2024

Informe de Testing y Pruebas de Código

PARTE 4

JUN XU CHENG

MSMK | PARA: GABRIELA GARCÍA

ÍNDICE DE CONTENIDOS

1. Introducción

El testing y las pruebas de código son esenciales durante la etapa de desarrollo del software, estas prácticas son fundamentales para garantizar la calidad, el rendimiento y la confiabilidad de las aplicaciones. Además de, identificar las posibles vulnerabilidades que puedan contener para posteriormente corregirlos.

2. Conceptos Básicos

2.1. Testing

Verifica y asegurara que un software funcione correctamente y cumpla con los requisitos esperados. Busca identificar fallos/errores antes de la implementación para poder corregirlos.

2.2. Pruebas de código

Es un proceso esencial para poder garantizar la calidad del código, validando el código fuente, además de ser una parte integral del desarrollo de software.

2.3. Objetivo y beneficios de realizar pruebas

Su objetivo principal es identificar errores de manera temprana y mejorar el código.

En cuando a los beneficios, son la disminución de errores y la optimización del desarrollo.

3. Tipos de pruebas

3.1. Pruebas Unitarias

Verifica la precisión de fragmentos aislados, como funciones o un método. Busca verificar que el código se ejecuta como deseaba el desarrollador. Su ejecución debe ser aislada, y cuando se requiera interacción con elementos del sistema, se deben de utilizar fragmentos de datos en lugar de datos externos.

- Herramientas populares: JUnit (Java), pytest (Python), NUnit (.NET).

3.2. Pruebas de Integración

Es un mecanismo de testeo de software que analiza el ensamblaje de componentes y su comportamiento en relación con múltiples partes del sistema. Evalúan la interacción entre unidades, asegurando la funcionalidad de los componentes del software juntos

- Herramientas populares: Postman (API), TestNG (Java).

3.3. Pruebas de Sistema

Asegura el correcto funcionamiento del sistema, verifica que cumple con todas las especificaciones establecidas. Consiste en la integración de todos los módulos y componentes individuales y confirmar su funcionamiento.

- Herramientas populares: Selenium (web), Appium (aplicaciones móviles).

3.4. Pruebas de Aceptación

Confirma que cumple con todos los requisitos del usuario.

- Herramientas populares: Cucumber (BDD), Behave (Python).

3.5. Pruebas de Carga y Estrés

Se somete al sistema una carga de trabajo predefinida para averiguar cuál es la tensión máxima que pueda soportar

- Herramientas populares: JMeter, LoadRunner.

3.6. Pruebas de Seguridad

Son pruebas para poder evaluar la resistencia del software ante ataques.

- Herramientas populares: OWASP ZAP, Nessus.

3.7. Pruebas de Regresión

Sirven para verificar superficialmente nuevas características y asegurar que no degraden la funcionalidad existente, garantizando la estabilidad y consistencia del software,

- Herramientas populares: Selenium, TestNG

4. Técnicas de Testing

4.1. Test Driven Development (TDD)

Se realizan pruebas unitarias antes de escribir el código, obteniendo un código limpio, robusto y simple[1], reduciendo así los costes de mantenimiento.

4.1.1. Beneficios:

- Precicisión y Evitación de Código Innecesario: Evita el código innecesario, centrándose en funcionalidades esenciales.

- Feedback de API: Permite anticipar y solucionar posibles problemas futuros.

- Generación de Documento: Genera gran cantidad de documentación detallada para ayudar a resolver dudas.

- Desarrollo Evolutivo: Permite modificaciones a medida que se crea.

- Interfaz Independiente: Asegura que el diseño no se vea afectada por el código.

- Mejora de la Comunicación: Mejora la comunicación entre equipos, facilitando la resolución de problemas.

- Código Claro y Limpio

- Escalabilidad: Permite un crecimiento constante.

- Satisfacción del Cliente: Mejora la satisfacción del cliente al entregar un producto de calidad.

- Metodología Ágil: TDD incorpora requisitos esenciales.

---

[1] Intelequia. (2023). *¿Qué es y para qué sirve un TDD o Test Driven Development?* [online] Available at: https://intelequia.com/es/blog/post/qu%C3%A9-es-y-para-qu%C3%A9-sirve-un-tdd-o-test-driven-development [Accessed 21 Jan. 2024].

- Reducción de Costes y Tiempo: El seguimiento del proceso reduce costos y tiempo.

4.1.2. Retos:

- **Curva de Aprendizaje:** Comprender cómo diseñar y escribir pruebas efectivas puede llevar un largo periodo de tiempo.

