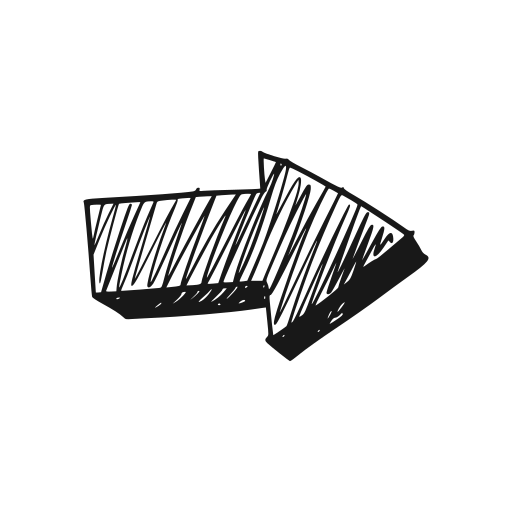
**Clase 02. ¿QUÉ SIGNIFICA SER UN TESTER?**

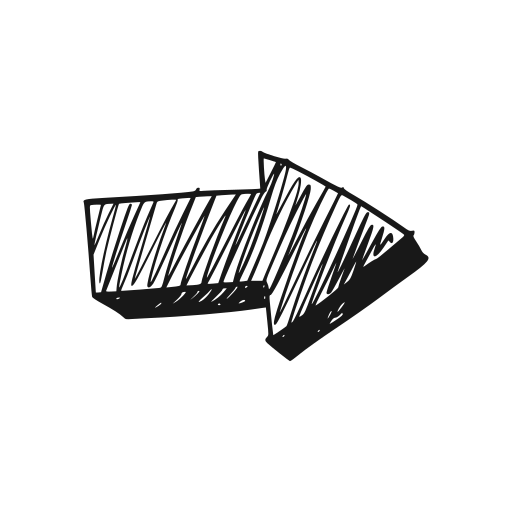
# Principios del testing

## Las máximas del Testing

Hace muchos años se han establecido 7 verdades absolutas dentro del universo del testing, que siempre se regirán, sin importar el proyecto en el que estés.

Podés encontrar las 7 máximas del testing en la siguiente presentación de genially.

Haz click en el siguiente [link](https://view.genial.ly/62067e49f20080001ac65e32/presentation-principios-del-testing-qa)

Estas máximas son:

1. Las pruebas muestran la presencia de defectos, no su ausencia.
2. Las pruebas exhaustivas son imposibles.
3. Las pruebas tempranas ahorran tiempo y dinero.
4. Los defectos se agrupan.
5. Cuidado con la paradoja del pesticida.
6. Las pruebas dependen del contexto.
7. La ausencia de errores es una falacia.

La prueba muestra la presencia de defectos, no su ausencia

La prueba puede mostrar la presencia de defectos, pero no puede probar que no hay defectos.

La prueba reduce la probabilidad de que queden defectos no descubiertos en el software pero, incluso si no se encuentran defectos, el proceso de prueba no es una demostración de la corrección.

Ejemplo Primer Principio:



El Samsung Galaxy Note 7 tenía una falla y hacía que se prendiera fuego. Tuvo que ser retirado del mercado.

Siempre puede haber una falla.

La prueba exhaustiva es imposible

No es posible probar todo, excepto en casos triviales. En lugar de intentar realizar pruebas exhaustivas se deberían utilizar el análisis de riesgos, las técnicas de prueba y las prioridades para centrar los esfuerzos de prueba.

Ejemplo Segundo Principio



Aplicación de temperatura global. Deberían probar los 195 países, por cada provincia y por cada ciudad… ¿Es un poco alta esa cantidad de casuísticas no?

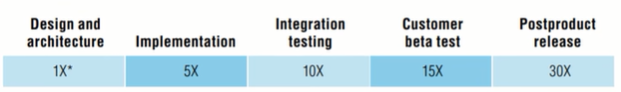
La prueba temprana ahorra tiempo y dinero

Para detectar defectos de forma temprana, las actividades de prueba, tanto estáticas como dinámicas, deben iniciarse lo antes posible en el ciclo de vida de desarrollo de software.

La prueba temprana en el ciclo de vida de desarrollo de software ayuda a reducir o eliminar cambios costosos.

Ejemplo Tercer Principio

Según un estudio realizado por IBM, el costo de reparación de un defecto en su etapa de post-producción es de hasta 30 veces más costoso, que en la etapa de diseño, y 3 veces más que en la etapa de testing.



