Contents

Clase 8: Inspección y Pruebas de Software

1. Objetivos de la Clase

- Comprender la importancia de la inspección y las pruebas en el aseguramiento de la calidad del software.
- Diferenciar entre inspección y pruebas, identificando sus fortalezas y debilidades.
- Conocer las diferentes técnicas de inspección de software (revisiones formales, walkthroughs, inspecciones técnicas).
- Comprender los diferentes niveles de prueba de software (unitaria, integración, sistema, aceptación).
- Seleccionar las técnicas de inspección y prueba apropiadas para diferentes tipos de proyectos de software.

2. Contenido Teórico Detallado

2.1. Introducción a la Inspección y Pruebas

La inspección y las pruebas son actividades cruciales en el ciclo de vida del desarrollo de software que tienen como objetivo verificar y validar que el software cumple con los requisitos especificados y funciona correctamente.

- **Inspección:** Implica la revisión estática del código, documentos de diseño, casos de prueba, y otros artefactos del software por parte de un equipo de revisores. El objetivo es identificar defectos, inconsistencias y desviaciones de los estándares sin ejecutar el código.
- **Pruebas:** Implica la ejecución dinámica del software con el propósito de encontrar errores, verificar que las funciones operan según lo esperado y evaluar el rendimiento y la confiabilidad.

2.2. Diferencias Clave entre Inspección y Pruebas

| Característica | Inspección | Pruebas | | :---------- | :---------- | :---------- | | **Enfoque** | Estático (revisión de código y documentos) | Dinámico (ejecución del software) | | **Detección** | Defectos lógicos, inconsistencias, violaciones de estándares | Errores de ejecución, problemas de rendimiento, fallas de usabilidad | | **Fases** | Tempranas en el ciclo de vida (diseño, codificación) | Tardías en el ciclo de vida (después de la codificación) | | **Costo** | Generalmente menos costosa para defectos detectados tempranamente | Más costosa para defectos detectados tardíamente | | **Herramientas** | Listas de verificación, herramientas de revisión de código, herramientas de comparación de documentos. | Entornos de prueba, herramientas de automatización de pruebas, depuradores. |

2.3. Técnicas de Inspección de Software

- **Revisiones Formales:** Proceso estructurado con roles definidos (moderador, revisor, autor) y un conjunto de pasos específicos (planificación, preparación, reunión de revisión, reelaboración, seguimiento) para identificar defectos.
- **Walkthroughs:** Reuniones informales donde el autor del código o documento presenta el trabajo a un grupo de colegas para obtener retroalimentación y detectar problemas.
- **Inspecciones Técnicas:** Similar a las revisiones formales, pero con un enfoque más técnico y detallado en el código o diseño. Se utilizan listas de verificación específicas para el tipo de artefacto que se está revisando.

2.4. Niveles de Prueba de Software

- **Prueba Unitaria:** Verifica el funcionamiento correcto de unidades individuales de código (funciones, métodos, clases). Se realiza por los desarrolladores utilizando *mocks* y *stubs* para aislar la unidad bajo prueba.
- **Prueba de Integración:** Verifica la interacción correcta entre diferentes unidades o componentes del software. Puede ser incremental (de abajo hacia arriba o de arriba hacia abajo) o *big-bang* (integración de todos los componentes a la vez).

- **Prueba de Sistema:** Verifica que el sistema completo cumple con los requisitos funcionales y no funcionales. Se realiza por un equipo de pruebas independiente. Incluye pruebas de rendimiento, seguridad, usabilidad, y confiabilidad.
- **Prueba de Aceptación:** Verifica que el sistema cumple con las necesidades del usuario final y es aceptable para su despliegue. Se realiza por los usuarios finales o clientes. Existen pruebas *alpha* (en el entorno del desarrollador) y *beta* (en el entorno del cliente).

3. Ejemplos y Casos de Estudio

- **Caso de Estudio: Revisión Formal en un Proyecto de Software Médico.** Un equipo de desarrollo está construyendo un software para la gestión de registros médicos electrónicos. Realizan revisiones formales del código para asegurar el cumplimiento de las regulaciones de HIPAA y para identificar posibles errores que podrían comprometer la seguridad de los datos de los pacientes.
- **Ejemplo: Prueba Unitaria para una Función de Cálculo de Impuestos.** Un desarrollador escribe una función en Python que calcula el impuesto sobre la renta. Crea una serie de pruebas unitarias utilizando el framework *unittest* para verificar que la función devuelve los resultados correctos para diferentes ingresos y escenarios fiscales.

4. Problemas Prácticos y Ejercicios con Soluciones

- **Ejercicio 1:** Describe el proceso de una revisión formal de código, incluyendo los roles, las actividades y los resultados esperados.
  - **Solución:** El proceso incluye la planificación (selección de los revisores y la programación de la reunión), la preparación (revisión individual del código), la reunión de revisión (discusión y identificación de defectos), la reelaboración (corrección de los defectos por parte del autor), y el seguimiento (verificación de las correcciones). Los roles incluyen el moderador, el autor y los revisores. Los resultados esperados son una lista de defectos y un código corregido.
- **Ejercicio 2:** Diseña un conjunto de pruebas unitarias para una función que valide si una contraseña cumple con los siguientes criterios: (a) al menos 8 caracteres, (b) al menos una letra mayúscula, (c) al menos un número, (d) al menos un símbolo.
  - **Solución:** “‘python import unittest

    def validar_password(password): tiene_longitud = len(password) >= 8 tiene_mayuscula = any(c.isupper() for c in password) tiene_numero = any(c.isdigit() for c in password) tiene_simbolo = any(not c.isalnum() for c in password) return tiene_longitud and tiene_mayuscula and tiene_numero and tiene_simbolo

    class TestValidarPassword(unittest.TestCase):

    ```
    def test_password_valida(self):
        self.assertTrue(validar_password("P@sswOrd1"))

    def test_password_corta(self):
        self.assertFalse(validar_password("P@ss1"))

    def test_password_sin_mayuscula(self):
        self.assertFalse(validar_password("p@ssword1"))

    def test_password_sin_numero(self):
        self.assertFalse(validar_password("P@sswOrd"))

    def test_password_sin_simbolo(self):
        self.assertFalse(validar_password("PasswOrd1"))
    ```

    if **name** == ’**main**’: unittest.main() “‘

5. Materiales Complementarios Recomendados

- **Libros:**
  - "Software Testing" de Ron Patton
  - "The Art of Software Testing" de Glenford J. Myers, Corey Sandler, Tom Badgett
- **Artículos:**
  - IEEE Transactions on Software Engineering: artículos sobre técnicas de inspección y pruebas.
- **Sitios Web:**
  - ISTQB (International Software Testing Qualifications Board): https://www.istqb.org/
  - ASQ (American Society for Quality): https://asq.org/
- **Videos:**
  - Tutoriales de inspección de código en YouTube.
  - Charlas de expertos sobre pruebas de software en plataformas como InfoQ.