente a las funcionalidades existentes del software.

El objetivo de estas pruebas es asegurar que el software sigue funcionando correctamente después de realizar modificaciones y que no se han introducido nuevos errores en partes del código que anteriormente funcionaban.

Estas pruebas pueden ser funcionales o estructurales, dependiendo de qué aspectos del software se estén probando. Se ejecutan con frecuencia, especialmente en proyectos ágiles, después de cada iteración o ciclo de desarrollo, así que, a menudo se automatizan para permitir pruebas rápidas y repetibles tras cada cambio.