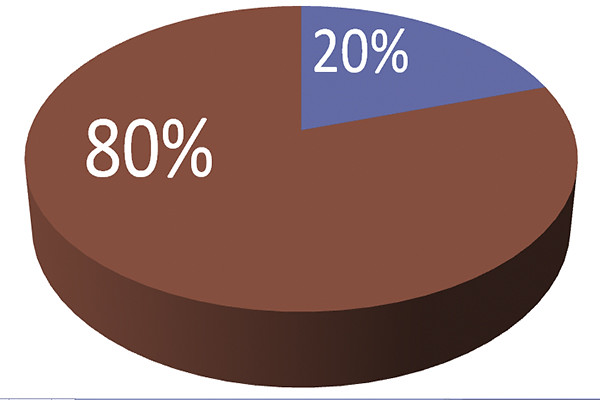
Fuente:[Link](https://docplayer.net/11413245-Minimizing-code-defects-to-improve-software-quality-and-lower-development-costs.html)

Los defectos se agrupan

En general, un pequeño número de módulos contiene la mayoría de los defectos descubiertos durante la prueba previa al lanzamiento, o es responsable de la mayoría de los fallos operativos.

Las agrupaciones de defectos observadas en la prueba o producción son una aportación importante a un análisis de riesgos utilizado para centrar el esfuerzo de la prueba.

Ejemplo Cuarto Principio

El 80% de los defectos se encuentra en el 20% de los módulos, ya que son el puntapié para desencadenar otros defectos en el resto del producto.

Cuidado con la paradoja del pesticida

Si las mismas pruebas se repiten una y otra vez, eventualmente estas pruebas ya no encontrarán ningún defecto nuevo.

Para detectar nuevos defectos, es posible que sea necesario cambiar las pruebas y los datos de prueba existentes, y es posible que sea necesario redactar nuevas pruebas.

Las pruebas ya no son efectivas para detectar defectos, de la misma manera que los pesticidas ya no son efectivos para matar insectos después de un tiempo

Ejemplo Quinto Principio

Si utilizamos los mismos métodos una y otra vez, estos dejan de surtir efecto, de la misma forma que el pesticida deja de surtir efectos en las plagas.

Las pruebas depende del contexto

Las pruebas se realizan de distintas maneras en diferentes contextos. Comprendiendo que el contexto va a indicar herramientas que pueden utilizarse para probar, ambiente o tipo de objeto que se prueba, metodología implementada por la empresa, o cualquier otro contexto que pueda existir.

