INDUCCIÓN – TESTING

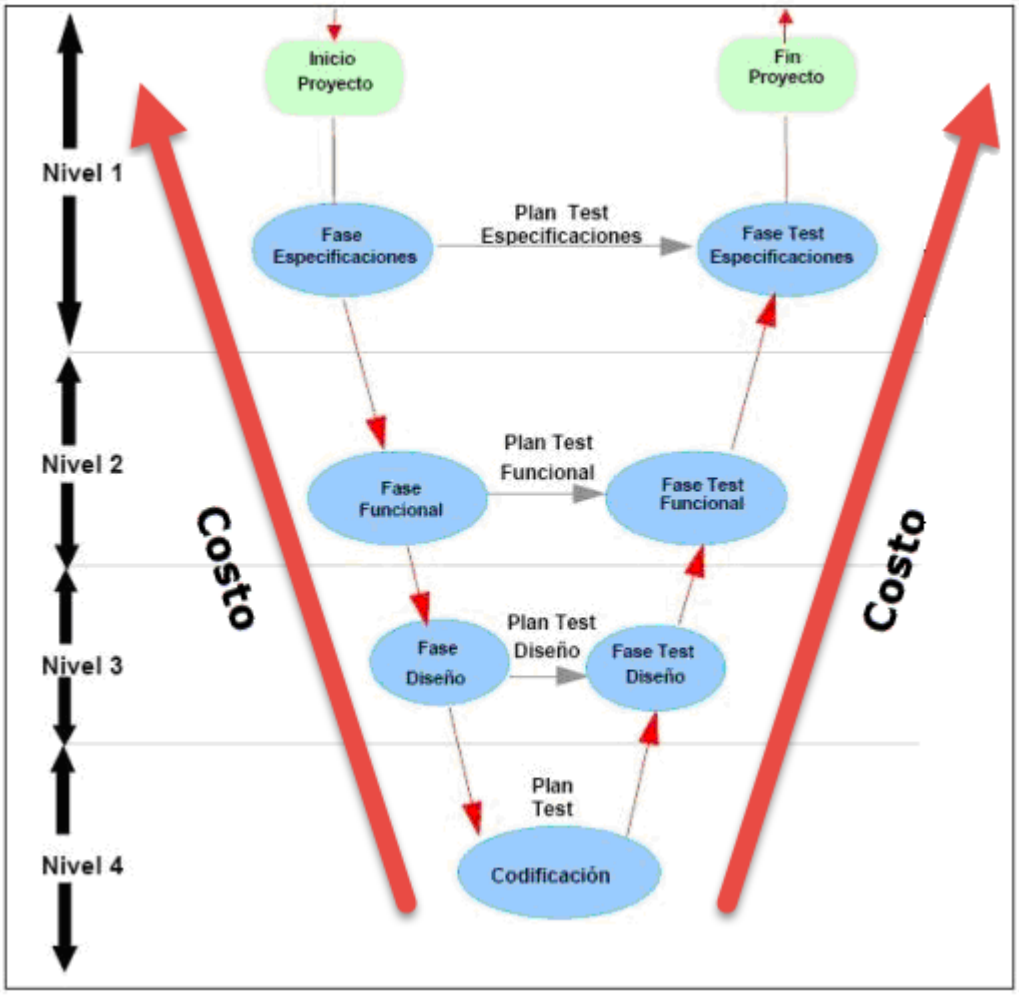
Que son las pruebas de software

Las **pruebas de software** (en inglés *software testing*) son las investigaciones empíricas y técnicas cuyo objetivo es proporcionar información objetiva e independiente sobre la calidad del producto a la parte interesada o stakeholder.

Las pruebas son básicamente un conjunto de actividades dentro del desarrollo de software. Dependiendo del tipo de pruebas, estas actividades podrán ser implementadas en cualquier momento de dicho proceso de desarrollo. Existen distintos modelos de desarrollo de software, así como modelos de pruebas. A cada uno corresponde un nivel distinto de involucramiento en las actividades de desarrollo.

Porque son necesarias las pruebas de software

La fase de pruebas que se incluye dentro del ciclo de vida del software, nos permite identificar diferentes tipos de incidencias (errores, fallas, mejoras, etc.), estas incidencias se pueden originar desde un diseño erróneo de software, una mala interpretación del objetivo entre otras cosas. Cuando más temprana en la fase en donde se detectan las incidencias, menos costosa es su resolución.



Diferentes Tipos de pruebas

Podemos separar los tipos de pruebas en 2 grandes grupos, Manuales y Automáticas, las manuales son enteramente diseñadas, creadas y ejecutadas por una persona, en cambio las Automáticas, si bien son diseñadas y creadas manualmente la ejecución se realizara a través de un robot (selenium, qtp, rotational, etc).

