Pruebas que permitan comprobar que el producto que se está creando, es correcto y cumple con las especificaciones solicitadas por el usuario.

La verificación es la comprobación de que un sistema o parte de un sistema, cumple con las condiciones impuestas. Con la verificación se comprueba si la aplicación se está construyendo correctamente.

La validación es el proceso de evaluación del sistema o de uno de sus componentes, para determinar si satisface los requisitos especificados.

Las pruebas empezarán con la pruebas unitarias de cada porción de código seguirá con las pruebas de integración, donde se ponen todas las partes del código en común,

la prueba de validación =>cumple requisitos de software.

Finalmente se alcanza la prueba de sistema que verifica el funcionamiento total del software y otros elementos del sistema.

Pruebas de la caja negra o funcionales.

-Aquí lo fundamental es comprobar que los resultados de la ejecución de la aplicación, son los esperados, en función de las entradas que recibe sin tener necesidad de conocer la estructura interna del sistema.

Pruebas de la caja blanca o estructurales.

-Analizar y probar directamente el código de la aplicación, intentando localizar estructuras incorrectas o ineficientes en el código.

Enfoque aleatorio.

-Utilizar modelos que representen las posibles entradas al programa, para crear a partir de ellos los casos de prueba

PRUEBAS DE REGRESION

Se deben llevar a cabo cada vez que se hace un cambio en el sistema, para comprobar que los cambios efectuados no han alterado el resto del codigo

DOCUMENTACION DE LA PRUEBA

-Plan de Pruebas: forma parte del Análisis del Sistema.

-Especificación del diseño de pruebas: Determina patuas de pruebas para los bloques en los que ha sido dividido el programa en la fase de diseño.

-Especificación de un caso de prueba: Fase de implementación.

-Especificación de procedimiento de prueba. será preciso detallar el modo en que van a ser ejecutados los casos prueba. Incluye los resultados esperados en cada caso de prueba y las pruebas unitarias a realizar.

-Registro de pruebas.

-Informe de incidente de pruebas.

VALIDACION

Muy importante el cliente, que es quien nos tiene que dar el ok a nuestra app.

CAJA BLANCA

Las **pruebas estructurales** son el conjunto de **pruebas de la caja blanca**.

Su función es comprobar que se van a ejecutar todas la instrucciones del programa, que no hay código no usado, comprobar que los caminos lógicos del programa se van a recorrer, etc.

**Los criterios de cobertura son**:

* **Cobertura de sentencias**: se han de generar casos de pruebas suficientes para que cada instrucción del programa sea ejecutada al menos una vez.
* **Cobertura de decisiones**: se trata de crear los suficientes casos de prueba para que cada opción resultado de una comprobación lógica del programa, se evalúe al menos una vez a cierto y otra a falso. En la decisión **MIENTRAS (A and B)**, habrá casos de prueba donde (A and B) sea verdadero y donde (A and B) sea falso.
* **Cobertura de condiciones**: cada elemento de una condición, se evalúe al menos una vez a falso y otra a verdadero. En la decisión **MIENTRAS (A and B)**, habrá casos de prueba donde A sea falso, A sea verdadero, B sea falso y B sea verdadero.
* **Cobertura de condiciones y decisiones**: consiste en cumplir simultáneamente las dos anteriores.
* **Cobertura del camino de prueba**: se pueden realizar dos variantes, una indica que cada bucle se debe ejecutar sólo una vez, ya que hacerlo más veces no aumenta la efectividad de la prueba y otra que recomienda que se pruebe cada bucle tres veces: la primera sin entrar en su interior, otra ejecutándolo una vez y otra más ejecutándolo al menos dos veces.

2. Complejidad de McCabe o ciclomática.

A partir del grafo se determina su **complejidad ciclomática**. Es posible hacerlo por tres métodos diferentes, pero todos ellos han de dar el mismo resultado.

* **V (G) = a - n + 2**, siendo a el número de arcos o aristas del grafo y n el número de nodos.
* **V (G) = r**, siendo r el número de regiones cerradas del grafo (incluida la externa).
* **V(G) = c + 1**, siendo c el número de nodos de condición

### Caminos de prueba.

El número de **caminos de prueba** debe ser igual a la complejidad calculada.

CAJA NEGRA

## Caso práctico de caja negra.

La técnica para determinar los casos de prueba de caja negra se realiza completando los siguientes pasos:

* Determinar **las clases de equivalencia**.
* Determinar un **análisis de valores límite**.
* **Conjetura de errores**.
* Generar los **caso de uso** necesarios para probar las clases válidas y no válidas. Establecer los datos de entrada y los resultados esperados.
* Lanzar una **ejecución del programa** por cada caso de uso y **comparar los resultados obtenidos con los esperados** para determinar la corrección del código.

**CLASES DE EQUIVALENCIA**

La técnica de **clases de equivalencia** es un tipo de prueba funcional, donde en cada caso de prueba se agrupa el mayor número de entradas posibles

Los **pasos**a seguir para identificar las clases de equivalencia son:

* **Identificar las condiciones de las entradas del programa**, es decir, restricciones de formato o contenido de los datos de entrada.
* A partir de ellas, **identificar clases de equivalencia** que pueden ser:
  + De datos válidos.
  + De datos no válidos o erróneos.

**Rango de valores** de entrada. Crear una clase válida y dos clases no válidas. La edad de acceso a un evento está comprendida entre 18 y 100 años

**Número finito y consecutivo de valores**. Creará una clase válida y dos no válidas. (IGUAL QUE EL RANGO… diferencias???)

Condición **verdadero/falso**.

**Conjunto de valores admitidos**. Se identifica una clase válida por cada valor y una no válida. Una opción de menú puede aceptar los valores 'A' para altas, 'B' para bajas y 'S' para salir del programa. (cualquier otra opcion introducida sería no válida)

**El AVL (Análisis de valores límite)**

Ahora la prueba se realizará sobre los valores límite de los rangos.