Ejemplo Sexto Principio

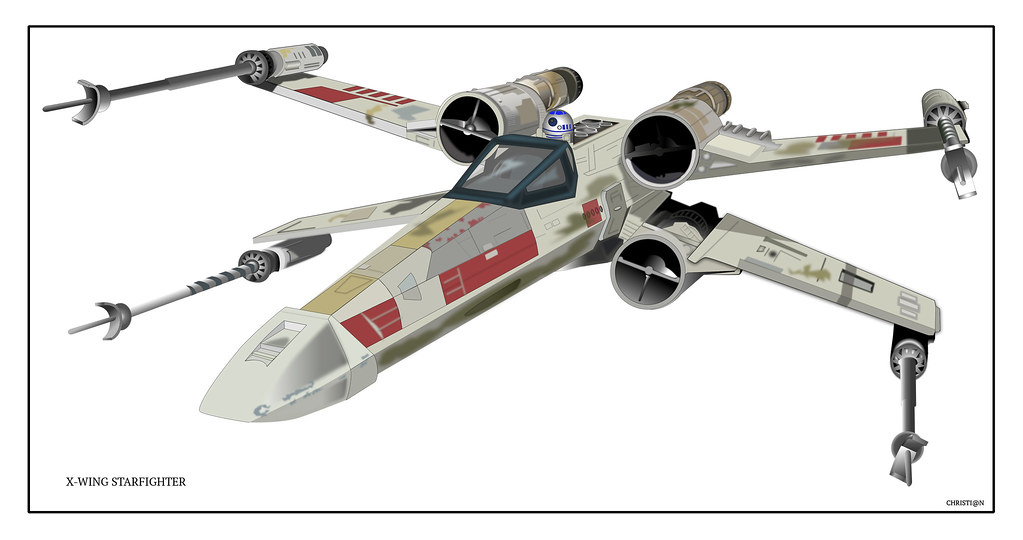


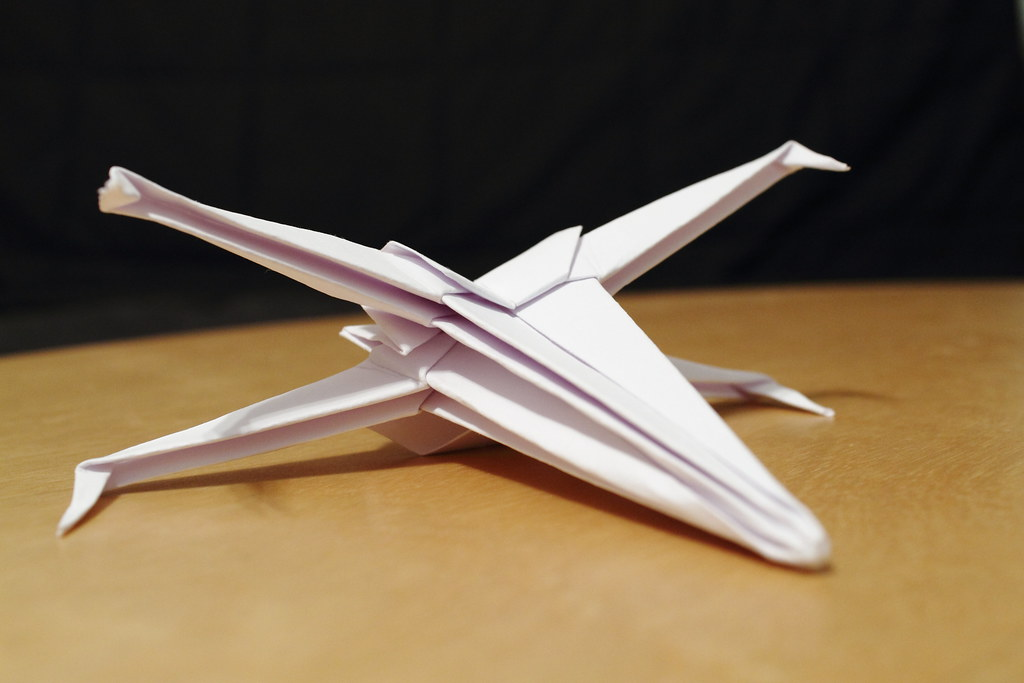
Tenemos 3 tostadoras en la imagen. ¿Creen que van a tener que hacer las mismas pruebas para las tres? ¿O cada tostadora tendrá su propio set de pruebas distinta a las otras?

La ausencia de errores es una falacia

Algunas organizaciones esperan que los tester puedan realizar todas las pruebas posibles y encontrar todos los defectos posibles, pero los principios 2 y 1, respectivamente, nos dicen que esto es imposible. Además, es una creencia equivocada esperar que sólo con encontrar y corregir un gran número de defectos se asegure el éxito de un sistema.

Ejemplo Séptimo Principio

Sabemos que el cliente espera entregar un x-wing al “no encontrar errores” pero…

...nos encontramos con uno de papel al dejar de lado la usabilidad, no cumplir con las necesidades y es de menor calidad que la competencia.

# ¿Cuáles son las responsabilidades de un tester manual?

### Responsabilidad en Calidad

En la clase pasada se explicó que significa la calidad, y quizá hacemos mucho hincapié en ello, pero es el punto más importante del QA, y por ello hacemos todas las pruebas necesarias para alcanzar el máximo nivel de calidad.

Nuestra responsabilidad como aseguradores de la calidad, es que el producto sea lo más confiable para el cliente.

### Máximo nivel de Calidad

Al encontrar errores y reportarlos al área de Desarrollo, estos volverán corregidos por lo que se probará nuevamente y en cada ciclo de testing el producto va a ir mejorando y tendrá mayor calidad.

No es nada malo que existan errores en el software, siempre y cuando el tester los encuentre.

### Objetivos de Probar

* Evaluar condiciones del trabajo a realizar como diseño, requisitos, código y/o historias de usuario.
* Verificar que se cumplen los requisitos funcionales.
* Validar si lo que se prueba funciona como debe, y cómo los usuarios y otros implicados esperan.
* Reducir la posibilidad de funcionamientos inadecuados del software.
* Generar confiabilidad en el nivel de calidad del programa.
* Prevenir defectos.
* Encontrar fallas y defectos a su previa implementación.
* Entregar suficiente información a todos los implicados en la toma de decisiones.
* Cumplir con requisitos contractuales, legales, reglamentarias, y/o verificar el cumplimiento de dichos requisitos por parte del sistema.

