

# 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.

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



# 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?



# 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.



# 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.

## 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.



# 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.



# 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.


# 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

**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 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 vesiones 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 realziando.

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

# 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 orgnizan 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 desarollos de funcione snuevas 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 idntificnado 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.



# 4.6.1 Herramientas para el control de cambios y versiones.

**a) Sistemas de control de versiones locales.**

Cuano 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 proeyctos grandes y con un equipo de desarrollo es necesario poder contar con un sistema colaborador de control de versiones centralizado.

# 4.6.1 Herramientas para el control de cambios y versiones.

**c) Sistemas de control de versiones distribuidas.**

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

# 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 de código abierto por lo tanto gratuito y es distribuido.

# 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.