- **Tiempo Adicional Inicial:** La dedicación de tiempo adicional al principio puede ser todo un reto, ya que puede parecer más lento. Sin embargo, se espera una mejora a largo plazo.

- **Complejidad en la Integración:** La integración de TDD en proyectos de gran escala puede ser desafiante, requiriendo esfuerzos adicionales para analizar y comprender el sistema ya existente.

- **Mantenimiento de Pruebas:** El mantenimiento ineficiente puede resultar en escenarios donde se vuelven obsoletas o difícil de comprender.

4.2 Behavior Driven Development (BDD)

Mejora la colaboración entre equipos y en el entendimiento entre desarrolladores, gestores de proyectos y equipos de negocio. Se centra en pruebas relacionadas con el usuario y el comportamiento del sistema.

4.2.1. Beneficios

- Mejora de la Colaboración: Adopta un lenguaje común para que aumente la colaboración entre todas las partes involucradas.

- Priorización del Valor de Negocio: Establecen prioridades con los clientes, permitiendo a los desarrollados entender los objetivos que se piden, obteniendo resultados óptimos alineándose con las necesitades del cliente.

- Reducción de Riegos en el Proyecto: Tanto el cliente como el desarrollador tienen un entendimiento claro del objetivo, por lo tanto, se corre menos riesgos durante el desarrollo e implementación.

- Creación de Metodologías Ágiles: La explicación detallada de los requerimientos y escenarios simplifica el desarrollo.

4.2.2. Retos

- Comunicación Efectiva: Puede llevar a malentendidos en la gestión del proyecto si hay desafíos de comunicación.

- Complejidad en la Escritura de Escenarios: Se requiere de experiencia y habilidad para poder redactar escenarios comprensibles y útiles.

- Necesidad de Participación Activa: Requiere de la participación constante de todas las partes.

- Selección Inadecuada de Herramientas: Es necesario hacer una selección cuidadosa y una compresión profunda de las herramientas para garantizar su efectividad.

4.3. Acceptance Test Driven Development (ATDD)

Implica crear y ejecutar pruebas de aceptación antes de la implementación, asegurando la alineación entre lo que se está haciendo y lo que debería hacerse.

4.3.1. Beneficios

- Eliminación de Trabajo Innecesario: Evitaremos dedicar esfuerzo y tiempo en trabajo que no se utilizarán.

- Flexibilidad para Cambios: Forjando un código adaptable a cambios, sin limitaciones por interfaces específicas o diseños de bases de datos.

- Evaluación Rápida de Objetos: Permite verificar si se cumplen los objetivos.

- Seguimiento del Progreso: Obtenemos una visión clara del proyecto.

- Confianza del Equipo: Permite al manager revisar y verificar el cumplimiento de criterios de aceptación.

4.3.2. Retos

- Aprendizaje inicial: Requiere de un tiempo y esfuerzo adicional para aprender a implementar eficientemente ATDD.

- Documentación Limitada: La documentación puede no ser ilimitada, por lo que puede generar desafíos para resolver dudas.

- Coordinación del Equipo: Requiere de una coordinación efectiva entre todos los miembros en todas las etapas del ciclo.

- Configuración Inicial: Pueden surgir desafíos en la configuración inicial de los ciclos de aceptación y la preparación de entornos y dispositivos de prueba.

5. Automatización de Pruebas

Es el proceso de utilizar herramientas de software que ejecutan software recién desarrollado o actualizaciones a través de una serie de pruebas para identificar posibles errores de codificación, cuellos de botella y otros obstáculos para el rendimiento.[2]

5.1. Ventajas:

Las principales ventajas son:

- Requieren de menos tiempo y son más baratas
- Detectan fallos más rápidamente
- Permite hacer pruebas en áreas específicas
- Son fáciles de ejecutar

5.2. Herramientas y Frameworks populares:

5.2.1. Selenium

Se ha convertido en una herramienta esencial para probar aplicaciones web complejas. Su alta flexibilidad en términos de lenguajes de programación y navegadores, lo convierte en una opción eficaz. Sin embargo, se recomienda usar otras herramientas para las pruebas de API o de aplicaciones móviles.

Aunque requiera conocimientos de programación, Selenium destaca como una excelente opción para la automatización de pruebas.

---

[2] https://www.facebook.com/Zaptest-701779380019387 (2022). *¿Qué es la automatización de pruebas? Una guía sencilla y sin jerga*. [online] ZAPTEST. Available at: https://www.zaptest.com/es/que-es-la-automatizacion-de-pruebas-una-guia-sencilla-y-sin-jerga [Accessed 21 Jan. 2024].

