Testing de software “testing por conocimiento”

Introducción

En el desarrollo de software, el testing es fundamental para garantizar que los programas funcionen correctamente y cumplan con las expectativas y requerimientos del usuario. “El testing es una técnica de verificación y validación que se utiliza para evaluar la calidad del software, permitiendo descubrir posibles errores o fallos antes de que el producto llegue a manos de los usuarios finales.” En este informe, vamos a hablar sobre el testing y nos enfocaremos en dos puntos de vista sumamente importantes: el testing de caja negra y el testing de caja blanca.

Pero antes que nada debemos conocer ¿Qué es el testing de software? Como ya mencionamos anteriormente el testing de software es un proceso de evaluación que se lleva a cabo durante el ciclo de desarrollo de un programa. Su objetivo principal es asegurarse de que el software funcione correctamente y cumpla con los requisitos y especificaciones definidos previamente en la etapa del análisis. A través de muchas pruebas, los testers buscan identificar defectos, errores y comportamientos inesperados en el software, para poder corregirlos antes de que el software llegue a los usuarios.

“El testing es una disciplina esencial en el desarrollo de productos digitales, ya que ayuda a mitigar riesgos, mejorar la calidad y la confiabilidad del software, y garantizar una experiencia satisfactoria para los usuarios.”

Ahora que ya conocemos que es el testing, para que sirve y en que nos beneficia vamos a hablar de dos enfoques muy importantes, primero es necesario aclarar que el testing por conocimiento engloba a estos dos enfoques, consiste en el nivel de conocimiento que se tiene sobre el software. Ahora bien, ¿Qué son estos dos enfoques?

Caja Negra

Una de las técnicas más utilizadas en el testing de software es el enfoque de caja negra. Pero ¿En qué consiste?

En este método, el tester evalúa el programa sin conocer su estructura interna o el código. El tester sólo se enfoca en las entradas y salidas del sistema.

En el testing de caja negra, las pruebas se deben hacer desde la perspectiva del usuario final, validando que el software funcione correctamente y cumpla con las funcionalidades que se espera que cumpla. El objetivo es asegurarse de que el software responda como se espera en todos los casos de uso y escenarios de posibles, sin importar el cómo funcione, solo se espera ver que funcione.

¿Quién participa en las pruebas de caja negra?

– Probador

Un probador es responsable de completar los casos de prueba manuales en una empresa, escribiendo casos de prueba exhaustivos que examinan la aplicación en detalle antes de ejecutarlos e informar de los resultados. Esta función existe principalmente en un proceso de pruebas manual, y los sistemas automatizados asumen el papel cuando existe automatización de las pruebas.

– Analista de control de calidad

Responsable de programar los casos de prueba en un proceso de control de calidad, principalmente cuando la empresa utiliza un proceso de automatización de pruebas de control de calidad. El proceso implica tanto el diseño de casos de prueba exhaustivos que garanticen un alto nivel de funcionalidad como la ejecución de los casos de prueba, recuperando los resultados una vez completados.

– Promotor

La persona responsable de desarrollar el software que prueba el equipo de control de calidad. Un desarrollador recibe los comentarios del equipo de pruebas y actualiza el software en consecuencia, trabajando como parte de un equipo de desarrollo pero estando en constante comunicación con los probadores.

– Responsable de control de calidad

Se encarga de gestionar todas las tareas que realizan los probadores. Esto incluye organizar el calendario de pruebas, organizar una lista de cosas por hacer para los miembros del personal y resolver cualquier conflicto en el equipo. También explican las pruebas de caja negra en la formación para nuevos empleados.

– Jefe de proyecto

Responsable de la calidad del proyecto final, supervisa tanto el proceso de pruebas como el de desarrollo, asegurándose de que el cliente recibe un paquete de software que cumple todas las especificaciones.

Ventajas de las pruebas de caja negra

1. Sin necesidad de conocimientos técnicos

El objetivo de las pruebas de caja negra es examinar cómo funciona la aplicación para un usuario final, y el usuario estándar no tiene conocimientos técnicos avanzados en la mayoría de las situaciones. Esto puede reducir el coste de las pruebas, ayudando a la organización a descubrir más fallos con un gasto menor, lo que resulta más eficiente desde el punto de vista financiero.

