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

4.4 Medir el costo de las pruebas

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 preba del softwarwe, alguna línea del código no se ejecuta, entonces dicha línea no está cubierta.

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

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

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.

Cobertura de riesgos.

Se determinan los riesgos que involucran el sistema y se deben a detalle.

Técnicas de cobertura de pruebas.

Cobertura de valor límite.

Se analizan los valor tanto límites inferiores como superiores de cada una de las pruebas del sistema que permite al tester realizarlas.

Cobertura de requisitos.

Las pruebas de los sistemas siempre van encaminadas a validar que los requisitos iniciales del sistema se cumplen, por lo que este tipo de pruebas es fundamental y tiene vital importancia al final del desarrollo del sistema.

4.5 Utilizar herramientas integradas para las pruebas.

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.

4.5.1 Herramientas abiertas para pruebas

A continuación se presenta un alista de herramientas open source las cuales puedes consultar en su página web.

Herramientas Open Source:

1) Herramientas de gestión de pruebas, indaga sobre sus características.

Bugzilla Testopia.
FitNesse
qaManager

4.5.1 Herramientas abiertas para pruebas

Herramientas Open Source:

2) Herramientas para pruebas funcionales , indaga sobre sus características.

- Selenium
- Soapui
- Watir

4.5.1 Herramientas abiertas para pruebas

Herramientas Open Source:

3) Herramientas para pruebas de carga y rendiniento , indaga sobre sus características.

FunkLoad
FWPTT load testing
loadUI
Jmeter

4.5.2 Herramientas de pago para pruebas

Herramientas Comerciales:

1) Herramientas de gestión de pruebas , indaga sobre sus características.

HP Quality Center/ALM
QA Complete
qaBook

4.5.2 Herramientas de pago para pruebas

Herramientas Comerciales:

2) Herramientas para las pruebas funcionales , indaga sobre sus características.

QuickTest Pro
Rational Robot
Sahi

4.5.2 Herramientas de pago para pruebas

Herramientas Comerciales:

3) Herramientas para las pruebas de carga y rendimiento , indaga sobre sus características.

HP LoadRunner
LoadStorm
NeoLoad

4.6 Llevar a cabo el control de cambios y versiones.

4.6 Llevar a cabo el control de cambios y versiones.

Una buena práctica en el desarrollo de software es llevar a cabo el la gestión de las versiones.

El objetivo es gestionar los cambios del código de software.

La gestión puede llevarse a cabo con el uso de herramientas o sistemas.

Los software automatizados para la gestión de versiones y cambios apoyan de manera significativa en los desarrollos rápido al controlando los de cada una de las versiones, lo que facilita al grupo de desarrolladores a trabajer más rápido logrando implementaciones de manera éxitosa.

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

Se minimizan las interrupciones durante el desarrollo.

Buenas prácticas de software

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 orgnizan en carpetas con una estructura de árbol.

El trabajo de varios desarrolladores no se ve afectado pues mientras unos trabajan en desarollos de funcione snuevas otros solucionan errores, cada uno trabaja en las ramas de carpetas del árbol sin provocar conflictos. Se recomienda hacer las pruebas cuando se concluya el desarrollo de módulos, funciones o procesos, hasta concluir la versión final.

Los cambios de versiones incluyen al autor así como la fecha de realización y no olvidar hacer anotaciones que dan origen al cambio.

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

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 distribuidas.

En este sistema cada desarrollador tiene una copia completa del proyecto y en conjunto cualquier integrante del equipo accesa al repositorio para compartir información.
Git es de código abierto por lo tanto gratuito y es distribuido.
Con Hit se logra:
Para conocer quien acceso al código se lleva a cabo una auditoría a los registros y a la plataforma.
Permite visualizar el control de los cambios del proyecto.
Permite regresar a los desarrollos anteriores.
El control de las versiones del código, lo realiza utilizando etiquetas.
La seguridad en el uso del código es una característica fundamental y Hit la tiene dado que se cifran con algoritmos las estructuras de los datos.