# 1.- Prueba del software

La prueba del software es un elemento crítico para la garantía de calidad del software y representa una revisión final de las especificaciones, del diseño y de la codificación.

## 1.1.- Fundamentos de la prueba del software

### Objetivos de la prueba

Reglas que sirven como objetivos de prueba:

1. La prueba es un proceso de ejecución de un programa con la intención de descubrir un error.
2. Un buen caso de prueba es aquel que tiene una alta probabilidad de mostrar un error no descubierto hasta entonces.
3. Una prueba tiene éxito si descubre un error no detectado hasta entonces.

El objetivo es diseñar pruebas que sistemáticamente saquen a luz diferentes clases de errores, haciéndolos con la menor cantidad de tiempo y esfuerzo.
Si la prueba se lleva a cabo con éxito descubrirá errores en el software. Como ventaja secundaria, la prueba demuestra hasta que punto las funciones del software parecen funcionar de acuerdo con las especificaciones. Además, proporciona una buena indicación de fiabilidad del software. Sin embargo, la prueba no puede asegurar la ausencia de defectos; sólo puede demostrar que existen defectos en el software.

### Diseño de casos de prueba

Los métodos de diseño de casos de prueba para el software proporcionan un enfoque sistemático a la prueba. Más importante aún; los métodos, proporcionan un mecanismo de ayuda para asegurar la completitud de las pruebas y para conseguir la mayor probabilidad de descubrimiento de errores en el software.
Cualquier producto de software puede ser probado de una de dos formas:

1. Conociendo la función específica para la que fue diseñado el producto, se pueden llevar a cabo pruebas que demuestren que cada función es completamente operativa.
2. Conociendo el funcionamiento del producto se pueden desarrollar pruebas que aseguren que la operación interna se ajusta a las especificaciones y que todos los componentes internos se han comprobado de forma adecuada.

El primer enfoque de prueba se denomina *prueba de la caja negra* y la segunda *prueba de la caja blanca.*

## 1.2.- Prueba de la caja blanca

La prueba de caja blanca también conocida como *prueba de caja de cristal*, es un método de diseño de casos de prueba que usa la estructura de control de diseño procedimental para derivar los casos de prueba. Mediante los métodos de prueba de la caja blanca, el ingeniero del software puede obtener casos de prueba que garanticen que:

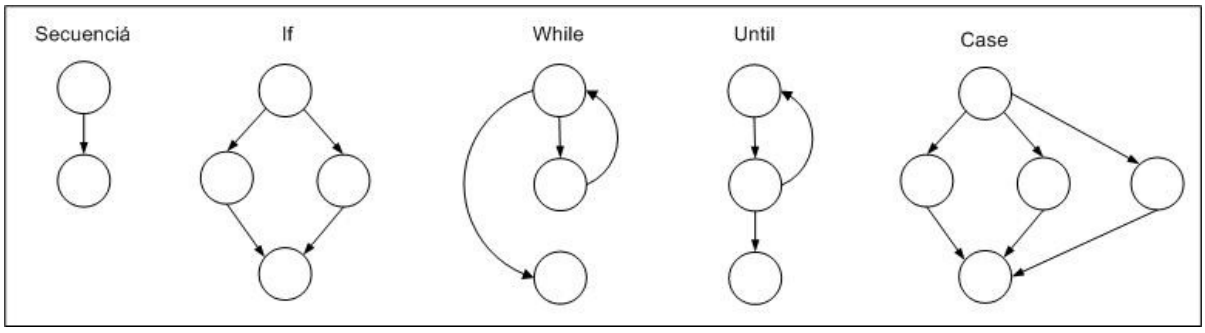
1. Ejecuten por lo menos una vez todos los caminos independientes de cada modulo
2. Ejecuten todas las decisiones lógicas por verdadero o falso
3. Ejecuten todas los bucles en sus límites y con sus límites operacionales
4. Ejerciten las estructuras internas de datos para asegurar su validez.

### Prueba del camino básico

La *prueba del camino básico* es una técnica de prueba de la caja blanca. El método del camino básico permite al diseñador de casos de prueba obtener una medida de la complejidad lógica de un diseño procedimental y usar esa medida como guía para la definición de un *conjunto básico* 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.