2. Modelar con precisión al usuario

Algunos tipos de pruebas de caja negra -que se centran en reproducir la forma en que se comporta un usuario- modelan el comportamiento de un usuario con un alto grado de precisión. Este es especialmente el caso de las pruebas de aceptación del usuario, en las que los usuarios finales experimentan el producto, no sólo modelando o simulando el comportamiento de un usuario, sino poniéndolo realmente en práctica.

3. Posibilidad de realizar pruebas mediante crowdsourcing

Las pruebas de caja negra son una forma muy accesible de realizar pruebas gracias a los requisitos de conocimientos relativamente bajos. Esto significa que las empresas no sólo pueden contratar a probadores con un menor nivel de conocimientos técnicos, sino que pueden subcontratar sus pruebas a clientes ávidos. Esto es cada vez más habitual en la industria del videojuego, con empresas que ofrecen lanzamientos en Acceso Anticipado, actualizando el juego con el tiempo para resolver los problemas que encuentran los usuarios. Encontrar errores en este caso es mucho más fácil, ya que todas las características reciben un nivel de exposición mucho mayor.

Retos de las pruebas de caja negra

1. Dificultad para encontrar las causas del problema

Uno de los principales inconvenientes de las pruebas de caja negra es que puede resultar más difícil encontrar la causa de los problemas cuando los probadores no tienen acceso al código fuente. Aunque pueden describir en qué consiste el error y cuándo se produce, no tienen ninguna indicación de qué parte del código fuente causa los problemas ni por qué..

2. La automatización es más complicada

Dado que lo que se pretende es reproducir la forma en que un usuario interactúa con un paquete de software, puede resultar extremadamente difícil automatizar un proceso de pruebas de caja negra. La primera causa es que el probador no tiene acceso al código fuente, lo que dificulta la codificación de un caso de prueba preciso. Esto se une al hecho de que las pruebas están diseñadas para replicar el comportamiento humano en la medida de lo posible, con una automatización específicamente diseñada para actuar de forma robótica. Puede equilibrar este problema automatizando las tareas más insignificantes y combinando la automatización con pruebas manuales siempre que sea posible.

3. Luchas con las pruebas a gran escala

Los problemas de automatización antes mencionados hacen que las pruebas a mayor escala sean más complicadas. Las pruebas a gran escala proporcionan a las empresas muchos más datos sobre el software y facilitan la detección y reproducción de errores. La exigencia de pruebas manuales como prioridad significa que puede ser más difícil organizar pruebas a mayor escala. Algunas empresas contrarrestan esta situación con un sistema de «beta abierta», en el que cualquier persona interesada en el producto puede colaborar en las pruebas previas al lanzamiento y ayudar a la empresa aportando voluntariamente sus comentarios sobre las primeras versiones.

Características de las pruebas de caja negra

1. Sin conocimiento interno previo

Las pruebas de caja negra no requieren ningún conocimiento interno previo del software. Esto puede ser difícil en algunos casos, ya que los probadores tienen una idea de los aspectos del software que están probando y algunas de las características que están buscando, pero esto se define en términos generales como no poder ver documentación interna de ningún tipo.

2. Separar a probadores y desarrolladores

Las fases de prueba y desarrollo las realizan personas diferentes en una situación de prueba de caja negra. Esta diferenciación proviene de la falta de conocimiento que tienen los probadores, ya que los desarrolladores tienen conocimiento del código fuente debido a que fueron ellos los responsables de desarrollarlo. Algunas optan por recurrir a una organización externa para llevar a cabo las pruebas, mientras que las empresas más grandes cuentan con departamentos especializados de probadores para realizar este trabajo.

