Tutorial de prueba unitaria: qué es, tipos y ejemplo de prueba

¿Qué es la prueba unitaria?

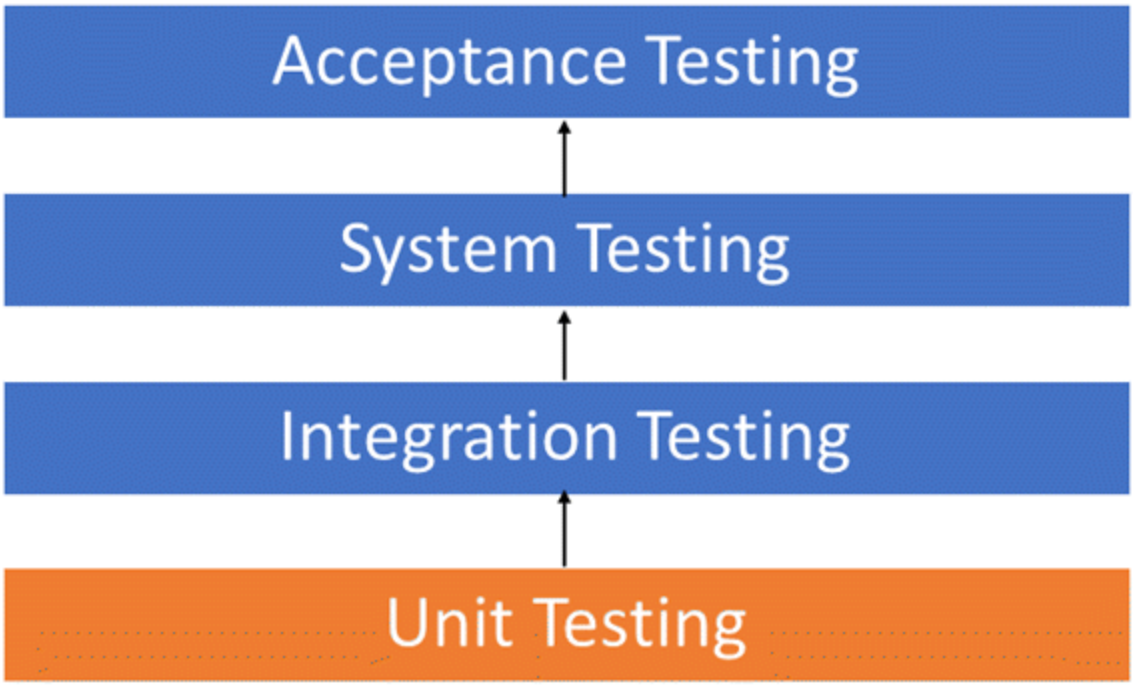
La prueba unitaria es un tipo de prueba de software donde se prueban unidades o componentes individuales de un software. El propósito es validar que cada unidad del código de software funcione como se espera. Las pruebas unitarias se realizan durante el desarrollo (fase de codificación) de una aplicación por parte de los desarrolladores. Las pruebas unitarias aíslan una sección de código y verifican su corrección. Una unidad puede ser una función, método, procedimiento, módulo u objeto individual.

En SDLC, STLC, V Model, la prueba unitaria es el primer nivel de prueba realizado antes de la prueba de integración. La prueba unitaria es una técnica de prueba WhiteBox que generalmente realiza el desarrollador. Sin embargo, en un mundo práctico debido a la escasez de tiempo o la reticencia de los desarrolladores a las pruebas, los ingenieros de control de calidad también realizan pruebas unitarias.

¿Por qué realizar Pruebas Unitarias?

Las pruebas unitarias son importantes porque los desarrolladores de software a veces intentan ahorrar tiempo haciendo pruebas unitarias mínimas y esto es un mito porque las pruebas unitarias inapropiadas conducen a una corrección de defectos de alto costo durante las pruebas del sistema, las pruebas de integración e incluso las pruebas beta después de que se crea la aplicación. Si se realizan pruebas unitarias adecuadas en las primeras etapas del desarrollo, al final se ahorra tiempo y dinero.

