**TESTING “Prueba de Software”**

Trabajo presentado al Instructor: Yohon Jairo Bravo

Trabajo presentado por el grupo el grupo SGEDI

Pedro Luis Aullon

Yurani Cambindo Granja

Urlandy Ibarra Peralta

Johnatan Rios Lopez

Hector Fabio Rodriguez

Asignatura: Requerimiento

Periodo: tercer trimestre

**PALMIRA**

**CENTRO DE BIOTECNOLOGIA INDUSTRIAL SENA PALMIRA**

**2017**

**OBJETIVOS**

1. Mostrar hasta qué punto las funciones del software operan de acuerdo con las especificaciones y requisitos del cliente.
2. Descubrir un error que aún no ha sido descubierto.
3. Probar si el software no hace lo que debe.
4. Identificar posibles fallos de implementación, calidad o usabilidad de un proyecto.

**TESTING “Prueba de Software”**

1. ¿Qué es testing de Software??

Investigación técnica de un producto bajo prueba con el fin de brindar información relativa de la calidad del software, a los diferentes actores involucrados en un proyecto.

A partir de la información obtenida del testing se pueden tomar decisiones. Las decisiones pueden ser desde cuándo liberar un producto a producción, conociendo los riesgos que esto implica, hasta cómo mejorar las diferentes áreas dentro de la empresa. En definitiva, el testing es un agente de cambio, lo importante es interpretar la información obtenida para que todos los actores puedan actuar en forma oportuna donde sea necesario.

En el software la confianza es un elemento importante ya que ciertas fallas pueden tener consecuencias indeseables como pérdidas de dinero, negocios e incluso de vidas, dependiendo de qué tan crítico sea el dominio en el cual el software interactúa. Las pruebas le dan valor agregado a cada proyecto brindando confianza a los distintos actores.

En resumen, el testing es una actividad cognitiva y no mecánica ni repetitiva que involucra varias funciones mentales como el lenguaje, la imaginación, percepción, entre otros. Un tester va desarrollando estas habilidades a medida que avanza en la carrera de testing y obtiene conocimiento y experiencia.

1. ¿Porque es importante realizar pruebas de Software?

Los proyectos basados en desarrollo de software son creados, desarrollados e implementados por seres humanos y por ende en cualquiera de sus etapas de creación se puede presentar una equivocación, que puede conllevar a un defecto en el software, por ejemplo, mala digitación, distracción al codificar, mala elaboración de un documento entre otras. Si no se ha identificado ese defecto y el software o la aplicación se ejecuta, hay un alto riesgo de que la aplicación no haga lo que debería hacer o el objetivo para el cual fue creado, es decir se genera un fallo o desperfecto, lo que podría generar una catástrofe y muchas otras más, es importante conocer que los fallos también se pueden presentar por situaciones del entorno, como la radiación, descarga eléctrica, contaminación, inundaciones, húmeda, fuego, etc.

Es importante realizar una prueba de Software para cumplir tres objetivos en específico:

* Identificar un error en el desarrollo de software.
* Descubrir errores no antes vistos.
* Tener un buen caso de prueba.

Aspectos que se deben tener en cuenta para realizar una prueba de software:

* Plantear y diseñar las pruebas antes de generar el código
* Deben realizarse por un equipo independiente al de desarrollo.

1. ¿Cuáles son los diferentes tipos de testing de software?

**Pruebas Funcionales**: Típicamente encontraremos el comportamiento del sistema, subsistemau o componente software descrito en especificaciones de requisitos o casos de uso, Es decir, con las funciones establecemos “*lo que el sistema hace*”.

Estas pruebas se definen a partir de funciones o características (como decimos, bien descritas en documentos o bien interpretadas por los probadores) y su interoperabilidad con sistemas específicos, pudiendo ejecutarse en todos los niveles de pruebas (componentes, integración, sistema, etc.).

Se consideran Pruebas de Caja Negra (“black-box testing”) puesto que valoramos el comportamiento externo del sistema. Las Pruebas de Seguridad o las Pruebas de Interoperabilidad entre sistemas o componentes son casos especializados de las pruebas funcionales.

**En segundo lugar, figuran las Pruebas no Funcionales** que incluyen las pruebas de: Rendimiento, Carga, Estrés, Usabilidad, Mantenibilidad, Fiabilidad o Portabilidad, entre otras. Por tanto, se centran en características del software que establecen “cómo trabaja el sistema “.

Estas pruebas también pueden ejecutarse en todos los [niveles de pruebas](http://panel.es/blog/software-qa-software-testing-levels-onion-model/). Las características no funcionales del software se pueden medir de diversas maneras, por ejemplo, por medio de tiempos de respuesta en el caso de pruebas de rendimiento o por número máximo de sesiones en pruebas de estrés.

Puesto que las Pruebas no Funcionales normalmente consideran el comportamiento externo del sistema, en la mayoría de los casos se utilizan técnicas de Pruebas de Caja Negra.