3. Pruebas finales

Se refiere a la fase de desarrollo en la que se producen estas pruebas. Las pruebas de caja negra se basan en una versión relativamente avanzada de una aplicación existente, con una interfaz de usuario completa que permita una navegación total por el software y el acceso a la parte frontal de cada función. Esto significa que las pruebas de caja negra sólo son posibles en algunas de las últimas fases del proceso de pruebas, cuando todo esto se ha desarrollado inicialmente. Aunque la interfaz de usuario y los controles pueden modificarse con el tiempo, deben existir de alguna forma para permitir que las pruebas de caja negra accedan a la funcionalidad.

Qué probamos en las pruebas de caja negra

1. Funcionalidad
2. Interfaz de usuario
3. Rendimiento

Técnicas de pruebas de caja negra

1. Pruebas por pares

La prueba por pares es una forma de prueba que se centra en probar todas las combinaciones posibles de entradas de datos en el software.

Por ejemplo, si los números del uno al diez son todas entradas válidas en una columna con todos los caracteres del alfabeto en otra, las pruebas por pares probarían todas las combinaciones posibles de 1A a 10Z. Se trata de una forma de prueba que puede llevar mucho tiempo y esfuerzo al usuario, lo que la convierte en una de las técnicas más abiertas a una posible hiperautomatización. Es extremadamente minucioso y detecta cualquier posible problema con la introducción de datos.

2. Análisis del valor límite

Muchos programas informáticos se basan en la introducción de datos, con unos límites específicos dentro de los cuales se espera que trabaje el cliente.

Por ejemplo, un sistema diseñado para calcular cifras de 1 a 100 podría tener problemas con valores en 0 o inferiores, o superiores a 100.

3. Pruebas de transición de estado

Muchos programas varían entre diferentes «estados» o «modos» y requieren una transición de una etapa de este proceso a la siguiente. Que estas transiciones funcionen correctamente significa que el sitio funciona como el usuario espera y no se producen retenciones inesperadas.

Ejemplo de Testing de Caja Negra:

Por ejemplo, cuando un usuario usa una cámara de foto. En este caso, al usuario solo le importa que al apretar el botón para sacar la foto y poder ver la imagen. No necesitas saber cómo funciona internamente la cámara o cómo se procesa la imagen, solo le interesa el resultado final.

Al apretar el botón “entrada” se obtiene la foto “salida”

Dejamos otro ejemplo en código de cómo sería el testing de caja negra en código:

```
// Test de Caja Negra
public class TestCajaNegra {
   public static void main(String[] args) {
      Sumador sumador = new Sumador();

      // Caso de prueba 1
      int resultado1 = sumador.sumar(5, 10);
      System.out.println("Resultado esperado: 15");
      System.out.println("Resultado obtenido: " + resultado1);

      // Caso de prueba 2
      int resultado2 = sumador.sumar(-3, 8);
      System.out.println("Resultado esperado: 5");
      System.out.println("Resultado obtenido: " + resultado2);

      // Caso de prueba 3
      int resultado3 = sumador.sumar(0, 0);
      System.out.println("Resultado esperado: 0");
      System.out.println("Resultado obtenido: " + resultado3);
   }
}
```

Caja Blanca

El enfoque de caja blanca, por otra parte, busca entender lógica interna y el código del software. El tester prueba directamente el código fuente y la arquitectura del programa para descubrir posibles defectos relacionados con la lógica interna y la funcionalidad del software. Las pruebas pueden realizarse en distintas fases del ciclo de pruebas para verificar el funcionamiento del código y la estructura interna. Lo más habitual es que estas, se realicen cuando los desarrolladores y los probadores llevan a cabo pruebas unitarias y, a veces, durante las pruebas de integración. Las pruebas de caja blanca casi siempre las llevan a cabo desarrolladores e ingenieros de software. Esto se debe a que las pruebas de caja blanca requieren un conocimiento detallado del código informático y de las técnicas de codificación

Ventajas

Las pruebas de caja blanca permiten a los desarrolladores e ingenieros de software probar más aspectos del código que las pruebas de caja negra.

1. Maximizar la cobertura de las pruebas

Probar la mayor parte posible del código del software suele maximizar las posibilidades de detectar cualquier fallo o error presente en el código, y el propósito de las pruebas de caja blanca suele ser probar la mayor parte posible del código.