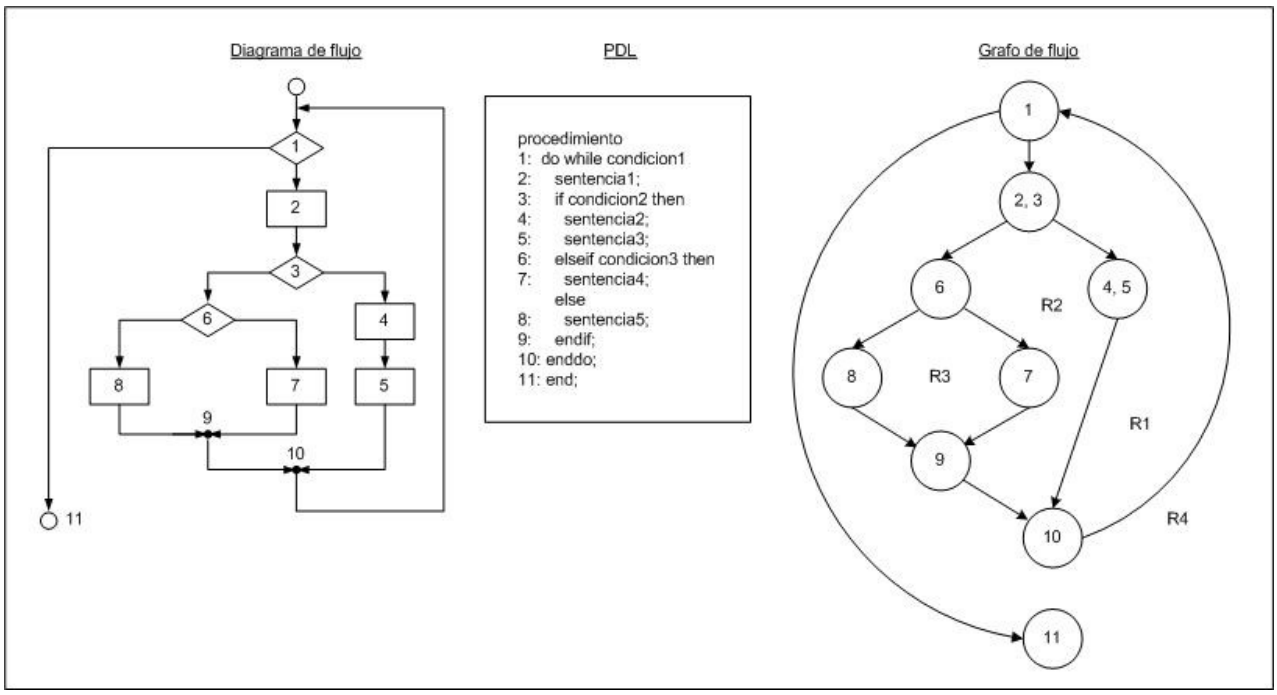
### Notación de grafo de flujo

El *grafo de flujo* representa el flujo de control lógico mediante la notación:




Cada círculo, denominado *nodo* del grafo de flujo, representa una o más sentencias procedimentales. Las flechas del grafo de flujo, denominadas *aristas,* representan el flujo de control. Una arista debe terminar en un nodo, incluso aunque el nodo no represente ninguna sentencia procedimental. Las áreas

delimitadas por aristas y nodos se denominan *regiones*. Cuando contabilizamos las regiones incluimos el área exterior del grafo, contándola como otra región más.
Cualquier representación del diseño procedimental se puede traducir a un grafo de flujo. En la siguiente figura se muestra la traducción de un diagrama de flujo y de un PDL a su correspondiente *grafo de flujo*




**Camino independiente:** Introduce un nuevo conjunto de sentencias o una condición

Ejemplo:

- Camino 1: 1-11
- Camino 2: 1-2-3-4-5-10-1-11
- Camino 3: 1-2-3-6-8-9-10-1-11
- Camino 4: 1-2-3-6-7-9-10-1-11

El conjunto básico de caminos no es único

## 1.3.- Pruebas de la caja negra

*Las pruebas de caja negra,* también denominada *prueba de comportamiento,* se centran en los requisitos funcionales del software. O sea, la prueba de caja negra permite al ingeniero del software obtener conjuntos de condiciones de entrada

que ejerciten completamente todos los requisitos funcionales de un programa. La prueba de caja negra no es una alternativa a las técnicas de prueba de caja blanca. Más bien se trata de un enfoque complementario que intenta descubrir diferentes tipos de errores que los métodos de caja blanca.
La prueba de caja negra ignora intencionadamente la estructura de control, centra su atención en el campo de la información. Esta prueba intenta encontrar errores de las siguientes categorías:

1. Funciones incorrectas o ausentes
2. Errores de interfaz
3. Errores de estructuras de datos o en acceso a bases de datos externas
4. Errores de rendimiento
5. Errores de inicialización y de terminación.

Ya que la prueba de la caja negra ignora intencionalmente la estructura de control, centra su atención en el campo de información.