### Validar vs Verificar

#### Validar

El software debería ajustarse a lo que el cliente realmente pidió

Responde a la pregunta: ¿estamos construyendo el sistema correcto?

#### Verificar

El software debería ajustarse a su especificación.

Responde a la pregunta: ¿estamos construyendo el sistema correctamente?

### ¿Cómo probamos?

La actividad de realizar pruebas se divide en dos etapas fundamentales. Las pruebas dinámicas y las pruebas estáticas.

Las pruebas dinámicas es la propia acción de probar un sistema.

Las pruebas estáticas son la acción de leer documentación.

\*Nota: En las siguientes clases se verá con detalle las pruebas dinámicas y estáticas

### ¿Qué es la Documentación?

Es toda la información que nos va a respaldar y es muy importante que exista en un Proyecto.

Existen muchos tipos de documentación, aca detallaremos los más importantes:

* Historias de usuario.
* Plan de pruebas.
* Manuales.
* Criterios de aceptación.

\*Nota: En las siguientes clases se verá con detalle los tipos de documentación

#### Importancia de la Documentación

El trabajo del tester es bastante invisible, ya que no se puede palpar en sí mismo, para eso entra en juego la documentación.

Es nuestro respaldo como profesionales para indicar que pudimos corroborar, que cosas se probaron, en base a que estándares, quien los pidió, y cualquier otro tipo de afirmación escrita.

#### Herramientas para documentar

Casos de prueba:

* Excel
* [TestLink](https://www.testlink.org/)
* [TestRail](https://www.gurock.com/testrail/test-management/)

Bugs:

* [Mantis](https://www.mantisbt.org/)
* [Bugzilla](https://www.bugzilla.org/)
* [Jira](https://www.atlassian.com/es/software/jira?&aceid=&adposition=&adgroup=109687540304&campaign=10332064761&creative=443576046641&device=c&keyword=jira&matchtype=e&network=g&placement=&ds_kids=p55122863591&ds_e=GOOGLE&ds_eid=700000001550060&ds_e1=GOOGLE&gclid=Cj0KCQiApL2QBhC8ARIsAGMm-KHR3FOE7jHq1NLZ6MsjQ4KcKHzSiLFCRmwYtuWPtvC8AFkPwGtaFWMaAqscEALw_wcB&gclsrc=aw.ds)

Capturas de pantalla:

* [Lightshot](https://app.prntscr.com/es/)
* [ApowerSoft](https://www.apowersoft.es/grabador-de-pantalla-gratis)
* [Greenshot](https://getgreenshot.org/)

#### ¿Falta documentación?

La falta de documentación, es una realidad más común de la que creemos, y es una de nuestras responsabilidades como tal hacer pedido de ella en caso de que no exista.

Ciertos tipos de pruebas, llamadas pruebas exploratorias, hacen uso nulo de documentaciones, pero son solo una primera instancia de trabajo. El uso de la documentación es algo sumamente prioritario.

### Métricas

Las métricas son otra de nuestras herramientas de trabajo, ya que es lo que usamos para medir y presentar la cantidad de trabajo realizado.

Se realizan métricas en base a casos realizados, bugs encontrados, tiempo de trabajo implementado, etc...

Es un estándar de medida que nos permite tomar decisiones.

Ejemplos de métricas en el Testing:

* Cantidad de errores encontrados con severidad alta
* Cantidad de casos de prueba que fallaron
* Cantidad de errores por tipo y criticidad
* Cantidad de ciclos de testing

### Adaptabilidad

Una cualidad del tester, y responsabilidad, es comprender que tenemos que adaptarnos a las circunstancias que nos van surgiendo. Ya sea ambientes de prueba, falta de documentación, cambios de último momento, etc… La adaptabilidad es una herramienta que se construye para poder adaptar a lo que tengamos que probar y bajo las condiciones que surjan.

# Técnicas de Prueba

## Introducción

El **objetivo de una técnica de prueba** es ayudar a identificar las condiciones de prueba, los casos de prueba y los datos de prueba.

Algunas técnicas son más adecuadas para ciertas situaciones y niveles de prueba; otras se pueden aplicar en todos los niveles de prueba.