2. Encontrar errores y fallos ocultos

Una de las mayores ventajas de las pruebas de caja blanca es que, dado que verifican la funcionalidad interna, facilitan a los desarrolladores la detección de errores y fallos que, de otro modo, podrían estar ocultos en lo más profundo del código.

3. Facilidad de automatización

Es muy fácil automatizar las pruebas de caja blanca, especialmente cuando se realizan pruebas unitarias. Las pruebas unitarias suelen requerir que los desarrolladores prueben pequeños fragmentos de código de forma individual para comprobar si se ejecutan según lo esperado. Es muy fácil de automatizar, lo que significa que es una forma rápida y eficaz de probar el software.

Esta es una de las razones por las que las pruebas unitarias se realizan antes que otros tipos de pruebas que requieren más tiempo.

4. Tiempo eficiente

Es relativamente fácil automatizar la mayoría de los tipos de pruebas de caja blanca, lo que significa que a menudo es más rápido llevar a cabo pruebas de caja blanca que pruebas de caja negra. Además, las pruebas de caja blanca facilitan a los desarrolladores la localización de los fallos y errores que identifican en el código, ya que los encuentran mientras prueban el propio código.

5. Calidad del código

Las pruebas de caja blanca permiten a los desarrolladores echar un segundo vistazo al código que han escrito y evaluar su calidad y limpieza.

Retos de las pruebas de caja blanca

Las pruebas de caja blanca no están exentas de dificultades. Hay algunas razones por las que algunos equipos de desarrollo pueden considerar que las pruebas de caja blanca son más difíciles de llevar a cabo que las pruebas de caja negra, así como otras razones por las que algunas personas pueden considerarlas menos importantes que las pruebas de caja negra.

1. Obstáculos técnicos

Las pruebas de caja blanca conllevan barreras técnicas que no tienen las pruebas de caja negra. Para realizar pruebas de caja blanca, los probadores necesitan conocer el funcionamiento interno del sistema, lo que, en las pruebas de software, suele significar conocimientos de programación. Por eso, las pruebas de caja blanca las realizan casi siempre los ingenieros y desarrolladores de software y no los evaluadores de control de calidad, que rara vez tienen los conocimientos técnicos necesarios para realizar este tipo de pruebas.

2. Coste

Los desarrolladores deben dedicar mucho tiempo a escribir pruebas unitarias intensivas, y las pruebas de caja blanca a menudo no pueden reutilizarse para otras aplicaciones, lo que significa que la realización de pruebas de caja blanca suele costar bastante.

3. Precisión

LLas pruebas de caja blanca sólo validan las características que ya existen, mientras que las de caja negra pueden utilizarse para probar características parcialmente implementadas o identificar las que realmente faltan en el software y deberían incluirse en iteraciones posteriores.

4. Alcance

Aunque los desarrolladores pueden utilizar las pruebas de caja blanca para verificar si el código funciona como debería, no pueden concluir que el código en funcionamiento ofrece los resultados correctos a los usuarios finales sin combinar las pruebas de caja blanca con las de caja negra. Esto significa que hay limitaciones en el alcance de las pruebas de caja blanca y en lo que pueden decirnos sobre el software.

Características de las pruebas de caja blanca

1. Mantenibilidad

Las pruebas de caja blanca conducen a un mayor nivel de mantenimiento del código, lo que simplifica el trabajo que el equipo debe realizar en el futuro.

2. Flexibilidad

Las pruebas se realizan sobre código lo suficientemente flexible como para aceptar cambios con relativa rapidez.

3. Modularidad

Las pruebas de caja blanca prosperan en código con cierto grado de modularidad, lo que significa que los distintos elementos del software se distinguen claramente unos de otros.

4. Integración

Los encargados de las pruebas pueden ver si una función funciona hasta el punto en que sale del software en cuestión y si vuelve del sistema integrado tan funcional como se esperaba.

¿Qué probamos en las pruebas de caja blanca?

