Pruebas de software

En los proyectos de desarrollo de software, es común que surjan situaciones que comprometan el éxito o la continuación de los proyectos debido a diferentes factores. Usualmente los problemas surgen debido a demoras, sobrecostos no previstos y mala calidad del producto final. Los ingenieros de software tienen directa responsabilidad de anticiparse a estas situaciones a fin de reducir la probabilidad de ocurrencia, y minimizar el impacto en sus proyectos.

Una buena práctica para poder enfrentar y evitar malos resultados en sus proyectos son las pruebas de software, estas son seguramente la actividad más común de control de calidad realizada en los proyectos de desarrollo o mantenimiento de sistemas. Aunque un aseguramiento de calidad de software más eficaz debería incluir otras técnicas como, por ejemplo, inspecciones y revisiones (automatizadas o no) de modelos y documentos no ejecutables de las primeras fases de desarrollo, no existe proyecto de desarrollo que no realice de manera más o menos exhaustiva y formal pruebas de software.

"La forma en que puede estar seguro acerca de la funcionalidad, el rendimiento y la experiencia del usuario. Ya sea que realice sus pruebas manualmente o a través de la automatización, cuanto antes y más a menudo pueda llevar a cabo pruebas, más probable es que identifique errores, no sólo ahorrándole a usted y a su equipo de posibles simulacros de incendio más adelante, sino también asegurándose de que su aplicación de software haya sido revisada y auditada a fondo antes de que esté frente a sus usuarios. Si los problemas se arrastran al entorno de producción, los más caros y lentos que van a solucionar."

Existen variados tipos de pruebas de software, cada una con objetivos y estrategias específicos:

- **Pruebas de aceptación:** Verificar si todo el sistema funciona según lo previsto.
- **Pruebas de integración:** Se asegura de que los componentes o funciones del software operen juntos.
- ***Pruebas unitarias:*** Valida que cada unidad de software funcione como se esperaba. Una unidad es el componente comprobable más pequeño de una aplicación.
- **Pruebas funcionales:** Verifica las funciones emulando escenarios comerciales, basados en requisitos funcionales. Las pruebas de caja negra son una forma común de verificar funciones.
- **Pruebas de rendimiento:** Prueba el rendimiento del software en diferentes cargas de trabajo. Las pruebas de carga, por ejemplo, se utilizan para evaluar el rendimiento en condiciones de carga reales.
- **Pruebas de regresión:** Verifica si las nuevas funciones rompen o degradan la funcionalidad. Las pruebas de cordura se pueden utilizar para verificar menús, funciones y comandos a nivel de superficie, cuando no hay tiempo para una prueba de regresión completa.
- **Pruebas de estrés:** Prueba cuánta tensión puede soportar el sistema antes de que falle. Está considerada como un tipo de prueba no funcional.
- **Pruebas de usabilidad:** Valida qué tan bien un cliente puede usar un sistema o una aplicación web para completar una tarea."

Pruebas unitarias

Las pruebas unitarias de software son las que se realizan sobre cada método o función de software una a una. El principal objetivo es comprobar que el método, entendido como una unidad funcional de un programa independiente, cumpla con su tarea y que de paso esté correctamente codificado. Los métodos deben cumplir con ciertas características mínimas:

- Debe ser un bloque básico de construcción de programas.
- Debe implementar una función independiente simple.
- Podrá ser probado al cien por cien por separado.
- No deberá tener más de 500 líneas de código.

Las pruebas unitarias deben estar optimizadas para ser rápidas y eficaces, su tiempo de ejecución no puede ser muy grande, esta es una de sus principales características. Es de suma importancia que se pruebe la totalidad del código con este tipo de pruebas, tanto así que existen algunos tipos de test unitarios que no dan por aprobado un archivo si no fueron abarcadas todas las funcionalidades por el test.

Si no se cumple esto, habrá una parte del software que estará sin probar y la probabilidad de resultados inesperados en la ejecución será exponencialmente más alta que si probamos todo o casi todo el código. Las pruebas unitarias tienen que ejecutarse aisladamente unas de otras y la ejecución de un test no debe influir en otro. Estas características que se han descrito están recogidas en el principio FIRST (Fast, Isolation, Repeteable, Small y Transparent). Si no se cumplen estas propiedades existe el riesgo de que un test unitario no sea válido.

Durante las pruebas unitarias, se ejecuta la unidad aplicándole un conjunto de casos de prueba y se estudia su comportamiento durante la ejecución. En caso de que el comportamiento observado no sea el esperado para algún caso de prueba, se dice que el caso de prueba reveló un defecto.

Generalmente los defectos encontrados provocan que el desarrollador depure la aplicación, encuentre la falla subyacente y la corrija, de modo que la próxima ejecución de los casos de prueba asociados resulte exitosa.

Las pruebas unitarias se suelen realizar por medio del patrón AAA recomendada por la comunidad, consiste en una forma de escribir cada prueba en 3 partes:

1. **Arrange:** Inicializar, consiste en establecer los valores de los datos que vamos a utilizar en las pruebas.
2. **Act:** Actuar, realiza la llamada al método a probar con los parámetros preparados para tal fin.
3. **Assert:** Comprobar, revisar que el método ejecutado se comporte tal y cómo lo tenemos previsto.