En general, cuando se crean casos de prueba, los probadores utilizan una combinación de técnicas de prueba para lograr los mejores resultados a partir del esfuerzo de prueba.

### Técnicas de Prueba y sus Características

Las técnicas de prueba se clasifican:

* Caja negra.
* Caja blanca
* Basadas en la experiencia.

### Caja negra

#### Definición

También llamadas técnicas conductuales o basadas en el comportamiento, se basan en un análisis de la base de prueba adecuada (por ejemplo, documentos de requisitos formales o historias de usuarios). **Estas técnicas son aplicables tanto a la prueba funcional como a la no funcional.**

Las técnicas de prueba de caja negra se concentran en las entradas y salidas del objeto de prueba sin referencia a su estructura interna.



Fuente: [Panel.es](https://www.panel.es/software-qa-cuales-son-los-tipos-de-pruebas-software/)

#### Características

* Las condiciones de prueba, casos de prueba y datos de prueba se deducen de la documentación.
* Los casos de prueba se pueden utilizar para detectar diferencias entre los requisitos y la implementación de los requisitos.
* La cobertura se mide en función de los elementos probados y de la técnica aplicada a la base de prueba.

#### Definición de Caso de Prueba

* Es un conjunto de pasos y resultados esperados que se crean a partir de los requerimientos del software que se va a probar.
* Es un conjunto de condiciones o variables bajo las cuales un tester determinará si un sistema o software o una característica de éstos es parcial o completamente satisfactoria.
* Conocido también como Test Case.

#### Técnicas de prueba de caja negra

##### Partición de Equivalencia

* Consiste en **clasificar las entradas de datos del sistema en grupos que representan un comportamiento similar.**
* Se pueden definir particiones tanto para datos válidos (aceptados por el sistema) como no válidos (no aceptados por el sistema)
* Para lograr una cobertura del 100% con esta técnica, **los casos de prueba deben cubrir todas las particiones identificadas utilizando, como mínimo, un valor de cada partición.**

Ejemplo

* Tenemos un sistema electoral donde personas de un país deberán votar. Analizaremos la variable “edad”.
* Dividimos en dos grupos “mayores de edad” y “menores de edad”

