



Buenas prácticas de software



Unidad 4. Buenas prácticas para los modelos de prueba de software.



Buenas prácticas de software



4.4 Medir el costo de las pruebas.



Buenas prácticas de software



4.4.1 Amplitud y cobertura de las pruebas

La cobertura es la cantidad de código que se revisa en las pruebas.

Se dice que si en la prueba del software, alguna línea del código no se ejecuta, entonces dicha línea no está cubierta.

La cobertura de camino es: El porcentaje de líneas de código que se reviso en las pruebas.

Si se realiza el 100% de la cobertura de caminos en las pruebas entonces el se cubrió el 100% de los recorridos en las pruebas.

El porcentaje de cobertura del código revisado en las pruebas se calcula con el porcentaje de cobertura de código = $(\text{Número de líneas de código ejecutadas por un algoritmo de prueba} / \text{Número total de líneas de código en un componente del sistema}) * 100$.



Buenas prácticas de software



Técnicas de cobertura de pruebas.

Cobertura del producto.

La cobertura del producto se refiere a las pruebas que se hacen desde la perspectiva del producto. Es decir considerar cada una de las partes que integra al software.

Supongamos que se debe probar una aplicación simple como una calculadora. Aunque debe verificar las funciones esenciales como las cuatro operaciones aritméticas, eso no es suficiente. También debe considerar otros factores al probar la aplicación de la calculadora. Hay un esfuerzo adicional en las pruebas de varios escenarios, como lo bien que la calculadora maneja grandes cantidades. O, ¿qué pasa si el usuario hizo algo inusual como pegar caracteres especiales en el campo de texto?



Buenas prácticas de software



Técnicas de cobertura de pruebas.

Cobertura de riesgos.

Consiste en evaluar los riesgos involucrados en una aplicación y probarlos en detalle.

Es necesario enumerar todos los posibles riesgos que pueden ocurrir en la aplicación y verificarlos adecuadamente.

Por ejemplo, en una aplicación de entrega de alimentos, el usuario elige el restaurante y las preferencias de comida y paga a través de la pasarela de pago integrada. Un riesgo común es que los usuarios se desconecten cuando están en el proceso de pago. ¿Cómo se comportará la aplicación bajo este escenario? Asimismo, se deben considerar otros factores de riesgo relevantes que intervienen en la aplicación y comprobarlos.



Buenas prácticas de software



Técnicas de cobertura de pruebas.

Cobertura de valor límite.

El análisis del valor límite es una parte de las pruebas de software que permite al tester crear los casos de prueba necesarios para un campo de entrada.

Por ejemplo, un campo de entrada numérico solo debe permitir valores del 0 al 50.

Por lo tanto, puede probar la aplicación ingresando números menores que 0, como negativos, y números mayores a 50; así como verificar que no se permitan escribir caracteres diferentes a números.



Buenas prácticas de software



Técnicas de cobertura de pruebas.

Cobertura de requisitos.

La cobertura de los requisitos es la más crucial de todas las técnicas discutidas en este artículo.

Por supuesto, incluso si su aplicación está libre de errores y funciona bien, ¿qué pasa si no cumple con los requisitos del usuario? Esto no solo pone en riesgo al proyecto, sino que afecta fuertemente la imagen de la empresa.

Por eso el llamado a tener un plan de pruebas que valide que la aplicación cumpla con todos los requisitos. Después de todo, esa es toda la lógica detrás del desarrollo de software.



Buenas prácticas de software



Importancia de la cobertura en las pruebas de software.

Encontrar y rectificar errores en etapas tempranas.

Puede encontrar brechas en los requisitos, casos de prueba y defectos en una etapa muy temprana del desarrollo del producto.

Por lo tanto, puede evitar retrasos, reelaboraciones o reprocesos y ayudar a ofrecer un producto de alta calidad.

Crear más casos de prueba para garantizar una mejor cobertura.

La cobertura de pruebas permite crear más casos de prueba, lo que anima a tener una cobertura premium.