**A continuación, en tercer lugar, tenemos las Pruebas Estructurales.** Nuevamente pueden ejecutarse en todos los niveles de pruebas (componentes, integración, sistema, etc.) y encajan muy bien si hemos utilizado técnicas de especificación de la estructura o arquitectura del Software. Es posible aplicar técnicas estáticas de análisis de código.

Para expresar el alcance con un conjunto de pruebas (“test suite”) que ha cubierto la estructura o arquitectura en cuestión, se utiliza el concepto de Cobertura (“Coverage”), normalmente en forma de porcentaje.

Es especialmente habitual utilizar herramientas de apoyo para calcular la cobertura del código en el caso de Pruebas de Componentes o en Pruebas de Integración de Componentes (por ejemplo, trazando la jerarquía de llamadas entre elementos). Puesto que indagamos en el comportamiento interno, estas pruebas se denominan también Pruebas de Caja Blanca (“white-box testing”).

**Finalmente, el cuarto tipo de pruebas que nos presenta el** [**ISTQB**](http://www.istqb.org/) **(Junta Internacional de Calificación de Pruebas de Software)** son las pruebas derivadas de la realización de cambios: las Pruebas de Regresión y las Re-pruebas.

Una vez que un defecto ha sido corregido, toca volver a probar el software para confirmar que el defecto ha sido eliminado. Son pruebas repetidas o Re-Pruebas.

**Las Pruebas de Regresión** consiste en volver a probar un componente, tras haber sido modificado, para descubrir cualquier defecto introducido, o no cubierto previamente, como consecuencia de los cambios. Los defectos pueden encontrarse tanto en el software que se ha cambiado como en algún otro componente. Se ejecutan cuando se cambia el software o su entorno. El criterio para decidir la extensión de estas Pruebas de Regresión está basado en el riesgo de no encontrar defectos en el software que anteriormente estaba funcionando correctamente.

Las Pruebas de Regresión se realizan sobre un componente ya probado, para verificar que no presenta nuevos defectos cuando se realiza una modificación después de dichas pruebas.

Este tipo de pruebas deben ser repetibles si han de usarse para pruebas de confirmación (o aseguramiento) y regresión (como [Sondas de Disponibilidad](http://panel.es/blog/software-qa-sondas-de-disponibilidad/), por ejemplo). Los conjuntos de pruebas de regresión (“*Regression test suites*“) suelen ser bastante estables por lo que son muy buenos candidatos para actividades de automatización de pruebas.

Quizá ya nos sorprenda que estas pruebas también pueden ejecutarse en todos los [niveles de pruebas](http://panel.es/blog/software-qa-software-testing-levels-onion-model/) e incluyen casos de prueba de los tipos vistos anteriormente: Pruebas Funcionales, No Funcionales y Estructurales.

1. ¿Qué es el testing de Caja Negra?

La estrategia de testing de la caja negra es la verificación de las funcionalidades de la aplicación: Datos que entran, resultados que se obtienen, interacción con los actores, funcionamiento de la interfaz de usuario y en general todo aquello que suponga estudiar el correcto comportamiento que se espera del sistema.

1. ¿Cómo se realiza el Testing de Caja Negra?

Las pruebas de caja negra se limitan a que el tester pruebe con *“datos”* de entrada y estudie como salen, sin preocuparse de lo que ocurre en el interior. Principalmente, se centran en módulos o charters de interfaz de usuario (pantalla, ficheros, canales de comunicación…) pero suelen ser útiles en cualquier módulo ya que todos o la mayoría tienen datos de entrada y salida que se pueden comprobar y verificar. Como cualquier otra prueba, las de caja negra se apoyan y basan en la especificación de requisitos y documentación funcional, estos requisitos suelen ser más complejos que los internos.

1. ¿Cuáles son los beneficios del testing funcional?

* Los errores son muy costosos de corregir una vez liberado el producto / versión, es mucho menor el costo si se encuentra el problema en la etapa de test.
* Asegura la calidad, si está bien realizado crea confianza en el cliente sobre el producto
* Al tener que pensar los casos de prueba, ayuda a que surjan nuevos casos de uso, elementos que no se tuvieron en cuenta.
* Satisfacción del usuario final con el producto
* Es importante conocer el comportamiento del sistema en los casos bordes, someterlo a estrés, etc.
* Mediante el resultado de la prueba, ayuda a tomar decisiones de arquitectura, usabilidad, etc.
* Determinante para determinar errores y fallas en forma oportuna en una etapa previa a la de puesta en producción.
* evitar la frustración de los clientes/usuarios
* Confiabilidad, estabilidad, escalabilidad, eficiencia y seguridad
* Rapidez frente a los cambios
* Conocer con cierto nivel de certeza en qué el sistema hace lo que se entendió que los usuarios esperan que haga.

CONCLUSIONES

* Las pruebas se organizan en metodologías que son aplicadas dependiendo del tipo de proyecto que se esté realizando.
* Las pruebas no pueden garantizar que el software no presente errores, su propósito es minimizarlos.
* Herramienta crítica para alcanzar la calidad del software.

Enlace del video.

<https://youtu.be/zjQlO5Z4UvU>