Tecnicatura Universitaria en Programación
Programación I

Apuntes para testear tus algoritmos y código

Este resumen adapta conceptos y técnicas de testing para la materia de programación I.

Índice

Testing

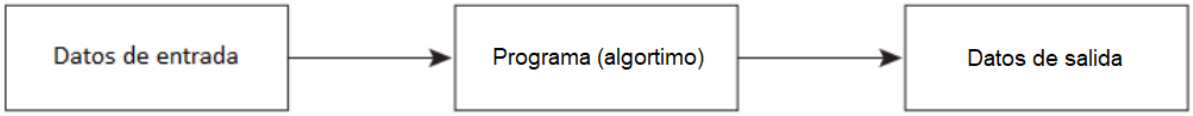
Introducción

Las pruebas de software son las investigaciones empíricas y técnicas cuyo objetivo es proporcionar información sobre la calidad o funcionamiento de un programa.

Las pruebas son básicamente un conjunto de actividades dentro del desarrollo de software. Dependiendo del tipo de pruebas, estas actividades podrán ser implementadas en cualquier momento de dicho proceso de desarrollo.

Pruebas de Caja Negra

Las pruebas de caja negra permiten la revisión final de las especificaciones y codificación de un programa. La prueba es considerada aceptable cuando su ejecución conlleva una probabilidad elevada de encontrar un error y es satisfactoria cuando se lo detecta. Esto se consigue mediante la exhaustiva elección de los datos de entrada y salida válidos y no válidos.



Este tipo de prueba detecta las siguientes tipologías de errores: Funciones incorrectas o ausentes, errores de inicialización, validaciones faltantes, mensajes con textos incorrectos, entre otros.

Veremos tres técnicas para realizar pruebas sobre nuestro programa: Particiones de equivalencia, análisis de valores límites y Tabla de decisión. Estas se pueden combinar para probar un mismo programa.

Particiones de equivalencias

En esta técnica se dividen los datos de entrada en particiones o clases de equivalencia, los datos que producen el mismo resultado son agrupados en una misma clase, es decir, se espera que todos los miembros de una partición sean procesados de la misma manera.

Algunas consideraciones a tener en cuenta con esta técnica:

1. Cada dato de entrada de debe pertenecer a una sola partición o clase de equivalencia.
2. Los valores válidos son los que se supone que el programa debe aceptar. Las particiones que contienen los valores válidos son llamadas "partición de equivalencia válida".
3. Los valores que son rechazados por un programa son llamados valores no válidos y las particiones que los contienen se conocen como "partición de equivalencia no válida".
4. Cada una de las particiones de equivalencia no válidas deben probarse de manera independiente.

Ejemplo 1: Imaginemos que tenemos un sistema que solo permite el acceso a personas de 18 a 60 años de edad (para fines de este ejemplo vamos a obviar la existencia de números negativos) y el sistema le pide al usuario que introduzca su edad, si el usuario tiene menos de 18 años o más de 60 no permite el acceso y lanza un mensaje dependiendo de la edad, a los menores de

18 les dice: "No puede acceder, su edad es menor a la requerida" y a los mayores de 60: "No puede acceder, su edad es mayor a la requerida".

```
Proceso ejemplo
    Definir edadPersona Como Entero
    Escribir "Ingrese la edad de la persona: "
    Leer edadPersona
    Si edadPersona≥18 y edadPersona≤60 Entonces
        Escribir "BIENVENIDO"
    SiNo
        Si edadPersona>60 Entonces
            Escribir "No puede acceder, su edad es menor a la requerida"
        SiNo
            Escribir "No puede acceder, su edad es mayor a la requerida"
        Fin Si
    Fin Si
FinProceso
```

Las particiones de equivalencia que podríamos armar para luego probar el programa son:

Partición no válida	**Partición Válida**	**Partición no Válida**
<=17	18 a 60	>=61

Ya que el sistema debe comportarse de la misma manera para todos los menores de 18 años, todas esas edades se agrupan en una misma partición y así igual con los otros dos grupos.

Para probar que el algoritmo funcione correctamente sólo necesitamos tres valores de entrada ***15, 20 y 62*** (uno de cada partición) y probamos todos los casos.

Sería lo mismo probar con 18 que probar con 25, o probar con 1 que probar con 17.

Análisis de valor de límite

A diferencia de la partición de equivalencia, que toma un valor dentro del rango de la partición como dato de entrada, en esta técnica obligatoriamente deben tomarse como mínimo los dos valores límite y opcionalmente, un valor entre los límites. Para utilizar esta técnica los valores deben ser numéricos y estar ordenados.

Para el ejemplo anterior deberíamos probar con los valores ***17, 18, 60, 61*** ya que son los valores límites de cada partición.

<=17		18 a 60		>=61	
Límite inferior	Límite superior	Límite inferior	Límite superior	Límite inferior	Límite superior
-	17	18	60	61	-

Esta técnica surge como respuesta a un caso muy común: es muy probable encontrar defectos en los valores límites.

Hace más fácil encontrar los valores para probar que tienen más probabilidad de tener defectos en el código: en los límites se tiende a encontrar más defectos que en el medio de un rango.

Tabla de decisión

Es una técnica utilizada para probar el algoritmo de un programa utilizando diferentes combinaciones de entrada. Este es un enfoque donde las diferentes combinaciones de entrada y los resultados se muestran en forma tabular.

Cada columna corresponde a una prueba la cúal es definida por las combinaciones de entrada.

Los valores a colocarse son generalmente booleanos como V (verdadero) y F (Falso). Si hay un valor que no importa para obtener cierto resultado se puede marcar con un guion "-".

Ejemplo 2: Para ingresar al sistema de facturación de una empresa se dispone de una credencial (usuario y contraseña) definida previamente. Diseñe un algoritmo que solicite el usuario y contraseña para ingresar e informe con mensajes aclaratorios al usuario si se ha ingresado las credenciales correctas o no.

```
Proceso ejemplo
    Definir userDefinido, userIngresado Como Texto
    Definir passwordDefinido, passwordIngresado Como Texto
    userDefinido ← "admin"
    passwordDefinido ← "hola123"

    Escribir "Ingrese el usuario"
    Leer userIngresado
    Escribir "Ingrese las password"
    Leer passwordIngresado

    Si (userDefinido = userIngresado) y (passwordDefinido = passwordIngresado) Entonces
        Escribir "Bienvenido"
    SiNo
        Escribir "Error"
    Fin Si
FinProceso
```

Las condiciones son simples, si se proporciona un usuario y una contraseña válida, el usuario puede iniciar sesión exitosamente, de lo contrario aparecerá un mensaje de error.
Armando la siguiente tabla podemos probar las 4 combinaciones diferentes del algoritmo antes descrito.

	Prueba 1	Prueba 2	Prueba 3	Prueba 4
Usuario Válido (Verdadero/Falso)	Falso	Verdadero	Falso	Verdadero
Contraseña Válida (Verdadero/Falso)	Falso	Falso	Verdadero	Verdadero
Resultado	Mensaje Error	Mensaje Error	Mensaje Error	Inicio exitoso

Bibliografía

Programa de Estudio de ISTQB® Probador Certificado de Nivel Básico Versión 2018 - Español
https://www.diariodeqa.com/post/tecnicas-de-prueba-de-caja-negra

Versiones

Fecha	Versión
17/04/2023	1.0

Autores

María Mercedes Valoni