1. Agujeros de seguridad internos
2. Vías en los procesos de codificación
3. Resultados esperados
4. Declaraciones, objetos y funciones
5. Funcionalidad de los bucles condicionales

Tipos de pruebas de caja blanca

1. Pruebas de trayectoria

Basada en la estructura de control de un programa. Los desarrolladores utilizan la estructura de control para crear un gráfico de flujo de control y probar diferentes rutas en el gráfico.

2. Pruebas en bucle

Son uno de los tipos más importantes de pruebas de caja blanca que comprueban los bucles dentro del código del programa. Los bucles se implementan en algoritmos dentro del código y la comprobación de bucles verifica si estos bucles son válidos. Un ejemplo de prueba de bucle es el seguimiento a través del bucle con un conjunto específico de puntos de datos que incitan al bucle a continuar, como la negativa a aceptar algunos términos y condiciones, antes de introducir una cifra que rompa específicamente el bucle. Si el bucle se comporta como se espera, la prueba se realiza correctamente.

3. Pruebas condicionales

Comprueba si las condiciones lógicas para los valores dentro del código son verdaderas o falsas. indican a los desarrolladores si el código es lógico y cumple los requisitos de la lógica de programación.

4. Pruebas unitarias

Los desarrolladores prueban componentes y módulos individuales y comprueban que funcionan como se espera antes de integrar las distintas unidades.

5. Pruebas de mutación

Las pruebas de mutaciones son un tipo de pruebas que analizan alteraciones y mutaciones. Los desarrolladores introducen pequeñas modificaciones en el código fuente para ver si esto puede revelar fallos en el código.

Si el caso de prueba pasa, esto indica que hay algún problema con el código porque no debería pasar después de haber realizado los cambios. Idealmente, en las pruebas de mutación, todos los casos de prueba fallarán.

6. Pruebas de integración

Comprueban si los distintos módulos funcionan correctamente cuando se integran con otros.

7. Pruebas de penetración

Pueden utilizarse para simular ciberataques específicos en el sistema. En las pruebas de penetración, los probadores tienen acceso a datos completos de la red y del sistema, como contraseñas y mapas de red. A continuación, intentan acceder a los datos del sistema o destruirlos intentando atacar por tantas vías como sea posible.

Técnicas de pruebas de caja blanca

1. Cobertura de la declaración

Los probadores deben intentar abarcar la mayor parte posible del código fuente cuando realicen pruebas. La cobertura del código es una buena medida de ello, y la cobertura de sentencias es una técnica que los evaluadores de caja blanca pueden utilizar para aumentar la cobertura de las sentencias dentro del código. La cobertura de sentencias es una métrica que mide el número de sentencias ejecutadas dividido por el número total de sentencias y multiplicado por 100. Los probadores de caja blanca deben aspirar a una cobertura de declaración alta.

2. Cobertura de ramas

La cobertura de ramas, al igual que la cobertura de sentencias, refleja la amplitud de la cobertura de determinados elementos del código en las pruebas de caja blanca. Las bifurcaciones equivalen a las sentencias «SI» de la lógica, en las que el código se bifurca en opciones verdaderas y falsas que influyen en el resultado de la operación.

Cuando se utilizan técnicas de cobertura de ramas, los probadores de caja blanca comprueban si cada rama se procesa al menos una vez y validan que ambas ramas funcionan correctamente.

3. Cobertura de la ruta

Evalúan las rutas dentro de una aplicación de software. Maximizar la cobertura de la ruta de prueba significa garantizar que todas las rutas del programa se exploran al menos una vez. Es un tipo de técnica de prueba similar a la cobertura de ramas, pero se considera más exhaustiva y eficaz.

4. Cobertura de decisiones

Proporciona datos sobre los resultados verdaderos y falsos de las expresiones booleanas en el código fuente.

Las pruebas de cobertura de decisiones validan el código fuente garantizando que cada marca de cada decisión potencial se recorre al menos una vez durante las pruebas.

5. Cobertura de condiciones

La cobertura de condiciones también se conoce como cobertura de expresión. Esta técnica de caja blanca evalúa las subvariables de las

sentencias condicionales dentro del código para verificar el resultado de cada condición lógica.
Este tipo de pruebas sólo tiene en cuenta las expresiones con operandos lógicos, mientras que las pruebas de cobertura de decisiones y las pruebas de cobertura de ramas se utilizan para garantizar otras operaciones lógicas.