Estas son las razones clave para realizar pruebas unitarias en ingeniería de software:



Niveles de prueba unitaria

1. Las pruebas unitarias ayudan a corregir errores al principio del ciclo de desarrollo y ahorran costos.
2. Ayuda a los desarrolladores a comprender la base del código de prueba y les permite realizar cambios rápidamente.
3. Las buenas pruebas unitarias sirven como documentación del proyecto.
4. Las pruebas unitarias ayudan con la reutilización del código. Migre tanto su código **como** sus pruebas a su nuevo proyecto. Modifique el código hasta que las pruebas se ejecuten nuevamente.

Cómo ejecutar pruebas unitarias

Para ejecutar pruebas unitarias, los desarrolladores escriben una sección de código para probar una función específica en la aplicación de software. Los desarrolladores también pueden aislar esta función para realizar pruebas más rigurosas, lo que revela dependencias

innecesarias entre la función que se está probando y otras unidades para que las dependencias puedan eliminarse. Los desarrolladores generalmente usan el marco UnitTest para desarrollar casos de prueba automatizados para pruebas unitarias.

Las pruebas unitarias son de dos tipos.

- Manual
- automatizado

Las pruebas unitarias suelen estar automatizadas, pero aún pueden realizarse manualmente. La ingeniería de software no favorece uno sobre el otro, pero se prefiere la automatización. Un enfoque manual para las pruebas unitarias puede emplear un documento instructivo paso a paso.

Bajo el enfoque automatizado-

- Un desarrollador escribe una sección de código en la aplicación solo para probar la función. Más tarde comentarían y finalmente eliminarían el código de prueba cuando se implemente la aplicación.
- Un desarrollador también podría aislar la función para probarla más rigurosamente. Esta es una práctica de prueba unitaria más completa que implica copiar y pegar código en su propio entorno de prueba que en su entorno natural. **Aislar el código ayuda a revelar dependencias innecesarias entre el código que se está probando y otras unidades o espacios de datos** en el producto. Estas dependencias se pueden eliminar.
- Un codificador generalmente usa un marco UnitTest para desarrollar casos de prueba automatizados. Usando un marco de automatización, el desarrollador codifica criterios en la prueba para verificar la exactitud del código. Durante la ejecución de los casos de prueba, el marco registra los casos de prueba fallidos. Muchos marcos también marcarán e informarán automáticamente, en resumen, estos casos de prueba fallidos . Dependiendo de la gravedad de una falla, el marco puede detener las pruebas posteriores.
- El flujo de trabajo de las pruebas unitarias es 1) Crear casos de prueba 2) Revisar/Reelaborar 3) Línea base 4) Ejecutar casos de prueba.

Técnicas de prueba unitaria

Las **técnicas de prueba unitaria** se clasifican principalmente en tres partes, que son la prueba de caja negra que implica la prueba de la interfaz de usuario junto con la entrada y la salida, la prueba de caja blanca que implica probar el comportamiento funcional de la aplicación de software y la prueba de caja gris que se utiliza para ejecutar la prueba. suites, métodos de prueba, casos de prueba y realización de análisis de riesgos.

Las técnicas de cobertura de código utilizadas en las pruebas unitarias se enumeran a continuación:

- Cobertura de estado de cuenta
- Cobertura de decisión
- Cobertura de Sucursales
- Cobertura de condición
- Cobertura de máquinas de estados finitos

Para obtener más información, consulte https://www.guru99.com/code-coverage.html

Ejemplo de prueba unitaria: objetos simulados

Las pruebas unitarias se basan en la creación de objetos simulados para probar secciones de código que aún no forman parte de una aplicación completa. Los objetos simulados reemplazan las partes faltantes del programa.

Por ejemplo, podría tener una función que necesita variables u objetos que aún no se han creado. En las pruebas unitarias, se contabilizarán en forma de objetos simulados creados únicamente con el propósito de la prueba unitaria realizada en esa sección de código.

