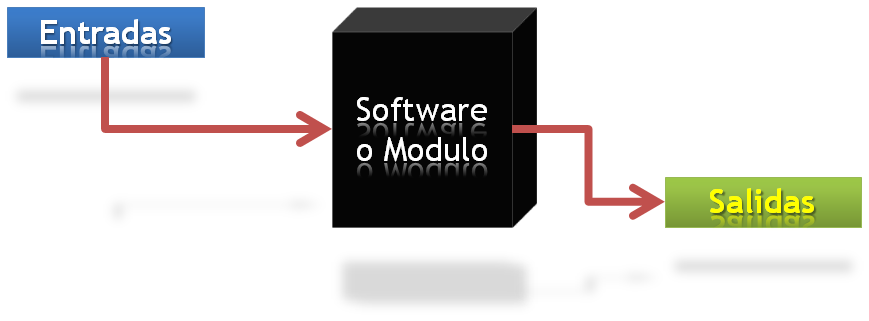
**Pruebas de caja negra**

Las **Pruebas de Caja Negra**, es una técnica de pruebas de software en la cual la funcionalidad se verifica sin tomar en cuenta la estructura interna de código, detalles de implementación o escenarios de ejecución internos en el software.

En las pruebas de caja negra, nos enfocamos solamente en las entradas y salidas del sistema, sin preocuparnos en tener conocimiento de la estructura interna del programa de software. Para obtener el detalle de cuáles deben ser esas entradas y salidas, nos basamos en los requerimientos de software y especificaciones funcionales.

* Se centran principalmente en lo que “se quiere” de un módulo o sección específica de un software.
* Las pruebas de caja negra son, ni más ni menos que, pruebas funcionales dedicadas a “mirar” en el exterior de lo que se prueba.
* Se denominan de varias formas, pruebas de caja “opaca”, pruebas de entrada/salida, pruebas inducidas por datos…los sinónimos son muchos y muy variados.



**Método de clases de equivalencia:**

Este método consiste en realizar agrupaciones de clases por comportamientos, se agrupan entonces los datos de entrada y de salida en clases diferentes en la que todos los componentes de cada clase están relacionados por su comportamiento.

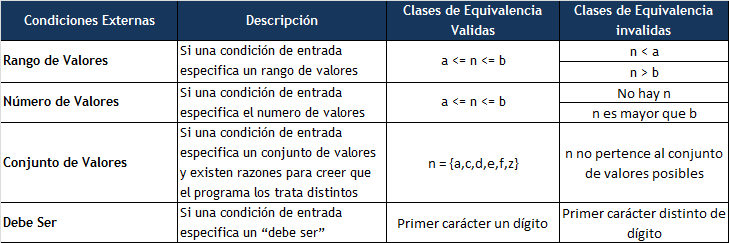
Cada una de estas clases será llamada una partición de equivalencia en la que el programa posee el mismo comportamiento indiferente de cual sea la entrada.

Se debe seleccionar un elemento representativo de la clase, donde se supone que si la agrupación es correcta, el sistema tendrá el mismo comportamiento para cualquier otro valor.

Para la aplicación de este método debemos de seguir 2 pasos:

**1. Identificación de las clases de equivalencia**

Identificamos las clases según el posible comportamiento de los valores

[](http://3.bp.blogspot.com/-sx3wEnRgpe8/UgfE41WQ5eI/AAAAAAAAAKY/XGx6l9w5mUw/s1600/TablaClases.png)

Si se considera o existe una razón para detectar que los valores de una clase no son idénticos,  se debe dividir en clases menores.

### 2. Identificación de los casos prueba.

Por cada dato de entrada y salida elegimos un valor perteneciente a una clase de equivalencia.  
Un caso de prueba está formado por un dato de prueba para cada dato de entrada y salida (resultado esperado)  
Basados en esto, estructuramos nuestros casos prueba, bajo las siguientes pautas:

* El objetivo es probar todas las clases válidas y no válidas al menos una vez cada una de ellas
* Cada caso debe cubrir el máximo numero de entradas
* En un caso de prueba pueden aparecer varias clases de entrada válidas
* En un caso de prueba solamente puede aparecer una entrada no válida. El criterio para las clases no válidas impide que los errores se enmascaren mutuamente
* Se debe especificar el resultado esperado

Con estas pautas seguimos los siguientes pasos:

* Asignar un número único a cada clase de equivalencia
* Escribir casos de prueba hasta que sean cubiertas todas las clases de equivalencias validas, intentando cubrir en cada caso tantas como sea posible
* Para cada Clase de Equivalencia Invalida, escribir un caso de prueba, cubriendo en cada caso tantas como sea posible.

