PRÁCTICA 2
TEMA 3

LUIS LEÓN, MARIO JIMÉNEZ, GUSTAVO HERNÁNDEZ Y ÓSCAR DÍAZ

ÍNDICE

1. DEFINICIÓN PRUEBA DE CAJA BLANCA

La “prueba de caja blanca” es una **técnica de prueba** que evalúa el **código** y la **estructura interna** de un programa. Se basa en un minucioso examen de los detalles procedimentales del **código** a evaluar, por lo que es necesario conocer la lógica del programa.

Este tipo de pruebas se centran en analizar cada uno de los posibles caminos en el **flujo de ejecución** de un programa ante unos valores de **entrada** concretos. Ejemplificando, si ante unos parámetros de un método, el flujo del programa ejecuta los bucles “if” o los “else”, o entra en un bucle o sale de él.

Con esto, se pretende comprobar que el **código fuente** está preparado para funcionar en todas las situaciones que se puedan presentar, basándose en la inspección detallada de dicho código.

Así es cómo el **software** se prepara para responder a circunstancias susceptibles de provocar **errores**. Estas pruebas tratan de buscar errores obligando a ejecutar todos los posibles flujos de ejecución.

El objetivo de la técnica es diseñar **casos de prueba** para que se ejecuten, al menos una vez, todas las **sentencias** del programa, y todas las decisiones **lógicas** tanto verdaderas como falsas.

Mediante este tipo de prueba, el ingeniero del software en cuestión puede obtener casos de prueba que ejecuten por lo menos una vez cada **instrucción** del programa (siendo esta la prueba del camino básico, que se basa en la complejidad del flujo de ejecución desglosado en un conjunto básico de caminos), que ejecuten los **bucles** (probando el caso general y los casos extremos, siendo esta la prueba de bucles que tiene como objetivo reducir la posibilidad de pasar por alto algún error no encontrado en otro tipo de prueba).

Las principales **técnicas** de diseño de pruebas de caja blanca son:

- Pruebas de **flujo de control** (con el objetivo de encontrar errores en la parte lógica del programa).
- Pruebas de **flujo de datos** (con el objetivo de probar las variables y definiciones en el programa, se hace una selección del flujo de datos para llegar a una conclusión correcta).
- Pruebas de **bifurcación** (se define si algún bucle está correctamente implementado y si las líneas de código donde exista una condición son la mejor opción o si deberían cambiarse).

Por todo esto se considera a este tipo de prueba como uno de los más importantes que se aplican en los software, consiguiendo el resultado de disminuir en un gran porcentaje el número de **errores** existentes en los sistemas y por ende una mayor **calidad** y **confiabilidad**.

2. DEFINICIÓN PRUEBA DE CAMINO BÁSICO

Se trata de un tipo de técnica de diseño de pruebas de caja blanca con la que se consiguen **casos** de prueba de caja blanca. En esta prueba se **verifica** que un programa **funciona** correctamente; para ello, cada una de las **instrucciones** del programa debe haberse ejecutado al menos una vez correctamente.

Esta técnica permite obtener una medida de la **complejidad lógica** del diseño procedimental para utilizar esa medida como una orientación para definir un **conjunto básico** de un diseño de casos de prueba de caminos de ejecución.

Los casos de prueba derivados del conjunto básico garantizan que, durante la prueba, se ejecuta por lo menos una vez cada **sentencia** del programa.

Con esto, se pretende derivar casos de prueba a partir de un conjunto dado de caminos independientes por los cuales puede circular el **flujo de control**. Para conseguir este conjunto de caminos independientes, se crea el **grafo de flujo** asociado y se calcula su **complejidad ciclomática**.

Un **grafo** es un conjunto de objetos llamados **nodos**, unidos por arcos que posibilitan la posibilidad de representar relaciones binarias entre elementos de un conjunto.

3. PASOS

Las pruebas de caja blanca se desarrollan en dos sencillos pasos:

- **Paso 1**) Sobre el **código fuente**: este consiste en poder detectar **problemas de seguridad** y prevenir **ataques** de piratas informáticos y usuarios ingenuos que puedan inyectar código malicioso en la aplicación mediante un evaluador. El evaluador debe aprender y comprender el código fuente de dicha aplicación.

- **Paso 2**) Crear **casos de prueba** y rendimiento: consiste en realizar pruebas del código fuente de la aplicación para determinar si el flujo y la estructura son los adecuados (este paso lo realizará el probador).
  Para este paso es necesario que el evaluador conozca de primera mano el código, lo que a veces lo convierte en **desarrollador**.

Además de esto, también existen **técnicas** para la realización de estas pruebas.
Sin embargo, lo más importante es el **análisis de cobertura de código**, que consiste en identificar **lagunas** en un caso de prueba y crear casos de prueba para verificar las partes del código que no se han probado, mejorando la **calidad** del producto de software.
Esta técnica puede ser automatizada por dos herramientas (las más importantes):

- Cubierta de **declaración**.
- Cubierta de **rama**.