Herramientas de prueba unitaria

Hay varios software de prueba de unidad automatizados disponibles para ayudar con las pruebas de unidad. Daremos algunos ejemplos a continuación:

1. Junit : Junit es una herramienta de prueba de uso gratuito utilizada para el lenguaje de programación Java. Proporciona afirmaciones para identificar el método de prueba. Esta herramienta prueba primero los datos y luego los inserta en la pieza de código.
2. NUnit : NUnit es un marco de prueba de unidad ampliamente utilizado para todos los lenguajes .net. Es una herramienta de código abierto que permite escribir scripts manualmente. Admite pruebas basadas en datos que pueden ejecutarse en paralelo.
3. JMockit : JMockit es una herramienta de prueba unitaria de código abierto. Es una herramienta de cobertura de código con métricas de línea y ruta. Permite simular API con sintaxis de grabación y verificación. Esta herramienta ofrece cobertura de línea, cobertura de ruta y cobertura de datos.
4. EMMA : EMMA es un conjunto de herramientas de código abierto para analizar e informar código escrito en lenguaje Java. Emma admite tipos de cobertura como método, línea, bloque básico. Está basado en Java, por lo que no tiene dependencias de bibliotecas externas y puede acceder al código fuente.

5. PHPUnit : PHPUnit es una herramienta de prueba unitaria para el programador de PHP. Toma pequeñas porciones de código que se llaman unidades y prueba cada una de ellas por separado. La herramienta también permite a los desarrolladores utilizar métodos de aserción predefinidos para afirmar que un sistema se comporta de cierta manera.

Esas son solo algunas de las herramientas de prueba unitaria disponibles. Hay muchos más, especialmente para lenguajes C y Java, pero seguramente encontrará una herramienta de prueba unitaria para sus necesidades de programación, independientemente del lenguaje que use.

Desarrollo dirigido por pruebas (TDD) y pruebas unitarias

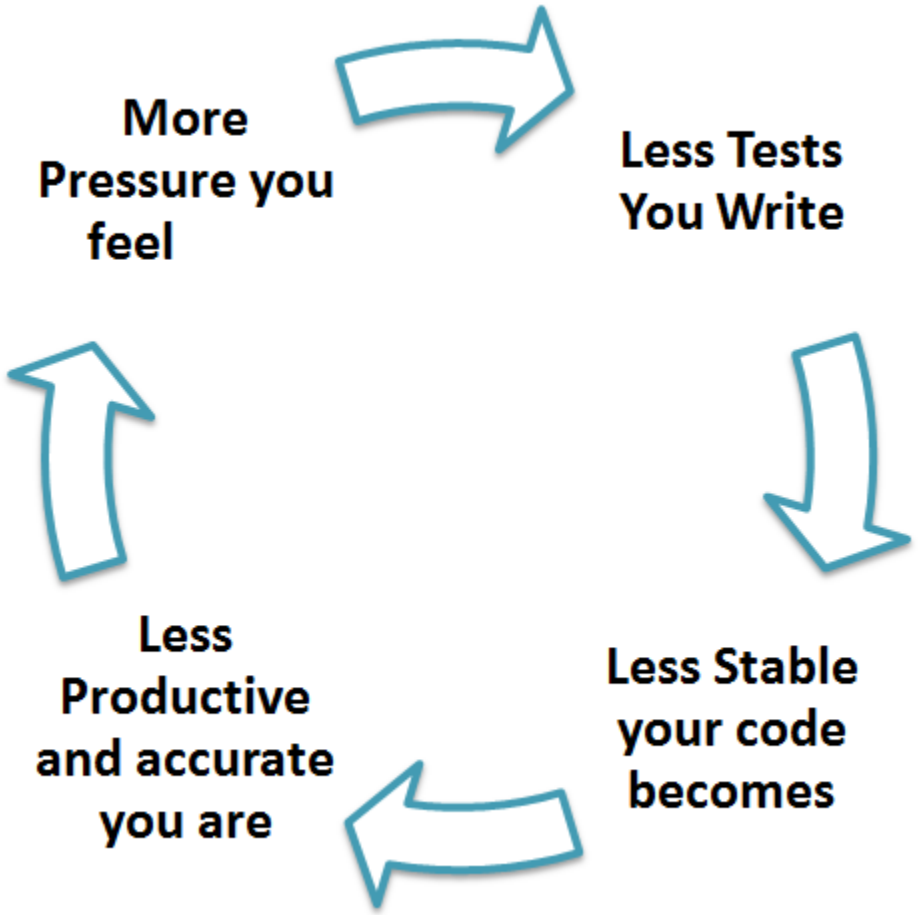
Las pruebas unitarias en TDD implican un uso extensivo de marcos de prueba. Se utiliza un marco de pruebas unitarias para crear pruebas unitarias automatizadas. Los marcos de pruebas unitarias no son exclusivos de TDD, pero son esenciales para él. A continuación, analizamos algunas de las aportaciones de TDD al mundo de las pruebas unitarias:

- Las pruebas se escriben antes que el código.
- Confíe en gran medida en los marcos de prueba
- Se prueban todas las clases en las aplicaciones.
- La integración rápida y fácil es posible

Mito de las pruebas unitarias

Mito: Requiere tiempo, y siempre estoy sobrecargado
. ¡Mi código es sólido como una roca! No necesito pruebas unitarias.