6. Cobertura de afecciones múltiples

En las pruebas de cobertura de condiciones múltiples, los probadores verifican diferentes combinaciones de condiciones y evalúan la decisión que toma el código para cada combinación.
Puede haber muchos casos de prueba diferentes para las pruebas de cobertura de condiciones múltiples debido al enorme número de combinaciones de condiciones que existen, por lo que este tipo de pruebas suele llevar mucho tiempo.

7. Cobertura de máquinas de estados finitos

La cobertura de máquinas de estados finitos es un tipo de prueba importante, pero también una de las formas más difíciles de lograr una alta cobertura de código en las pruebas de caja blanca. Trabaja sobre la funcionalidad del diseño y requiere que los desarrolladores cuenten el número de veces que se visita o transita por un estado durante el proceso de prueba, así como cuántas secuencias contiene cada sistema de estados finitos.

8. Pruebas de flujo de control

Trata de establecer el orden de ejecución del programa utilizando una estructura de control sencilla.
Los desarrolladores construyen casos de prueba de flujo de control eligiendo una sección específica del programa y construyendo una ruta de prueba. Las pruebas de flujo de control suelen utilizarse en las pruebas unitarias.

Ejemplo de Testing de Caja Blanca

En este tipo de testing se pueden diseñar casos de prueba específicos para probar cada escenario y seguir cada camino posible. Continuando con el ejemplo de la cámara en el testing de caja blanca podríamos ver cómo se procesa la imagen y todo el de más proceso requerido para sacar una foto y obtenerla como salida.

Dejamos un código de ejemplo en el cual podemos ver el proceso que se realiza para hacer en este caso una suma y obtener la salida en base a la entrada de dos números.

```
// Test de Caja Blanca
public class TestCajaBlanca {
   public static void main(String[] args) {
      Sumador sumador = new Sumador();

      // Caso de prueba 1: Números positivos
      int resultado1 = sumador.sumar(5, 10);
      System.out.println("Resultado esperado: 15");
      System.out.println("Resultado obtenido: " + resultado1);

      // Caso de prueba 2: Número positivo y negativo
      int resultado2 = sumador.sumar(-3, 8);
      System.out.println("Resultado esperado: 5");
      System.out.println("Resultado obtenido: " + resultado2);

      // Caso de prueba 3: Números negativos
      int resultado3 = sumador.sumar(-5, -10);
      System.out.println("Resultado esperado: -15");
      System.out.println("Resultado obtenido: " + resultado3);
   }
}
```

En este ejemplo, el testing de caja negra se enfoca únicamente en probar las entradas y salidas del método `sumar`, sin necesidad de conocer cómo funciona internamente. Mientras que el testing de caja blanca nos permite diseñar pruebas específicas para probar diferentes escenarios

dentro del código, como sumas con números positivos, negativos o combinaciones. Ambos enfoques son importantes y se complementan para garantizar la calidad y que el método cumpla.

Conclusión

El testing de software es una técnica fundamental para garantizar la calidad y el funcionamiento de los programas. A través el enfoque de caja negra, podemos validar que el software cumpla con lo esperado desde la perspectiva del usuario, mientras que el enfoque de la caja blanca analiza el código interno para asegurar que funcione y que sea de calidad.

Fuentes:

https://es.wikipedia.org/wiki/Caja_blanca_(sistemas)
https://openwebinars.net/blog/que-es-black-box-testing-o-pruebas-de-caja-negra/
https://www.testermoderno.com/caja-blanca-vs-caja-negra/
https://www.zaptest.com/es/pruebas-de-caja-blanca-que-es-como-funciona-retos-metricas-herramientas-y-mas
https://www.zaptest.com/es/pruebas-de-caja-negra-que-son-tipos-procesos-enfoques-herramientas-y-mucho-mas
https://www.zaptest.com/es/pruebas-de-caja-gris-profundice-en-que-son-tipos-procesos-enfoques-herramientas-y-mucho-mas