Eso, a su vez, conduce a menos defectos y menos trabajo en etapas posteriores. Sobre todo, puede mejorar la experiencia de sus clientes con un producto de mayor calidad.



Buenas prácticas de software



Importancia de la cobertura en las pruebas de software.

Eliminar casos de prueba no deseados.

La cobertura de pruebas permite identificar los casos de prueba innecesarios, que no son relevantes, para su proyecto actual.

Por lo tanto, sus desarrolladores pueden eliminar esos casos de prueba y hacer que el código general sea más ligero.

Posea un mejor control sobre su proyecto.

La cobertura de pruebas permite ahorrar tiempo y reducir costos, ya que puede encontrar y corregir defectos antes y más rápido.

En última instancia, puede tener control directo sobre el proyecto.



Buenas prácticas de software



Importancia de la cobertura en las pruebas de software.

Ciclos de prueba eficientes.

Con el análisis de cobertura de pruebas, puede prevenir el ingreso de defectos.

Además, la cobertura de pruebas ayuda en las pruebas de regresión, priorizando los casos de prueba, aumentando y minimizando los conjuntos de pruebas.

Todo esto conducirá a ciclos de prueba suaves, impecables y eficientes.

Mayor ROI.

La cobertura de pruebas tendrá un impacto significativo en el ROI (Retorno de la inversión), ya que, al permitir identificar defectos temprano, se tendrá muy pocos errores en las etapas de producción.



Buenas prácticas de software



Unidad 4. Buenas prácticas para los modelos de prueba de software.



Buenas prácticas de software



4.5 Utilizar herramientas integradas para las pruebas.



Buenas prácticas de software



4.5 Utilizar herramientas integradas para las pruebas.

Hoy en día existen herramientas para realizar las pruebas, se recomienda según el proyecto considerar las de *open source*.

Ya que ofrecen flexibilidad y a la fecha han alcanzado un desarrollo y una madurez que cubre la necesidad de automatizar el proceso de pruebas del software.

A continuación, se da una lista de herramientas, que dan buenos resultados en el proceso de pruebas del software.



Buenas prácticas de software



4.5.1 Herramientas abiertas para pruebas.

Herramientas Open Source:

1) Herramientas de gestión de pruebas

Bugzilla Testopia.

FitNesse

qaManager

qaBook

RTH (open source)

Salome-tmf

Squash TM

Test Environment toolkit

TestLink

Testitool

Xqual Studio

Radi-testdir

Data Generator



Buenas prácticas de software



4.5.1 Herramientas abiertas para pruebas.

Herramientas Open Source:

2) Herramientas para pruebas funcionales.

Selenium

Soapui

Watir

Capedit

Canoo WebTest

Solex

Imprimatur

SAMIE

ITP

WET

WebInject



Buenas prácticas de software



4.5.1 Herramientas abiertas para pruebas.

Herramientas Open Source:

3) Herramientas para pruebas de carga y rendimiento.

- FunkLoad
- FWPTT load testing
- loadUI
- Jmeter



Buenas prácticas de software



4.5.2 Herramientas de pago para pruebas.

Herramientas Comerciales:

1) Herramientas de gestión de pruebas.

HP Quality Center/ALM

QA Complete

qaBook

T-Plan Profesional

SMARTS

QAS.Test Case Studio

PractiTest

SpiraTest

TestLog

ApTest Manager

Zephyr



Buenas prácticas de software



4.5.2 Herramientas de pago para pruebas.

Herramientas Comerciales:

2) Herramientas para las pruebas funcionales.

- QuickTest Pro
- Rational Robot
- Sahi
- SoapTest
- Test Complete
- QA Wizard
- Squish
- vTest
- Internet Macros



Buenas prácticas de software



4.5.2 Herramientas de pago para pruebas.

Herramientas Comerciales:

3) Herramientas para las pruebas de carga y rendimiento.

- HP LoadRunner
- LoadStorm
- NeoLoad
- WebLOAD Profesional
- Forecst
- ANTS – Advanced.NET Testing System
- Webserver Stress Tool
- Load Impact



Buenas prácticas de software



Unidad 4. Buenas prácticas para los modelos de prueba de software.



