

**INSTITUTO DE ESTUDIOS SUPERIORES**

**TRABAJO PRÁCTICO**

**Carrera: Desarrollo de Software.**

**Año: 2do Año.**

**Materia: Unidad de Definición Institucional ll.**

**Profesor: Larocca Rubén.**

**Alumno: Kerps Maximiliano Ezequiel.**

**PRUEBAS ESTRUCTURALES**

Este trabajo práctico está enfocado en un tipo de prueba, es decir, ensayos estructurales. Como el nombre lo puede sugerir, son aquellas destinadas a la estructura del código.

**Pruebas estructurales frente a otros tipos de pruebas**

La ISTQB define cuatro principales tipos de pruebas de software.

Funcionales

Implica verificar la funcionalidad del software con los requisitos estipulados. Los datos de prueba se utilizan como entrada. También comprobamos que el resultado dado sea el esperado.

No funcionales

Implica un proceso de prueba para analizar qué tan bien funciona el software.

Basadas en el cambios

Implica probar los efectos de realizar cambios en el código y asegurarse de que estos se hayan implementado. También tienen la intención de asegurar que los cambios en el código, no generen efectos negativos.

Estructurales

Como se mencionó anteriormente, estas se basan en la estructura del código. Un ejemplo sería, si un código está destinado a calcular el promedio de números pares en una matriz, entonces las pruebas basadas en la estructura estarían interesadas en los pasos que conducen al cálculo del promedio y no si el resultado final es un valor numérico correcto.

Supongamos que tenemos que comprobar si hemos definido el código que diferencia los números pares de los impares. Podemos tener una instrucción condicional, decir, si un elemento de la matriz es divisible por dos sin un resto, entonces podemos categorizar ese valor como número par.

**¿Qué no son las pruebas estructurales?**

Hay que tener la consideración de que en estas pruebas se trata con el código. No se verifican los requisitos ni se prueban las entradas con los resultados esperados. No son pruebas a las que les preocupe la funcionalidad, la experiencia del usuario o incluso el rendimiento en este momento.

**¿Qué son las pruebas estructurales?**

Se pueden definir como un tipo de prueba de software que prueba la estructura del código y los flujos previstos. Por ejemplo, verificar el código real para aspectos como la implementación correcta de declaraciones condicionales, y si cada declaración en el código se ejecuta correctamente.

Para llevar a cabo este tipo de pruebas, es necesario comprender el código. Esta es la razón por la que estas pruebas las realizan generalmente los desarrolladores que escribieron el código.

**Ventajas**

**Permite pruebas de código exhaustivas, lo que resulta en errores mínimos.**

Las pruebas basadas en estructuras dan espacio para que el software se pruebe a fondo. Los diferentes niveles de cobertura, declaración por declaración, cada punto de decisión y ruta, apuntan a lograr una cobertura del 100%, lo que reduce en gran medida las posibilidades de que los errores pasen desapercibidos.

**La capacidad de automatizar**.

Hay varias herramientas que podemos utilizar para automatizar las pruebas. Esto nos ayudará a lograr la máxima cobertura de código y en un tiempo más corto en comparación con hacer las pruebas manualmente.

**Da como resultado un código de mayor calidad**.

Los desarrolladores tienen la oportunidad de estudiar la estructura y la implementación del código y corregir cualquier error, así como mejorar estos aspectos. Nos permite tener en cuenta la gran estructura a medida que escribimos las siguientes partes del código o implementamos las funciones restantes.

**Se puede hacer a través de cada fase del SDLC.**

Las pruebas estructurales se pueden realizar en cada fase del SDLC sin esperar a que el desarrollo se complete al 100%. Esto facilita la identificación de errores en la fase inicial y, por lo tanto, ahorra mucho tiempo en comparación con las pruebas una vez finalizado el desarrollo.

**Ayuda a deshacerse del código muerto**.

Esto puede verse como un código “adicional” o innecesario, **por ejemplo,** código que calculará un resultado, pero nunca lo usará en ningún cálculos.

**Eficiencia.**

Dado que los desarrolladores que escriben el código son los mismos que lo prueban, no es necesario involucrar a otras personas como QA.

**Desventajas**

Los desarrolladores que realizan pruebas basadas en estructuras deben tener un conocimiento profundo del lenguaje. Otros desarrolladores y QA que no estén bien versados ​​en el idioma no pueden ayudar con las pruebas.

Puede resultar bastante caro en términos de tiempo y dinero. Se requiere mucho tiempo y recursos para realizar pruebas de manera eficiente.

Provoca retrasos en la entrega de funciones. Esto se debe a que los desarrolladores se ven obligados a dejar de crear software para realizar pruebas.

La escala es un problema, especialmente cuando se trata de grandes aplicaciones. Una aplicación grande equivale a un número excesivamente alto de rutas por cubrir. Alcanzar una cobertura del 100% se vuelve imposible.

Puede haber casos y rutas perdidas , por ejemplo, en un caso en el que las características no estén completamente desarrolladas o aún no se hayan desarrollado. Esto significa que debe combinarse con otros tipos de pruebas como, pruebas de requisitos (donde verificamos las características especificadas que debían construirse).

**Mejores prácticas**

**Etiquetar y nombrar claramente las pruebas**.

Si alguien más necesita ejecutar las pruebas, debe poder ubicarlas fácilmente. Antes de mejorar el código, es decir, refactorizarlo y optimizarlo para su uso en diferentes entornos, asegúrese de que su estructura y flujo sean ideales.