## 1.4.- Estrategias de prueba del software

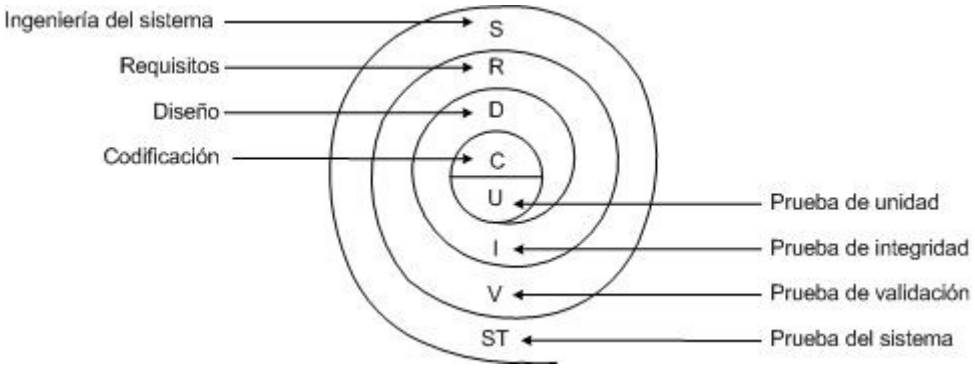
Cualquier estrategia de prueba debe incorporar la planificación de la prueba, el diseño de casos de prueba, la ejecución de pruebas y la agrupación y evaluación de los datos resultantes.
Una estrategia para la prueba del software debe acomodar pruebas de bajo nivel que verifiquen que cada pequeño segmento de código fuente se ha implementado correctamente, así como pruebas de alto nivel que muestren la validez de las principales funciones del sistema frente a los requisitos del cliente.

### Verificación y validación

La prueba de software es un elemento de un concepto más amplio que, a menudo, se referencia como *verificación y validación.* La verificación se refiere al conjunto de actividades que aseguran que el software implementa correctamente una función específica. La validación se refiere a un conjunto diferente de actividades que aseguran que el software construido se ajusta a los requisitos del cliente.
*Verificación:* ¿estamos construyendo el producto correctamente?
*Validación:* ¿estamos construyendo el producto correcto?

### Una estrategia de prueba del software

El proceso de ingeniería del software se puede ver como un espiral.




Inicialmente, la ingeniería del sistema define el papel del software y conduce al análisis de los requerimientos del software, donde se establece el campo de información, la función, el comportamiento, el rendimiento, las restricciones y los criterios de validación del software. Al movernos hacia el interior del espiral, llegamos al diseño y, por último a la codificación.
Para la prueba del software nos movemos hacia afuera del espiral. La *prueba de unidad* comienza en el vértice del espiral y se centra en cada unidad del software. La prueba avanza al movernos hacia afuera del espiral, hasta llegar a la *prueba de integración*, donde el foco de atención es el diseño y la construcción de la arquitectura del software. Dando otra vuelta por el espiral hacia afuera, encontramos la *prueba de validación*, donde se validan los requisitos establecidos como parte del análisis de requisitos del software, comparándolos con el sistema que ha sido construido. Finalmente, llegamos a la prueba del sistema, en la que se prueban como un todo el software y otros elementos del sistema.

## Prueba de unidad

La prueba de unidad centra el proceso de verificación en la menor unidad del diseño del software (componente o modulo). La prueba de unidad siempre está orientada a la caja blanca y este paso se puede llevar a cabo en paralelo para múltiples módulos.
Se aplican sobre funciones o grupos de métodos asociados a módulos u objetos.

- Probar interfaz de la componente
- Probar estructuras de datos locales
- Probar impacto de los datos globales
- Probar condiciones límites
- Probar caminos independientes
- Probar caminos de manejo de errores

## Prueba de integración

La prueba de integración es una técnica sistemática para construir la estructura del programa mientras que, al mismo tiempo, se llevan a cabo pruebas para detectar errores asociados con la interacción.
En una *integración no incremental:* o "big bang", se combinan todos los módulos por anticipado. Se prueba todo el programa en conjunto. Normalmente se llegan al caos. Se encuentra un gran conjunto de errores. La corrección se hace difícil.
La *integración incremental:* el programa se construye y se prueba en pequeños segmentos en los que los errores son más fáciles de aislar y de corregir, es más probable que se puedan probar completamente las interfaces y se pueda aplicar un enfoque de prueba sistemática.

## *Prueba de validación*

La validación se logra cuando el software funciona de acuerdo con las expectativas razonables del cliente.