5.2.2. Cypress

Es un framework de automatización basado en pruebas agradables y simples por lo que es perfecta para los que están empezando a automatizar, además de su fácil instalación y configuración.

Cypress tiene algunas limitaciones como que solo es compatible con Chrome y Firefox.

Utiliza JavaScript, es rápido, y facilita la visualización y depuración de pruebas.

5.2.3. Appium

Es un framework compatible con Android e iOS, su compatibilidad con múltiples lenguajes de programación, proporciona flexibilidad a desarrolladores y equipos de pruebas.

Aunque su configuración puede ser un desafío para los principiantes y se señalan problemas de estabilidad en algunos dispositivos y plataformas.

5.2.4. Robot Framework

Está basado en Python, ofrece un enfoque basado en palabras clave para la creación de pruebas legibles y comprensibles. Utiliza la sintaxis “Action Table” que permite a los usuarios a escribir pruebas en un formato fácil de entender.

Puede presentar desafíos en la documentación, pero su adaptabilidad, facilidad de uso y su compatibilidad con diversos lenguajes lo consolidan como una opción en la automación de pruebas

5.2.5. Junit

Es un marco de pruebas unitarias de código abierto para Java. Destaca por su simplicidad y eficacia para la realización de pruebas unitarias. Cuenta con una capacidad para definir y ejecutar pruebas de manera fácil y rápida. Las características clave incluyen la organización de casos de prueba y su integración fluida con herramientas de construcción.

5.2.6. JMeter

Es una herramienta popular para pruebas de carga y rendimiento en aplicaciones web y servicios. Es capaz de simular la carga de usuarios, identificar cuellos de botella y evaluar el rendimiento del sistema.

Destaca por su compatibilidad con varios protocolos, la extensibilidad, y funciones de informes y análisis para interpretar los resultados de las pruebas de carga.

5.2.7. TestNG

Es un marco de pruebas para el lenguaje de programación Java. Aunque, TestNG se inspire en Junit, este ofrece características más avanzadas y flexibles para la ejecución.

Con TestNG, puedes generar un informe adecuado y puedes saber fácilmente cuántos casos de prueba se aprobaron, fallaron y se omitieron. Además de poder ejecutar los casos de prueba fallidos por separado.[3]

6. Caso de usos y ejemplos

6.1. Pruebas Unitarias

Caso de uso: En el desarrollo de una aplicación bancaria.

Aplicación: Creación de pruebas unitarias para funciones como el cálculo de intereses o la validación de transacciones.

6.2. Pruebas de Integración:

Caso de uso: En un sistema de planificación de recursos empresariales.

Aplicación: Verificación de la integración fluida entre módulos, como la conexión entre el módulo de finanzas y el de recursos humanos.

6.3. Pruebas de Sistema:

Caso de uso: Desarrollo de un sistema de compra de entradas para el cine

---

[3] Guru99. (2023). *TestNG vs JUnit: diferencia entre ellos*. [online] Available at: https://www.guru99.com/es/junit-vs-testng.html#:~:text=TestNG%20es%20un%20marco%20basado,de%20c%C3%B3digo%20abierto%20para%20JAVA. [Accessed 21 Jan. 2024].

Aplicación: Confirma que el sistema ha comprado la entrada, emite la entrada y actualiza la disponibilidad de butacas.

6.4. Pruebas de Aceptación:

Caso de uso: Desarrollo de una tienda online.

Aplicación: Validación de funciones desde la perspectiva del usuario, como agregar productos al carrito o realizar pagos.

6.5. Pruebas de Carga y Estrés

Caso de uso: En una red social como Twitter.

Aplicación: Evalúan la capacidad del sistema ante un gran volumen de usuarios y publicaciones.

6.6. Pruebas de Seguridad:

Caso de uso: Aplicaciones de banca online.

Aplicación: Protección contra ataques de inyección SQL.

6.7. Pruebas de Regresión:

Caso de uso: Actualización de un software

Aplicación: Confirmar que las nuevas características no afectan negativamente a las funciones ya existentes.

6.8. Test Driven Development (TDD)

Caso de uso: Desarrollo de un componente de un software para análisis de datos.

Aplicación: Escribir pruebas antes de la implementación para garantizar que el código cumple con los requisitos deseados.

6.9. Behavior Driven Development (BDD)

Caso de uso: Desarrollo de un sistema de compra de entradas para el cine

Aplicación: Cuando va a ir a pagar, deber de ver una notificación de la compra y recibir un correo electrónico de confirmación.