**Ejecutar pruebas por separado**.

De esta forma, es fácil identificar errores y corregirlos. Por otro lado, es menos probable que pasemos por alto errores o rutas como resultado de superposiciones en secciones de código, bloques o rutas.

**Genere pruebas antes de realizar cambios**.

Se requiere que las pruebas se ejecuten como se esperaba. De esta manera, si algo se rompe, es fácil rastrear y solucionar el problema.

**Mantenga las pruebas para cada sección o bloque de código por separado**.

De esta forma, si hay cambios en el futuro, no necesitamos cambiar muchas pruebas.

**Corrija errores antes de continuar con las pruebas**.

Si identificamos algún error, es mejor que lo solucionemos antes de proceder a probar la siguiente sección o bloque de código.

**Nunca omita las pruebas estructurales con la suposición de que un control de calidad “seguirá realizando pruebas de todos modos”.**

Incluso si los errores pueden parecer insignificantes al principio, acumulativamente, pueden resultar en un código defectuoso que nunca podrá lograr su propósito previsto.

**Preguntas frecuentes**

**¿Cuál es la diferencia entre las pruebas funcionales y las pruebas estructurales?**

La prueba funcional es un tipo de prueba de software que se basa en los requisitos estipulados en las SRS (Especificaciones de requisitos de software). Por lo general, se hace en un intento por encontrar disparidades entre las especificaciones del SRS y cómo funciona el código. Las pruebas estructurales se basan en la estructura interna del código y su implementación. Se requiere una comprensión profunda del código.

**¿Cuáles son los tipos de pruebas estructurales?**

Prueba de flujo de datos

Prueba de mutación

Prueba de flujo de control

Pruebas basadas en cortes

**¿Cuál es la diferencia entre las pruebas de flujo de datos y las pruebas de flujo de control?**

Tanto las pruebas de flujo de datos como las pruebas de flujo de control utilizan gráficos de flujo de control. La única diferencia es que, en las pruebas de flujo de control, nos centramos en las rutas generadas a partir del código, mientras que, en las pruebas de flujo de datos, nos centramos en los valores de los datos, su definición y uso dentro de las rutas identificadas dentro de un programa.

**¿Para qué se utilizan las pruebas de flujo de datos?**

Las pruebas de flujo de datos son ideales para identificar anomalías en el uso de valores de datos dentro de las rutas en un gráfico de flujo de control.

**Por ejemplo,** una variable a la que se le ha asignado valor dos veces, una variable que se ha definido y no se ha utilizado, o una variable que se ha utilizado o referenciado y no se ha definido.

**¿Cuál es la diferencia entre cortar y cortar en cubitos en las pruebas de software?**

Cortar significa centrarse en declaraciones particulares de interés en un programa e ignorar el resto. Cortar en cubos es cuando identificamos un segmento que tiene una entrada incorrecta y luego lo cortamos aún más para rastrear el comportamiento correcto.

**¿Cuál es la diferencia entre las pruebas de mutación y la cobertura del código?**

En las pruebas de mutación, consideramos el número de mutantes muertos como porcentaje del total de mutantes. La cobertura de código es simplemente la cantidad de código que se ha probado en un programa.

**Conclusión**

En este trabajo se analizaron las pruebas estructurales en profundidad: qué es, qué no es, cómo hacerlo, tipos de cobertura, ventajas y desventajas, mejores prácticas e incluso algunas preguntas frecuentes sobre este tipo de prueba de software.

Todavía hay mucho más que podemos aprender sobre las pruebas basadas en estructuras como lo son la cobertura de código (declaración, decisión, rama y ruta), tipos de pruebas estructurales (mutación, flujo de datos y basado en cortes) e incluso las herramientas que podemos usar para automatizar estos procesos de prueba.

Es importante tener en cuenta que no existe ningún tipo o enfoque de prueba de software que sea 100% eficiente. Siempre es recomendable combinar diferentes tipos y enfoques de prueba.

**Por ejemplo,** Las pruebas estructurales se complementan en gran medida con las pruebas de requisitos, ya que puede haber características que no se hayan desarrollado en el momento en que se estaban llevando a cabo las pruebas basadas en estructuras.

**Bibliografía**

* <https://www.softwaretestinghelp.com/types-of-software-testing/>
* <https://spa.myservername.com/structural-testing-tutorial-what-is-structural-testing>
* <https://medium.com/la-region-vulnerable/tipos-de-pruebas-de-software-131ec4ebc012>
* <https://www.panel.es/software-qa-cuales-son-los-tipos-de-pruebas-software/>
* <https://www.youtube.com/watch?v=40U8Ad3uSiM>
* <https://www.youtube.com/watch?v=hSxXuRxA9mo>
* <https://www.youtube.com/watch?v=QFUYSRtX-VM>
* <https://www.youtube.com/watch?v=rOc_v1Uii9s>
* <https://www.loadview-testing.com/es/blog/tipos-de-pruebas-de-software-diferencias-y-ejemplos/>
* <https://es.wikipedia.org/wiki/Pruebas_de_caja_blanca>
* <http://www.pmoinformatica.com/2014/01/tipos-de-pruebas-de-software-istqb.html>
* <http://gestflori.wikidot.com/pruebasestructuralesi>