| **Menores** | **Mayores** |
| --- | --- |
| 1….17 | 18….n |

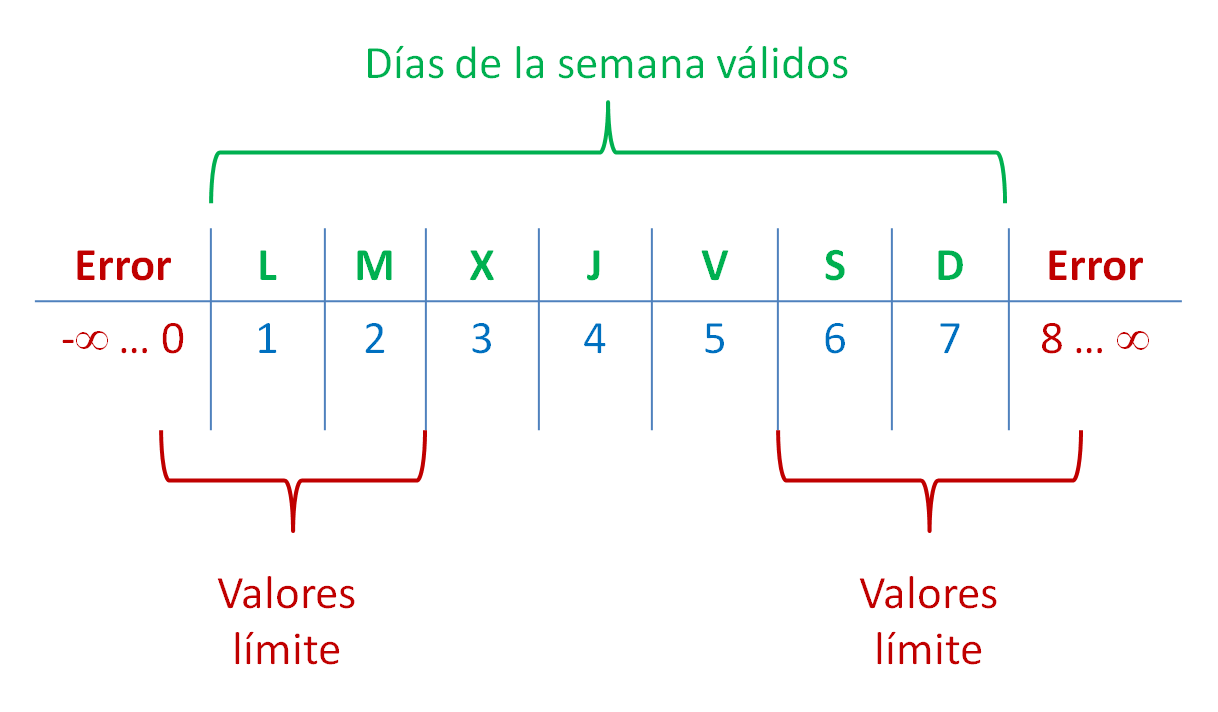
##### Análisis de Valores Frontera

Es una **extensión de la partición de equivalencia, pero sólo se puede utilizar cuando la partición está ordenada, y consiste en datos numéricos o secuenciales.**

Los valores mínimo y máximo de una partición son sus valores frontera.

El análisis de valores frontera se puede aplicar en todos los niveles de prueba. Esta técnica **se utiliza generalmente para probar los requisitos que requieren un rango de números, fechas y horas.**

Por ejemplo, suponer que un campo de entrada acepta un único valor entero como entrada, utilizando un teclado numérico para limitar las entradas de modo que las entradas no enteras sean imposibles. El rango válido es de 1 a 5, ambos valores incluidos. Por lo tanto, hay tres particiones de equivalencia: inválida (demasiado baja); válida; inválida (demasiado alta). Para la partición de equivalencia válida, los valores frontera son 1 y 5. Para la partición no válida (demasiado alta), los valores frontera son 6 y 9. Para la partición inválida (demasiado baja), sólo hay un valor frontera, 0, porque se trata de una partición con un solo miembro.



Fuente: [Educación con TIC](https://educandocontic.com/valores-limite-pruebas/)

Ejemplo 2

* Un campo de un formulario solo acepta números del 1 al 5.

Partición de equivalencia

| **Inválida (demasiado baja)** | **Válida** | **Inválida (demasiado alta)** |
| --- | --- | --- |
| 0 | 1,2,3,4,5 | 6,7,8,9 |

Valores frontera

| **Inválida (demasiado baja)-Válida** | **Válida- Inválida (demasiado alta)** |
| --- | --- |
| 0 y 1 | 5 y 6 |

##### Prueba de Tabla de Decisión

Las técnicas de prueba combinatorias son **útiles para probar la implementación de requisitos de sistema**; que especifican como diferentes combinaciones de condiciones generan diferentes resultados.

Al crear tablas de decisión, el probador identifica las condiciones y las acciones resultantes del sistema. Éstas conforman las condiciones en la parte superior y las acciones en la parte inferior.

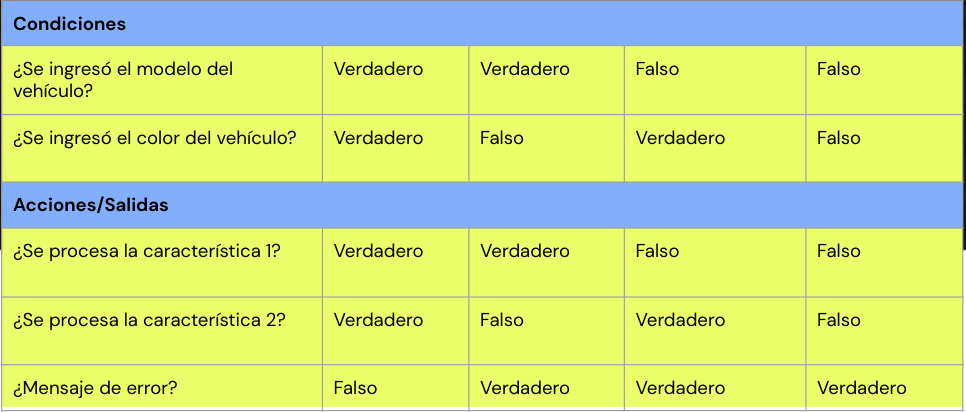
Ejemplo

1. Identificamos condiciones o entradas
2. Identificamos acciones o salidas
3. Identificamos combinaciones

Tenemos las siguientes condiciones:

* Ingresar modelo de vehículo
* Ingresar color de vehículo
* Condición/Combinación