En las pruebas de caja blanca existen varios tipos de pruebas que se pueden evaluar según la **usabilidad** de la aplicación:

- Prueba de **unidad**: es realizada por el programador y consiste en en desarrollar y probar **líneas de código**, una sola **función** o un **objeto** para asegurarse de que funciona antes de que “Unit Testing” continúe ayudando a identificar la mayoría de los errores.
  Las fallas identificadas son más **baratas** y fáciles de solucionar.
- Prueba de **pérdida de memoria**: consiste en detectar las **fugas de memoria** para evitar tener una aplicación software de ejecución lenta. Es recomendable contar con un **especialista** en control de calidad.
- Prueba de **penetración**: consiste en atacar el código desde varios ángulos para exponer las **amenazas** a la seguridad.
- Prueba de **transformación** de caja blanca: consiste en determinar las mejores técnicas de **codificación** para ampliar una solución de software.

Realizar estas pruebas conlleva una serie de ventajas, pero a su vez una cantidad similar de desventajas.

Las ventajas más significativas son:

- Optimizar el código encontrando **errores ocultos**.
- Los casos de pruebas de caja se pueden **automatizar** fácilmente.
- La prueba es mucho más **completa** ya que cubre todas las rutas de código.
- Además, las pruebas pueden comenzar temprano en **SDLC**, incluso si no hay una GUI disponible.

Sin embargo, también presentan desventajas:

- A su vez, las pruebas de caja blanca pueden ser bastante **complicadas** y **costosas**.
- Las pruebas de caja blanca de los desarrolladores no pueden dar lugar a **errores de producción** y, de requerir recursos profesionales, con una comprensión detallada de la programación y la implementación, también necesitan una **prueba de tiempo** en un recuadro blanco.

La representación del flujo del control se realizará con un **grafo de flujo**. Cada nudo representará una **secuencia** del código fuente; de esta manera, se puede representar cualquier código de un programa en un grafo de flujo.
No obstante, hay que tener en cuenta tres componentes que son los que permiten comprender, elaborar y obtener **información** para conocer si el trabajo que se está haciendo se hace adecuadamente.

4. COMPONENTES

- **Nodo**: cada círculo se denomina nodo, el cual representa una o más **secuencias procedimentales**. Un solo nodo puede corresponder a una secuencia de **procesos** o a una sentencia de **decisión**.
  Puede haber nodos que no se asocien. Se suelen utilizar al inicio y final del **grafo**.
- **Aristas** : las flechas se denominan aristas y representan el **flujo de control**. Una arista debe terminar en un nodo aunque este no represente ninguna **sentencia procedimental**.
- **Regiones**: las regiones son las **áreas** delimitadas por las aristas y nodos, incluyéndose el área exterior del grafo. Las regiones se enumeran, siendo la cantidad de regiones equivalente a la cantidad de **caminos independientes** del conjunto básico de un programa.
- **Complejidad Ciclomática**: la complejidad ciclomática es una **métrica** de software extremadamente útil, pues proporciona medición cuantitativa de la complejidad lógica de un programa.
  La complejidad ciclomática define el número de **caminos independientes** del conjunto básico de un programa y da un límite superior para el número de pruebas que se deben realizar para asegurar que se ejecute cada sentencia al menos una vez.
  Un camino independiente es cualquier camino del programa que introduce por lo menos un nuevo conjunto de **sentencias de procesamiento** o una **nueva condición**. El camino independiente se debe mover al menos por una arista que no haya sido recorrida anteriormente.
- **Derivación de casos de prueba** : luego de tener elaborados los grafos de flujos y los caminos a recorrer, se preparan los casos de prueba que forzará la **ejecución** de cada uno de esos caminos. Se escogen los **datos** de forma que las condiciones de los nodos predicados estén adecuadamente establecidas, con el fin de comprobar cada camino.

5. WEBGRAFÍA

- https://es.wikipedia.org/wiki/Pruebas_de_caja_blanca
- https://www.monografias.com/docs113/ingenieria-software-prueba-caja-blanca-y-camino-basico/ingenieria-software-prueba-caja-blanca-y-camino-basico.shtml
- http://www.jc-mouse.net/ingenieria-de-sistemas/caja-blanca-prueba-del-camino-basico
- https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwjg3_2fy_X0AhUJDmMBHSDWBAUQFnoECAMQAQ&url=https%3A%2F%2Fjddiosvivar.files.wordpress.com%2F2014%2F06%2F2-1-tipos-de-pruebas.pdf&usg=AOvVaw3AFhwBSm9AvZVRdp42jPlO
- https://es.wikipedia.org/wiki/Pruebas_de_caja_blanca
- https://ebooksonline.es/que-es-una-prueba-de-caja-blanca-tecnicas-muestras-y-tipos/
- https://sistemasumma.com/2011/06/11/diseno-procedimental/
- https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwi1ns-rz_X0AhUu7rsIHeGNApIQFnoECB8QAQ&url=https%3A%2F%2Fwww.infor.uva.es%2F~jvalvarez%2Fdocencia%2Ftema7.pdf&usg=AOvVaw04_k3omvsnIBBos_UFUq2I
- https://docplayer.es/23351057-Proceso-de-compilacion-sobre-la-tecnica-de-prueba-de-caja-blanca-camino-basico.html
- https://www.monografias.com/docs113/ingenieria-software-prueba-caja-blanca-y-camino-basico/ingenieria-software-prueba-caja-blanca-y-camino-basico