Los mitos, por su propia naturaleza, son suposiciones falsas. Estas suposiciones conducen a un círculo vicioso de la siguiente manera:



La verdad es que las pruebas unitarias aumentan la velocidad de desarrollo.

Los programadores piensan que las pruebas de integración detectarán todos los errores y no ejecutarán la prueba unitaria. Una vez que las unidades están integradas, los errores muy simples que podrían haberse encontrado y corregido muy fácilmente en la unidad probada toman mucho tiempo para rastrearse y corregirse.

Ventaja de las pruebas unitarias

- Los desarrolladores que buscan saber qué funcionalidad proporciona una unidad y cómo usarla pueden consultar las pruebas de unidad para obtener una comprensión básica de la API de la unidad.
- Las pruebas unitarias permiten al programador refactorizar el código en una fecha posterior y asegurarse de que el módulo aún funcione correctamente (es decir, pruebas de regresión). El procedimiento consiste en escribir casos de prueba para todas las funciones y métodos, de modo que cada vez que un cambio cause una falla, se pueda identificar y corregir rápidamente.
- Debido a la naturaleza modular de las pruebas unitarias, podemos probar partes del proyecto sin esperar a que se completen otras.

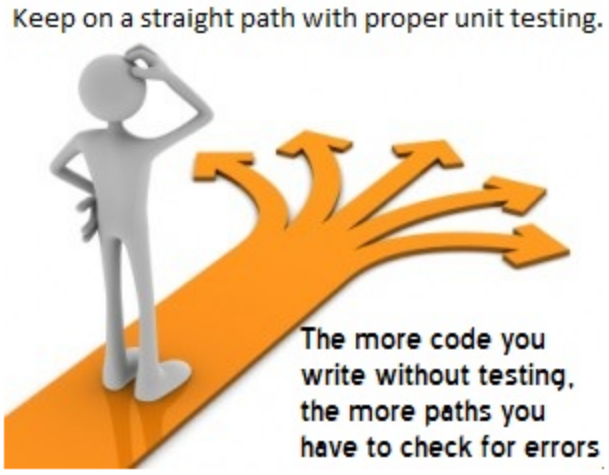
Desventajas de las pruebas unitarias

- No se puede esperar que las pruebas unitarias detecten todos los errores en un programa. No es posible evaluar todas las rutas de ejecución incluso en los programas más triviales
- Las pruebas unitarias, por su propia naturaleza, se centran en una unidad de código. Por lo tanto, no puede detectar errores de integración o errores de nivel general del sistema.

Se recomienda que las pruebas unitarias se utilicen junto con otras actividades de prueba.

Mejores prácticas de pruebas unitarias

- Los casos de prueba unitaria deben ser independientes. En caso de mejoras o cambios en los requisitos, los casos de prueba unitaria no deberían verse afectados.
- Pruebe solo un código a la vez.
- Siga convenciones de nomenclatura claras y consistentes para sus pruebas unitarias
- En caso de un cambio en el código de cualquier módulo, asegúrese de que haya un caso de prueba de unidad correspondiente para el módulo y que el módulo pase las pruebas antes de cambiar la implementación.
- Los errores identificados durante las pruebas unitarias deben corregirse antes de pasar a la siguiente fase en SDLC
- Adopte un enfoque de "prueba como su código". Cuanto más código escriba sin probar, más rutas tendrá para comprobar si hay errores.



Resumen

- PRUEBA DE UNIDAD se define como un tipo de prueba de software donde se prueban unidades o componentes individuales de un software.

- Como puede ver, puede haber muchas cosas involucradas en las pruebas unitarias. Puede ser complejo o bastante simple según la aplicación que se esté probando y las estrategias, herramientas y filosofías de prueba utilizadas. Las pruebas unitarias siempre son necesarias en algún nivel. Eso es una certeza.