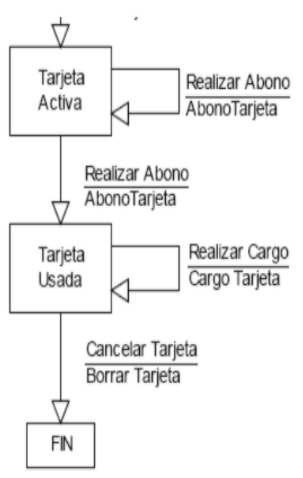


##### Prueba de Transición de Estado

Un diagrama de transición de estado muestra los posibles estados del software, así como la forma en que el software entra, sale y realiza las transiciones entre estados.

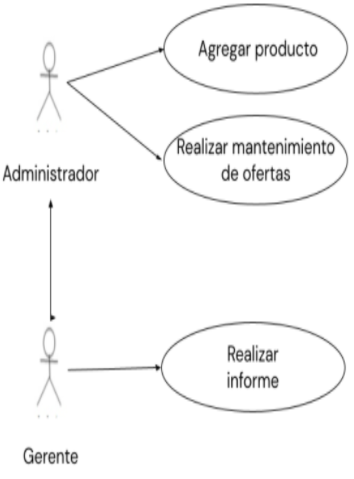
Una transición se inicia con un evento. El evento resulta en una transición.

El cambio de estado puede provocar que el software tome una acción.

Este es un ejemplo de los posibles estados por los que pasa una tarjeta en un sistema. La idea es analizar y realizar pruebas en todos los estados por los que pasa una entidad en esta caso la tarjeta. Si el software tiene distintas transiciones de estados tengo que analizar y ver como responde y se comporta en cada una de ellas.

##### Prueba de Caso de Uso

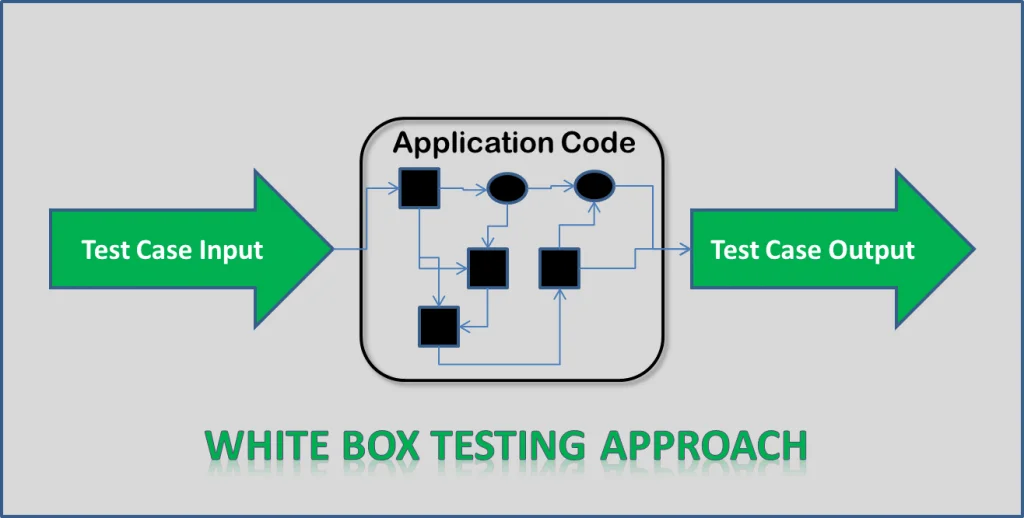
Las pruebas se pueden obtener a partir de casos de uso, que son una forma específica de diseñar interacciones con elementos software, incorporando requisitos para las funciones del software representadas por los casos de uso. Los casos de uso están asociados con actores y sujetos.



### Caja blanca

#### Definición

También llamadas técnicas estructurales o basadas en la estructura, **se basan en un análisis de la arquitectura, el diseño detallado, la estructura interna o el código del objeto de prueba.** A diferencia de las técnicas de prueba de caja negra, las técnicas de prueba de caja blanca se concentran en la estructura y el procesamiento dentro del objeto de prueba.



Comúnmente se dice que son “las pruebas de código”, pero esto es una falacia. Es importante tener en mente que la caja blanca también lleva arquitectura, infraestructura, y otros análisis más en detalle de cómo el sistema está construido.