Ejemplo: **result function (x, y, z)**. Sean Vi las clases válidas y Ni las clases inválidas

Clases de x: V1, N1,N2

Clases de y: V2,N3

Clases de z: V3, N4, N5

Clases de result: V4, N6, N7

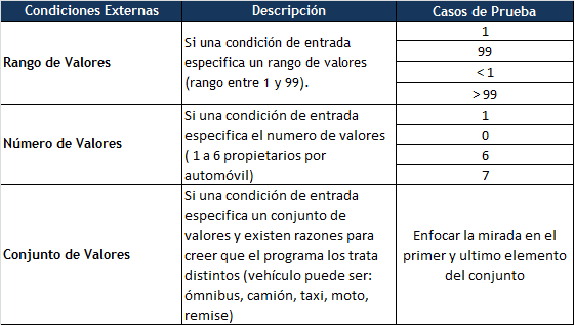
Un posible caso de prueba sería: ( x1 en V1, y1 en V2, z1 en V3, r1 en V4)

#### Método de Análisis de valores frontera o valores límite:

Este método parte de del método de clases de equivalencia, su diferencia radica en el modo de generar los casos prueba, en este caso más que seleccionar cualquier elemento como representativo de la clase de equivalencia, el análisis de valor límite requiere que se seleccionen uno o más elementos de forma tal que cada margen de la clase de equivalencia sea sujeto  a una prueba.

Los casos de prueba se derivan también de considerando el espacio de resultados (clases de equivalencia de salida).

La diferencia entre esta técnica y la de las particiones de equivalencia es que se exploran en estas situaciones existentes sobre y alrededor de los bordes de las particiones de equivalencia.

[](http://1.bp.blogspot.com/-s0_roRBoukI/UgfWaFJx0WI/AAAAAAAAAKo/_86vw6bAp2o/s1600/TablaLimite.png)

Con estas premisas, entonces estructurar los casos de salida de la siguiente forma:

1. Usar el rango de valores para los valores de salida, o sea su calculo debe dar entre 1 y 100. escribir casos de prueba que causen salidas de 1 y 100.

2. Usar el numero de valores para la salida.

3. Analizar las posibles situaciones para encontrar otros limites.

#### 

#### Método Adivinación de errores:

#### Este método de testing se basa en la experiencia del ejecutor. Consiste en realizar un análisis de los requisitos y el diseño del sistema, buscando intuir en que puntos del sistema se pudieron insertar errores y elaborar el diseño de las pruebas enfocado a atacar el sistema sobre estos casos.

Es curioso, pero en la gran mayoría de las ocasiones este método es utilizado por los tester,  no existe mucha documentación del método ni una estructuración de los pasos a seguir para este tipo de prueba.  
  
Las historias de defectos pueden ser útiles; hay una alta probabilidad de que los errores que han aparecido en el pasado, puedan volver a aparecer. Generalmente se enfocan en:

* listas o strings vacíos o nulos
* cero ocurrencias o instancias
* números negativos

Métodos de Caja Negra, (2010), [www.fing.edu.uy](http://www.fing.edu.uy/), Obtenida desde: <http://www.fing.edu.uy/~bperez/protest/guias/guias/tecnicas/cubrimiento.htm>

Técnicas de testing caja negra, (2009), indalog.ual.es, Obtenida desde: <http://indalog.ual.es/mtorres/LP/Prueba.pdf>

Black box testing, (2009), agile.csc.ncsu.edu, Obtenida desde: <http://agile.csc.ncsu.edu/SEMaterials/BlackBox.pdf>

Pruebas de caja negra: Ejemplos, (2017), [www.pmoinformatica.com](http://www.pmoinformatica.com) ,Obtenida desde:

<http://www.pmoinformatica.com/2017/02/pruebas-de-caja-negra-ejemplos.html>