6.10. Acceptance Test Driven Development (ATDD)

Caso de uso: Desarrollo de un sistema como Uber.

Aplicación: Si el usuario selecciona el destino, especifica la hora de recogida y confirmación del pago, entonces debería de recibir un correo de confirmación y un número de seguimiento.

7. Conclusión

El testing y las pruebas de código son esenciales para garantiza la calidad, el rendimiento y la seguridad de las aplicaciones en el desarrollo de software. El uso de diversas técnicas y la automatización no solo mejoran la eficiencia sino ahorran costes a futuro y fortalecen la entrega de un producto de alta calidad. Invertir en pruebas es crucial para asegurar un entorno seguro y que además cumpla con los requisitos del usuario.

8. Bibliografía

Ibm.com. (2016). *¿Qué es la prueba de software y cómo funciona? | IBM*. [online] Available at: https://www.ibm.com/es-es/topics/software-testing [Accessed 20 Jan. 2024].

Testingit.com.mx. (2022). *¿Qué son las pruebas unitarias de software?* [online] Available at: https://www.testingit.com.mx/blog/pruebas-unitarias-de-software#:~:text=Una%20prueba%20unitaria%20de%20software,funci%C3%B3n%2C%20proceso%20o%20actividad%20espec%C3%ADfica. [Accessed 20 Jan. 2024].

Amazon Web Services, Inc. (2021). *¿Qué son las pruebas unitarias?: explicación de las pruebas unitarias en AWS*. [online] Available at: https://aws.amazon.com/es/what-is/unit-testing/ [Accessed 20 Jan. 2024].

Constantin Singureanu (2023). *¿Qué es la comprobación de sistemas? Una inmersión en profundidad en enfoques, tipos, herramientas, consejos y trucos, ¡y mucho más!* [online] ZAPTEST. Available at: https://www.zaptest.com/es/que-es-la-comprobacion-de-sistemas-una-inmersion-en-profundidad-en-enfoques-tipos-herramientas-consejos-y-trucos-y-mucho-mas#:~:text=La%20prueba%20de%20sistemas%20es,funciona%20conjuntamente%20como%20se%20esperaba. [Accessed 20 Jan. 2024].

Dotcom-monitor.com. (2023). *Pruebas de rendimiento en línea (carga y estrés) con LoadView*. [online] Available at: https://www.dotcom-monitor.com/wiki/es/knowledge-base/solucion-de-prueba-de-carga/#:~:text=%C2%BFQu%C3%A9%20es%20las%20pruebas%20de%20estr%C3%A9s%3F,funcionamiento%20normales%20o%20se%20bloquea. [Accessed 20 Jan. 2024].

Inesdi. (2022). *¿Qué es el TDD o Test Driven Development?* [online] Available at: https://www.inesdi.com/blog/que-es-TDD-test-driven-development/ [Accessed 21 Jan. 2024].

Zapater, S. (2022). *BDD Testing. ¿Cómo funciona el Behavior Driven Development? - Blog de hiberus*. [online] Blog de hiberus. Available at:

https://www.hiberus.com/crecemos-contigo/bdd-behavior-driven-developement/ [Accessed 21 Jan. 2024].

https://www.facebook.com/Zaptest-701779380019387 (2022). *¿Qué es la automatización de pruebas? Una guía sencilla y sin jerga*. [online] ZAPTEST. Available at: https://www.zaptest.com/es/que-es-la-automatizacion-de-pruebas-una-guia-sencilla-y-sin-jerga [Accessed 21 Jan. 2024].

Cleveritgroup.com. (2023). *5 Frameworks comunes en Automatización de Pruebas y cuándo usarlos | QA*. [online] Available at: https://www.cleveritgroup.com/blog/5-frameworks-comunes-en-automatizacion-de-pruebas-y-cuando-usarlos [Accessed 21 Jan. 2024].

Guru99. (2023). *TestNG vs JUnit: diferencia entre ellos*. [online] Available at: https://www.guru99.com/es/junit-vs-testng.html#:~:text=TestNG%20es%20un%20marco%20basado,de%20c%C3%B3digo%20abierto%20para%20JAVA. [Accessed 21 Jan. 2024].

https://www.facebook.com/grokkeepcoding (2023). *¿Qué es ATDD (Acceptance Test Driven Development)?* [online] KeepCoding Bootcamps. Available at: https://keepcoding.io/blog/que-es-atdd-acceptance-test-driven-development/ [Accessed 21 Jan. 2024].