#### Características

* Las condiciones de la prueba, los casos de prueba y los datos de la prueba se deducen del código, la arquitectura, el diseño detallado o cualquier información de la estructura.
* La cobertura se mide en base a los elementos probados dentro de una estructura seleccionada.
* Las especificaciones se utilizan como fuente adicional para determinar el resultado esperado de los casos de prueba.

#### Técnicas de prueba de caja blanca

##### Prueba y Cobertura de Sentencia

La prueba de sentencia practica las sentencias ejecutables en el código. **La cobertura se mide como el número de sentencias ejecutadas por las pruebas dividido por el número total de sentencias ejecutables en el objeto de prueba.**

##### Prueba y Cobertura de Decisión

La prueba de decisión **practica las decisiones en el código y prueba el código que se ejecuta basado en los resultados de la decisión.** Para ello, siguen los flujos de control que se producen desde un punto de decisión.

La cobertura se mide como el número de resultados de decisión ejecutados por las pruebas dividido por el número total de resultados de decisión en el objeto de prueba.

##### El Valor de la Prueba de Sentencia y Decisión

Una cobertura del 100% de sentencia, se asegura de que todas las sentencias ejecutables del código se han probado al menos una vez.

**En una cobertura del 100% de decisión, se ejecutan todos los resultados de decisión, lo que incluye probar el resultado verdadero y también el resultado falso.**

Una cobertura del 100% de decisión garantiza una cobertura del 100% de sentencia.

### Basadas en la experiencia

#### Definición

Las técnicas de prueba basadas en la experiencia aprovechan la experiencia de desarrolladores, probadores y usuarios para diseñar, implementar y ejecutar pruebas. A menudo, estas técnicas se combinan con técnicas de prueba de caja negra y caja blanca.

#### Características

* Las condiciones de prueba, los casos de prueba y los datos de prueba se deducen de una base de prueba que puede incluir el conocimiento y la experiencia de probadores, desarrolladores, usuarios y otros implicados.
* Este conocimiento y experiencia incluye el uso esperado del software, su entorno, los posibles defectos y la distribución de dichos defectos.

#### Técnicas de prueba basadas en la experiencia

##### Predicción de Errores

Es una técnica utilizada para anticipar la ocurrencia de equivocaciones, defectos y fallos, basada en el conocimiento del probador, incluido:

* Cómo ha funcionado la aplicación en el pasado.
* Qué tipo de equivocaciones tienden a cometer los desarrolladores.
* Fallos que se han producido en otras aplicaciones.

##### Prueba Basada en Listas de Comprobación

En la prueba basada en listas de comprobación, **los probadores diseñan, implementan y ejecutan pruebas para cubrir las condiciones de prueba que se encuentran en una lista de comprobación.**

Estas listas de comprobación pueden elaborarse basándose en la experiencia, el conocimiento de lo que es importante para el usuario o la comprensión de por qué y cómo falla el software.

##### Prueba Exploratoria

**Se diseñan, ejecutan, registran y evalúan de forma dinámica pruebas informales durante la ejecución de la prueba.** Los resultados de la prueba se utilizan para aprender más sobre el componente o sistema, y para crear pruebas para las áreas que pueden necesitar ser probadas con mayor intensidad.

Esta prueba es más útil cuando las especificaciones son escasas o inadecuadas.

#### Glosario

**Verificar:** cuando el software se ajusta a su especificación.

**Validar:** cuando el software se ajusta a lo que el cliente realmente pidió

**Pruebas estáticas:** son la acción de leer documentación.

**Pruebas dinámicas:** es la propia acción de probar un sistema.

**Caso de Prueba:** conjunto de pasos y resultados esperados que se crean a partir de los requerimientos del software que se va a probar.

**Métricas:** es un estándar de medida que nos permite tomar decisiones.

#### ¿QUIERES SABER MÁS? TE DEJAMOS MATERIAL AMPLIADO DE LA CLASE

* [Ejemplos de los 7 Principios de Testing](https://www.youtube.com/watch?v=nI28eoU6POQ)
* [Documento ISTQB en Español - Página 70 a 78](https://141647fc-7a5b-4407-a84a-78567bb3f5b0.filesusr.com/ugd/acfdb9_743bf3d31dde49578c94d97e5b96b9da.pdf)
* [Ejemplo prueba exploratoria](https://www.youtube.com/watch?v=8v17O8RlKdA)

#DEMOCRATIZANDOLAEDUCACIÓN