### Criterios para la prueba de validación

La validación del software se consigue mediante una serie de pruebas de la caja negra que demuestran la conformidad con los requisitos. Tanto el plan como el procedimiento estarán diseñados para asegurar que se satisfacen todos los requisitos funcionales, que se alcanzan todos los requisitos de rendimiento, que la documentación es correcta e inteligible y que se alcanzan otros requisitos (por ejemplo: portabilidad, compatibilidad, recuperación de errores, posibilidad de mantenimiento).
Una vez que se procede con cada caso de prueba de validación, puede darse una de dos condiciones: (1) las características de funcionamiento o de rendimiento están de acuerdo con las especificaciones y son aceptables o (2) se descubre una desviación de las especificaciones y se crea una lista de deficiencias.

### Pruebas alfa y beta

La mayoría de los constructores de productos de software llevan a cabo un proceso denominado *prueba alfa y beta* para descubrir errores que parezcan que solo el usuario final puede descubrir.
La *prueba alfa* es conducida por un cliente en el lugar de desarrollo. Se usa el software de forma natural, con el encargado registrando erres y problemas de uso. Las pruebas alfa se llevan a cabo en un entorno controlado.

La *prueba beta* se lleva a cabo en uno o más lugares de clientes por los usuarios finales del software. A diferencia de la prueba alfa, el encargado del desarrollo, normalmente, no está presente. El cliente registra todos los problemas que encuentra durante la prueba beta e informa a intervalos regulares al equipo de desarrollo.

## Prueba del sistema

El software es incorporado a otros elementos del sistema y se realiza una serie de pruebas de integración del sistema y de validación. Estas pruebas caen fuera del alcance del proceso de ingeniería del software y no son conducidas únicamente por el encargado del desarrollo del software.
El ingeniero del software debe anticiparse a los posibles problemas de interacción y: (1) diseñar caminos de manejo de errores que prueben toda la información procedente de otros elementos del sistema; (2) llevar a cabo una serie de pruebas que simulen la presencia de datos en mal estado o de otros posibles errores en la interfaz del software; (3) registrar los resultados de las pruebas como evidencia; (4) participar en la planificación y el diseño de pruebas del sistema para asegurarse de que el software es probado en forma adecuada.
Tipos de prueba del sistema:

### Prueba de recuperación

La *prueba de recuperación* es una prueba del sistema que fuerza el fallo del software de muchas formas y verifica que la recuperación se lleva a cabo apropiadamente.

### Prueba de seguridad

La *prueba de seguridad* intenta verificar que los mecanismos de protección incorporados al sistema lo protegerán.
Durante la prueba de seguridad, el encargado de la prueba desempeña el papel de un individuo que desea penetrar en el sistema.
Con el suficiente tiempo y los suficientes recursos, una buena prueba de seguridad terminará por penetrar en el sistema. El papel del diseñador del sistema es hacer que el costo de penetración sea mayor que el valor de la información obtenida mediante la penetración.

### Prueba de resistencia (stress)

Las *pruebas de resistencia* están diseñadas para enfrentar a los programas con situaciones anormales.
La prueba de resistencia ejecuta un sistema de forma que demande recursos en cantidad, frecuencia o volúmenes anormales.

### Prueba de rendimiento

La *prueba de rendimiento* está diseñada para probar el rendimiento del software en tiempo de ejecución dentro del contexto de un sistema integrado. La prueba de rendimiento se da durante todos los pasos del proceso de prueba. Incluso al nivel de unidad, se debe asegurar el rendimiento de los módulos individuales a medida que se llevan a cabo las pruebas de caja blanca. Sin embargo, hasta que no están completamente integrados todos los elementos del sistema no se puede asegurar realmente el rendimiento del sistema.

Documentación de la prueba:

## Depuración

La *depuración* aparece como una consecuencia de una prueba efectiva. O sea, cuando un caso de prueba descubre un error, la depuración es el proceso que resulta de la eliminación del error.

### El proceso de depuración

La depuración no es una tarea de prueba aunque siempre se da como una consecuencia de la prueba. El proceso de depuración comienza con la ejecución de un caso de prueba. Se evalúan los resultados y aparece una falta de correspondencia entre los esperados y los reales. El proceso de depuración intenta hacer corresponder el sistema con una causa, llevando así a la corrección del error.
El proceso de depuración siempre tiene uno de dos resultados:

1. Se encuentra la causa, se corrige y se elimina
2. No se encuentra la causa. En este último caso, la persona que realiza la depuración debe sospechar de la causa, diseñar un caso de prueba que ayude a validar sus sospechas y el trabajo vuelve hacia atrás a la corrección del error de una forma iterativa.