UNIVERSIDAD DOCTOR ANDRÉS BELLO

FACULTAD: TECNOLOGÍA E INNOVACIÓN

CARRERA: INGENIERÍA EN SISTEMAS Y COMPUTACIÓN

ASIGNATURA: “TÉCNICAS DE CALIDAD DE SOFTWARE”

TEMA: “Cómo controlar la calidad de software y la Cobertura de pruebas funcionales”

MATERIAL DE LECTURA DE UNIDAD 3: DEFINICIÓN DE ESTRATEGIA DE PRUEBAS.

NOMBRE DEL DOCENTE: ING. RUBEN EDGARDO PORTILLO

Introducción

La calidad del software es un aspecto fundamental en el desarrollo de productos tecnológicos y garantiza que los sistemas cumplan con los requisitos del usuario y los estándares de la industria. En la actualidad, la alta competencia y las crecientes expectativas de los usuarios exigen que las empresas desarrollen productos de software de alta calidad, que no solo funcionen de manera correcta, sino que también sean seguros, eficientes y fáciles de usar. La estrategia de pruebas juega un papel crucial en este contexto, ya que permite identificar y corregir errores antes de que el software llegue al usuario final.

Esta unidad aborda dos aspectos esenciales para asegurar la calidad del software: cómo controlar la calidad durante el desarrollo y la importancia de la cobertura de pruebas funcionales. Controlar la calidad implica la implementación de diversas técnicas y herramientas de pruebas que aseguren que el software cumpla con los estándares de calidad establecidos. Por otro lado, la cobertura de pruebas funcionales es una métrica crítica que permite evaluar qué tan exhaustivas son las pruebas en relación con los requisitos funcionales del sistema.

Objetivos o competencias a adquirir

Al finalizar esta unidad, los estudiantes serán capaces de:

- Comprender la importancia de controlar la calidad del software durante el proceso de desarrollo.
- Identificar y aplicar diferentes estrategias de control de calidad de software.
- Evaluar la cobertura de pruebas funcionales y su impacto en la calidad del software.
- Aplicar técnicas para mejorar la cobertura de pruebas funcionales en proyectos de software.
- Desarrollar habilidades críticas para diseñar estrategias de pruebas efectivas que aseguren la calidad de los productos de software.

Contenido

1. Cómo Controlar la Calidad de Software

1.1. Definición de Control de Calidad en Software

El control de calidad en el desarrollo de software se refiere a un conjunto de actividades diseñadas para evaluar la calidad de los productos de software a lo largo de su ciclo de vida y asegurarse de que cumplen con los estándares de calidad establecidos. Estas actividades incluyen revisiones, pruebas, auditorías y otros procesos que identifican defectos y aseguran que el software se ajuste a los requisitos especificados.

1.2. Estrategias para el Control de la Calidad de Software

1. **Pruebas de Software**: Las pruebas son el método más común para controlar la calidad. Involucran la ejecución de los programas con el fin de identificar defectos. Existen diferentes tipos de pruebas, como pruebas unitarias, de integración, de sistema y de aceptación, cada una con un enfoque específico para detectar errores en diferentes niveles del software (Myers, Sandler, & Badgett, 2011).
2. **Revisiones de Código**: Las revisiones de código son inspecciones manuales realizadas por desarrolladores para identificar errores antes de la ejecución. Estas revisiones pueden ser informales, como revisiones entre pares, o formales, como inspecciones estructuradas que siguen un proceso definido (Fagan, 1976).
3. **Integración Continua (CI)**: CI es una práctica que permite integrar y probar el código de manera continua, lo que facilita la detección temprana de errores. Esta práctica mejora la calidad al asegurar que el código integrado funcione correctamente en conjunto (Fowler & Foemmel, 2006).
4. **Análisis Estático de Código**: El análisis estático evalúa el código fuente sin ejecutarlo, utilizando herramientas automatizadas para identificar defectos y malas prácticas de programación. Esta técnica es eficaz para detectar problemas de calidad como errores de sintaxis, violaciones de estándares de codificación y posibles vulnerabilidades (Ayewah et al., 2008).
5. **Gestión de Configuración**: Involucra la gestión y el control de cambios en los artefactos del software, asegurando que los elementos del sistema estén correctamente versionados y probados. Esto ayuda a mantener la integridad del producto a lo largo del ciclo de vida del desarrollo (Bersoff, Henderson, & Siegel, 1980).

1.3. Herramientas para Controlar la Calidad del Software

Las herramientas automatizadas juegan un rol fundamental en el control de la calidad del software, ya que facilitan la identificación de defectos y aseguran que los estándares de calidad se cumplan consistentemente.

