**Pruebas de Aceptación**

El uso de cualquier producto de software tiene que estar justificado por las ventajas que ofrece. Sin embargo, antes de empezar a usarlo es muy difícil determinar si sus ventajas realmente justifican su uso. El mejor instrumento para esta determinación es la llamada **prueba de aceptación**. En esta prueba se evalúa el grado de calidad del software con relación a todos los aspectos relevantes para que el uso del producto se justifique.  
  
Para eliminar la influencia de conflictos de intereses, y para que sea lo más objetiva posible, la prueba de aceptación nunca debería ser responsabilidad de los ingenieros de software que han desarrollado el producto.   
Para la preparación, la ejecución y la evaluación de la prueba de aceptación ni siquiera hacen falta conocimientos informáticos. Sin embargo, un conocimiento amplio de métodos y técnicas de prueba y de la gestión de la calidad en general facilita esta labor.   
  
La persona adecuada (o el equipo adecuado) para llevar a cabo la prueba de aceptación dispone de estos conocimientos y además es capaz de interpretar los requerimientos especificados por los futuros usuarios del sistema de software en cuestión.

**Pruebas de Caja Blanca**

La prueba de la caja blanca es un método de diseño de casos de prueba que usa la estructura de control del diseño procedimental para derivar los casos de prueba.

Las pruebas de caja blanca se llevan a cabo en primer lugar, sobre un módulo concreto, para luego realizar las de caja negra sobre varios subsistemas (integración).

En los sistemas orientados a objetos, las pruebas de caja blanca pueden aplicarse a los métodos de la clase, pero según varias opiniones, ese esfuerzo debería dedicarse a otro tipo de pruebas más especializadas (un argumento podría ser que los métodos de una clase suelen ser menos complejos que los de una función de programación estructurada). Dentro de las Pruebas de Caja Blanca encontramos las llamadas coberturas (sentencia, decisión, condición y múltiple además de los mencionados caminos ciclomáticos propuestos por McCabe).

Las pruebas de caja blanca intentan garantizar que:

* Se ejecutan al menos una vez todos los caminos independientes de cada módulo.
* Se utilizan las decisiones en su parte verdadera y en su parte falsa.
* Se ejecuten todos los bucles en sus límites.
* Se utilizan todas las estructuras de datos internas.

**Pruebas de Caja Negra**

Estas pruebas permiten obtener un conjunto de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. En ellas se ignora la estructura de control, concentrándose en los requisitos funcionales del sistema y ejercitándolos.

La prueba de Caja Negra no es una alternativa a las técnicas de prueba de la Caja Blanca, sino un enfoque complementario que intenta descubrir diferentes tipos de errores a los encontrados en los métodos de la Caja Blanca. Muchos autores consideran que estas pruebas permiten encontrar:

1. Funciones incorrectas o ausentes.
2. Errores de interfaz.
3. Errores en estructuras de datos o en accesos a las Bases de Datos externas.
4. Errores de rendimiento.
5. Errores de inicialización y terminación.

Para preparar los casos de pruebas hacen falta un número de datos que ayuden a la ejecución de los estos casos y que permitan que el sistema se ejecute en todas sus variantes, pueden ser datos válidos o inválidos para el programa según si lo que se desea es hallar un error o probar una funcionalidad. Los datos se escogen atendiendo a las especificaciones del problema, sin importar los detalles internos del programa, a fin de verificar que el programa corra bien.

Para desarrollar la prueba de caja negra existen varias técnicas, entre ellas están:

1. Técnica de la Partición de Equivalencia: esta técnica divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software.
2. Técnica del Análisis de Valores Límites: esta Técnica prueba la habilidad del programa para manejar datos que se encuentran en los límites aceptables.
3. Técnica de Grafos de Causa-Efecto: es una técnica que permite al encargado de la prueba validar complejos conjuntos de acciones y condiciones.

Dentro del método de Caja Negra la técnica de la Partición de Equivalencia es una de las más efectivas pues permite examinar los valores válidos e inválidos de las entradas existentes en el software, descubre de forma inmediata una clase de errores que, de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico. La partición equivalente se dirige a la definición de casos de pruebas que descubran clases de errores, reduciendo así en número de clases de prueba que hay que desarrollar.

**Pruebas de Humo**

Una prueba de humo o smoke test, es un testing rápido que se realiza sobre aspectos funcionales no tanto para encontrar bugs sino para asegurarse que la funcionalidad básica del software o de una parte del software se encuentre estable y responda al comportamiento esperado.

El objetivo es verificar, con pruebas sencillas y que demanden poco tiempo, que ciertos caminos de la aplicación funcionen correctamente. Normalmente se elige un conjunto de funcionalidades significativas, no hace falta que sean todas las de la aplicación.

Es frecuente aplicar este tipo de pruebas previo a una entrega al cliente, de manera que se realice una última comprobación para verificar que a alto nivel todo está bien (antes, como es lógico, se deberían haber realizado pruebas en profundidad del sistema en todos los niveles y no solo pruebas funcionales).

Las pruebas de humo ponen de manifiesto si el software está o no lo suficientemente estable para afrontar un ciclo de pruebas y son un paso previo a la ejecución del plan de pruebas diseñado previamente.

Las características de un smoke test son las siguientes:

* Tests cortos, idealmente un caso para probar por arriba que el feature funcione. Si esta es compleja siempre tratar de tener la mínima cantidad de casos.
* Idealmente no deben encontrarse bugs durante este test o muy pocos. Como dije antes, sirve para quedarnos tranquilos que todo funciona correctamente. Para encontrar bugs estuvieron las pruebas previas, más exhaustivas en cada feature.
* El tiempo de ejecución del smoke test completo debe ser corto.
* Los casos por lo general son positivos. O sea, comportamiento esperado en situaciones en la que el usuario no hace nada raro.
* Debe correrse una vez que se corrieron todos los casos nuevos y regresiones en el producto. Como un chequeo final antes del release.