Podemos destacar también los tipos de prueba de Caja Blanca y Caja Negra

Dentro de estos tipos de pruebas podemos encontrar sub-categorías en cada uno de los grupos

Pruebas Funcionales, no funcionales, estrés, volumen, performance, pruebas de humo, regresión, usabilidad entre otras

INDUCCIÓN – TESTING

Tipos de Incidencias

Existen varios tipos de incidencias, aquí veremos las más usualmente utilizadas, las cuales son: Error, Falla, Mejora

Diferencia entre Error, Defecto y Falla:

- Error: Acción Humana que produce un resultado incorrecto
- Defecto: desperfecto en un componente o sistema que puede causar que el componente o sistema falle en desempeñar las funciones requeridas
- Falla: Manifestación física o funcional de un defecto

```
void MinMax (int Min, int Max)
{
      int Help;
            if (Min>Max)
            {
                  Max = Help;
                  Max = Min;
                  Help = Min;
            }
End MinMax;
```

Error ("Error"):

- Acción humana que produce un resultado incorrecto.
- **Ejemplo: Un error de programación**

Defecto ("Defect"):

- Desperfecto en un componente o sistema que puede causar que el componente o sistema falle en desempeñar las funciones requeridas.
- **Ejemplo: Una sentencia o una definición de datos incorrectas.**

Fallo ("Failure"):

- Manifestación física o funcional de un defecto.
- **Ejemplo: Desviación de un componente o sistema respecto de la prestación, servicio o resultado esperados.**

INDUCCIÓN – TESTING

Que es un caso de prueba

Un caso de prueba, en ingeniería del software, un conjunto de condiciones o variables bajo las cuáles un analista determinará si una aplicación, un sistema software, o una característica de éstos son parcial o completamente satisfactoria.

Estructura de Casos de prueba

Código Test (nroOT_orden)
Funcionalidad
Descripción de Caso de Test
Pre-Condiciones
Pasos
Resultado Esperado
Resultado Obtenido
Nivel de Critidad del TC
Browser
Numero de Incidencia
Comentarios
Fecha
Responsable

Template TC.xlsx

Como Construir un caso de Prueba

1. Identificar el Alcance y el Propósito de las Pruebas

El primer punto clave antes de empezar es identificar los requerimientos verificables. Es preciso entender el propósito de las Pruebas, las funcionalidades o características, y los requisitos del usuario.

2. Descripción del Caso de Prueba

La descripción es donde se mencionan todos los detalles sobre las funcionalidades o características a ser verificadas, y también el comportamiento particular verificado por la Prueba en sí.

Lo que necesita verificarse, el ambiente de Prueba y la información sobre la misma; toda ésa información debe ser incluída en la descripción.

3. Asunciones y Precondiciones

Al escribir los Casos de Prueba, se deben comunicar todas las asunciones que aplican a las Pruebas, junto con las pre-condiciones que deben cumplirse antes que se pueda ejecutar el Caso de Prueba

4. Escribir un Caso de Prueba para cada condición

Hay una confusión bastante común en la que algunos desarrolladores traducen “un Caso de Prueba para cada condición” como “un Caso de Prueba para cada aserción”, lo cual está mal. Una condición puede contener una o más aserciones. A la hora de seleccionar las condiciones a verificar, es también una buena idea concentrarse en los puntos límites del sistema, experimentando rangos de valores que pongan a prueba los mismos

5. Casos de Prueba positivos y negativos

Los métodos utilizados para escribir Casos de Prueba pueden variar, desde Particiones de Equivalencia, Análisis de los valores de límite, condiciones normales o anormales, etc. También se debe tener en cuenta a los Casos de Prueba negativos, condiciones de fallos y manejo de errores que podrían ayudar a descubrir los errores más probables en el código. Está prohibido asumir el comportamiento de cualquier funcionalidad, los Casos de Prueba deben escribirse con referencia al documento de requerimientos y especificaciones.

6. Legibles y fáciles de entender

Es importante tener en cuenta que no siempre es la misma persona quien diseña y ejecuta los Casos de Prueba. Entonces, los mismos deben ser objetivos, legibles y fáciles de entender.

7. Mantenimiento y Reusabilidad

Los Casos de Prueba deben ser escritos teniendo en cuenta que pueden ser reutilizados en el futuro para otros proyectos o equipos.

8. Estructuración de los Casos de Prueba

Los Casos de Prueba de un Plan de Pruebas deben ser independientes, pero a la vez agrupados por una idea en común. En ése caso, las dependencias entre los mismos son inevitables, una cadena de Pruebas relacionadas no debería tener más de 3-5 unidades y un Analista de QA debe verificar el estado de las Pruebas anteriores antes de ejecutar la siguiente.

9. Ingreso de los datos de las Pruebas

Identificar los datos de las Pruebas puede ser una actividad que consume mucho tiempo. Muchas veces la preparación se lleva la mayoría del tiempo en un ciclo de Pruebas.

10. Verificar sólo una acción o una funcionalidad por vez.

Verificar sólo una acción o funcionalidad por vez puede ser complicado algunas veces, debido a las dependencias en un sistema. Se puede escribir un Caso de Pruebas para una clase que indirectamente use la base de datos, o una tercera clase, en éste caso, la mejor solución es usar un objeto falso.

*Los Casos de Prueba deben ser transparentes, atómicos y fáciles de entender. Una Prueba **debe verificar sólo un sujeto**, que puede ser pequeño (parte de una funcionalidad) o grande (de extremo a extremo)*

Plan de Pruebas

Un plan de pruebas detallara el/los Objetivos, la Estrategia de Pruebas y el Alcance de lo que se va a probar, entre otras cosas, algunos Planes de prueba incluyen las pruebas a realizar.

***Fuente**: http://www.pmoinformatica.com*

Ejemplo de Plan de pruebas.doc