- **JUnit**: Framework para la realización de pruebas unitarias en Java.
- **Selenium**: Herramienta para pruebas automatizadas de aplicaciones web.
- **SonarQube**: Plataforma que permite el análisis continuo de la calidad del código y la detección de vulnerabilidades.
- **Jenkins**: Herramienta de integración continua que facilita la automatización de pruebas y la implementación de la calidad del software.

2. Cobertura de Pruebas Funcionales

2.1. Definición de Cobertura de Pruebas Funcionales

La cobertura de pruebas funcionales mide la extensión en la que los casos de prueba ejecutan las funcionalidades especificadas del software. Es una métrica clave que evalúa la efectividad de las pruebas y garantiza que se cubran todos los requisitos del sistema. Esta cobertura se centra en validar que el software cumple con lo que se espera de él desde el punto de vista del usuario final (Kaner, Falk, & Nguyen, 1999).

2.2. Tipos de Cobertura de Pruebas Funcionales

1. **Cobertura de Requisitos**: Evalúa si todos los requisitos funcionales del software han sido probados. Esta cobertura se mide mediante el mapeo de casos de prueba a los requisitos especificados, asegurando que no se omitan funcionalidades críticas.
2. **Cobertura de Casos de Uso**: Se basa en la validación de los casos de uso definidos en el diseño del sistema. Asegura que todas las interacciones posibles entre el usuario y el sistema sean probadas.
3. **Cobertura de Ramas y Decisiones**: Evalúa las rutas de decisión dentro del software para asegurarse de que todas las condiciones posibles han sido ejecutadas al menos una vez.

2.3. Técnicas para Mejorar la Cobertura de Pruebas Funcionales

1. **Análisis de Rastreabilidad**: Relaciona los casos de prueba con los requisitos del software para asegurar que todos los aspectos funcionales estén cubiertos. Esto facilita la identificación de áreas no probadas y la creación de casos de prueba adicionales.
2. **Generación Automática de Pruebas**: Utilización de herramientas automatizadas que generan casos de prueba basados en modelos del

sistema. Estas herramientas pueden identificar automáticamente escenarios de prueba adicionales para aumentar la cobertura.

3. **Pruebas Basadas en Modelos**: Involucran la creación de modelos que representan el comportamiento del software, a partir de los cuales se generan casos de prueba. Esta técnica ayuda a cubrir escenarios complejos y verificar que todas las funcionalidades especificadas se comporten como se espera (Utting & Legeard, 2010).
4. **Análisis de Código**: La revisión del código puede revelar rutas no cubiertas por las pruebas funcionales. Esto ayuda a identificar áreas críticas del código que no están siendo probadas adecuadamente y requiere una atención adicional.

2.4. Beneficios de una Cobertura de Pruebas Funcionales Alta

- **Mayor Fiabilidad**: Garantiza que todas las funcionalidades críticas del sistema estén probadas, reduciendo el riesgo de fallos en producción.
- **Mejora de la Satisfacción del Usuario**: Un software con alta cobertura de pruebas funcionales suele cumplir mejor con las expectativas del usuario, lo que se traduce en una mejor experiencia.
- **Reducción de Costos de Mantenimiento**: Identificar y corregir errores en etapas tempranas del ciclo de vida del desarrollo reduce significativamente los costos de corrección y mantenimiento.

Conclusiones

Controlar la calidad del software y asegurar una alta cobertura de pruebas funcionales son aspectos fundamentales para el desarrollo de productos robustos y confiables. Las técnicas y estrategias discutidas en esta unidad proporcionan una base sólida para implementar prácticas de prueba efectivas que aseguren que el software cumpla con los requisitos funcionales y de calidad esperados. El uso de herramientas automatizadas, la integración continua y la revisión constante de la cobertura de pruebas son estrategias clave que los desarrolladores y probadores deben adoptar para mejorar la calidad del software de manera continua.

Referencias Bibliográficas

Ayewah, N., Pugh, W., Morgenthaler, J. D., Penix, J., & Zhou, Y. (2008). Using static analysis to find bugs. IEEE Software, 25(5), 22-29.

Bersoff, E. H., Henderson, V. J., & Siegel, S. G. (1980). Software Configuration Management: An Investment in Product Integrity. Proceedings of the IEEE, 68(9), 1298-1306.

Fagan, M. E. (1976). Design and code inspections to reduce errors in program development. IBM Systems Journal, 15(3), 182-211.

Fowler, M., & Foemmel, M. (2006). Continuous Integration. ThoughtWorks.

Kaner, C., Falk, J., & Nguyen, H. Q. (1999). Testing Computer Software (2nd ed.). Wiley.

Myers, G. J., Sandler, C., & Badgett, T. (2011). The Art of Software Testing (3rd ed.). John Wiley & Sons.