Buenas prácticas de software



4.6 Llevar a cabo el control de cambios y versiones.



Buenas prácticas de software



4.6 Llevar a cabo el control de cambios y versiones.

Una buena práctica en el desarrollo de software es llevar a cabo el control de versiones.

El control de versiones requiere de rastrear y gestionar los cambios en el código de software.

Para llevar a cabo el control de versiones actualmente se cuenta con herramientas o sistemas de control de versiones.

Las herramientas o sistemas de control de versiones son software que ayudan en la gestión de los cambios del código que el equipo de desarrollo va realizando.

Las herramientas de gestión de cambios apoyan de manera significativa en los desarrollos rápido al llevar a cabo el control de las versiones lo que permite que el equipo de desarrollo trabaje más rápido logrando implementaciones de manera exitosa.



Buenas prácticas de software



En el control de versiones se realiza un seguimiento de todas las modificaciones en el código.

Se pueden comparar las versiones del código, lo que ayuda a detectar y resolver errores.

Se minimizan las interrupciones durante el desarrollo.

Se debe tomar en cuenta que para todo proyecto de software el código fuente es muy por lo tanto se debe proteger.

Un componente, proceso o función de un proyecto se conforman de código y se organizan en carpetas con una estructura de árbol.



Buenas prácticas de software



El trabajo de varios desarrolladores no se ve afectado pues mientras unos trabajan en desarrollos de funciones nuevas otros solucionan errores, cada uno trabaja en las ramas de carpetas del árbol sin provocar conflictos.

El control de versiones realiza un seguimiento de todos los cambios individuales y del equipo.

Por lo que evita que el trabajo concurrente entre en conflicto.

Se recomienda hacer las pruebas cuando se concluya el desarrollo de módulos, funciones o procesos, hasta concluir la versión final.



Buenas prácticas de software



Los cambios de versiones deben incluir el autor, la fecha y notas escritas sobre el propósito de cada cambio.

Con lo anterior se logra volver a las versiones anteriores ayudando a analizar la causa de los cambios en la raíz identificando la fuente de los errores.

El trabajo cotidiano de programación del software activa siempre una versión que se continua en la siguiente sesión de trabajo.



Buenas prácticas de software



4.6.1 Herramientas para el control de cambios y versiones.

a) Sistemas de control de versiones locales.

Cuando se trabaja de manera individual generalmente el control de versiones y cambios se lleva a cabo haciendo una copia de la carpeta de desarrollo en el computador personal o local indicando la fecha de modificación para identificar la última versión guardada.

b) Sistemas de control de versiones centralizados.

Cuando se trabaja en proyectos grandes y con un equipo de desarrollo es necesario poder contar con un sistema colaborador de control de versiones centralizado.



Buenas prácticas de software



4.6.1 Herramientas para el control de cambios y versiones.

c) Sistemas de control de versiones distribuidas.

En este sistema cada desarrollador contiene una copia completa del proyecto de forma local y en su conjunto cualquier repositorio permite compartir información.



Buenas prácticas de software



4.6.1 Herramientas para el control de cambios y versiones.

Los sistemas de control de versiones (VCS) se conocen como herramientas de SCM (Gestión del código fuente) o RCS (Sistema de control de revisiones).

Una de las herramientas de VCS más utilizadas hoy en día es Git.

Git es un VCS distribuido de código abierto por lo tanto gratuito.

Git guarda un snapshot (instantánea) del estado de cada archivo.

Si el archivo no ha cambiado, no crea una nueva copia, sino que crea una referencia al archivo original.



Buenas prácticas de software



4.6.1 Herramientas para el control de cambios y versiones.

Con Hit se logra:

Auditoría completa del código, identificando en todo momento quién acceso al código.

Permite visualizar el control de los cambios del proyecto.

Permite regresar a los desarrollos anteriores de forma rápida.

El control de versiones lo realiza por medio de etiquetas.

La seguridad es una característica fundamental y Hit la tiene ya que todas las estructuras internas de datos irán cifradas con el algoritmo SHA1.