Curso de QA & Software Testing

Parte IV

Capacitador

Ronnel Vélez Manzano
Gerente de QA & SW Testing
Miembro del HASTQB

Certificaciones: SCRUM MASTER
ISTQB | CTFL | Certified Tester Foundation Level
ISTQB | CTAL-TM | Certified Tester Advanced Level - Test Manager
ISTQB | CTFL- AT | Certified Agile Tester

E-mails: rvelez@crnova.com
rvelez@hastqb.org
ronnel.velez@gmail.com

Servicios Computacionales Novacomp
San Jose – Costa Rica
Web: http://www.crnova.com
Teléfono (Costa Rica): (506) 2216-5800
Fax: (506) 2216-5900

Instrucciones

El presente material de formación ha sido elaborado por Servicios Computacionales Novacomp. Está basado en el programa de estudios de:

- IREB: International Requirements Engineering Board
- ISTQB: International Software Testing Qualifications Board
- Manifesto for Agile Software Development
- Scrum Alliance

Las marcas contenidas e incorporadas en los documentos son propiedad de sus propietarios respectivos incluso si no son mencionados de forma explícita.

Técnicas de diseño de pruebas

- Proceso de desarrollo de prueba
- Categorías de las técnicas de diseño de prueba
- Técnicas basadas en la especificación o de caja negra
- **Técnicas basadas en la estructura o de caja blanca**
- **Técnicas basadas en la experiencia**
- **Selección de las técnicas de prueba**

VI. Técnicas de diseño de pruebas

04. Técnicas basadas en la estructura o de caja blanca

Técnicas basadas en caja blanca ("white box") (1/3)

El probador conoce la estructura interna del programa / código

- Es decir, la jerarquía de los componentes, flujo de control, flujo de datos, etc.

Las casos de prueba son seleccionados en base a la estructura interna del programa / código

- A lo largo de las pruebas es posible que se interfiera con la ejecución de otro tipo de prueba

¡La estructura del programa es el foco de atención!

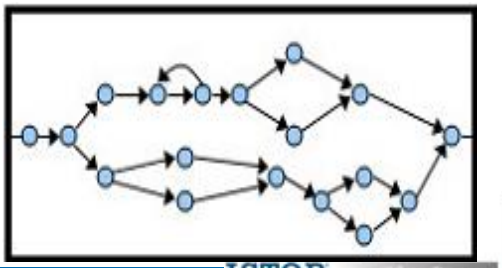
- La técnica de caja blanca también es conocida como **prueba basada en la estructura** o **prueba basada en el flujo de control**

Diseño de caso de prueba

Los casos de prueba están basados en la estructura del programa

El proceso de pruebas es controlado externamente

Análisis del flujo de control dentro del objeto de prueba a lo largo de la ejecución de pruebas



VI. Técnicas de diseño de pruebas

04. Técnicas basadas en la estructura o de caja blanca

Técnicas basadas en caja blanca ("white box") (2/3)

En el presente apartado se explicarán en detalle las siguientes técnicas de caja blanca ("white box"):

- **Pruebas de sentencia y cobertura** ("statement testing and coverage").
- **Pruebas de decisión y cobertura** ("decision testing and coverage").
- **Pruebas de condición y cobertura** ("condition testing and coverage").

Observación:

Estas técnicas representan las técnicas más importantes y utilizadas frecuentemente. Estas técnicas están relacionadas con las técnicas de análisis estático, las cuales forman parte de las técnicas estáticas.

VI. Técnicas de diseño de pruebas

04. Técnicas basadas en la estructura o de caja blanca

Técnicas basadas en caja blanca ("white box") (3/3)

Está basada en la estructura identificada del software o del sistema y es aplicada normalmente en los siguientes niveles de prueba:

- **Nivel de componente:** la estructura de un componente software, es decir sentencias, decisiones, ramas o distintos caminos.
- **Nivel de integración:** la estructura puede ser un árbol de llamada (un diagrama en el cual unos módulos llaman a otros módulos o interfaces).
- **Nivel de sistema:** la estructura puede ser una estructura de un menú, un proceso de negocio o la estructura de una página web.

Tres técnicas de diseño de pruebas estructurales relacionadas al código para cobertura del código basadas en sentencias, decisiones y condiciones.

VI. Técnicas de diseño de pruebas

04. Técnicas basadas en la estructura o de caja blanca

Herramientas para pruebas en caja blanca ("white box") (1/2)

Durante las pruebas de caja blanca, el programa objeto de las pruebas es ejecutado de la misma forma que las pruebas de caja negra. Ambas categorías (caja blanca y caja negra) conforman las pruebas dinámicas

- La teoría establece que todas las partes de un programa deberían ser **ejecutadas por lo menos una vez** durante las pruebas

El grado de cobertura de un programa se mide con el uso de herramientas (por ejemplo, analizadores de cobertura):

- La **instrumentación del código** se lleva a cabo con el objeto de contar la ejecución de caminos, es decir se insertan contadores en el código del programa del objeto de prueba

VI. Técnicas de diseño de pruebas

04. Técnicas basadas en la estructura o de caja blanca

Herramientas para pruebas en caja blanca ("white box") (2/2)

Las técnicas de caja blanca requieren el apoyo de herramientas:

- **Especificación de caso de prueba**
  - Generación automática del diagrama del flujo de control a partir del código fuente del programa
- **Ejecución de la prueba**
  - Herramientas para monitorizar y controlar el flujo del programa dentro del objeto de prueba

El soporte de herramientas asegura la calidad de las pruebas e incrementa su eficiencia

- Dada la complejidad de las mediciones necesarias para las pruebas de caja blanca, la ejecución manual de pruebas implica:
  - **Consumo de tiempo, consumo de recursos**
  - **Dificultad en la implementación y propensión a errores**

VI. Técnicas de diseño de pruebas

04. Técnicas basadas en la estructura o de caja blanca

Principales tipos de cobertura

- **Cobertura de sentencia ("statement coverage")**
  - Porcentaje de sentencias ejecutables que han sido practicadas por los casos de prueba
  - También puede ser aplicado a módulos, clases, elementos de un menú, etc.
- **Cobertura de decisión ("decision coverage")**
  - Porcentaje de resultados de decisión que han sido practicados por los casos de prueba.
- **Cobertura de condición ("condition coverage")**
  - Porcentaje de todos los resultados individuales de condición que afectan de forma independiente al resultado de una decisión que ha sido practicada por casos de prueba.
  - La cobertura de condición se presenta en distintos grados, por ejemplo, cobertura de condición simple y cobertura de condición múltiple.

VI. Técnicas de diseño de pruebas

04. Técnicas basadas en la estructura o de caja blanca

Cobertura de sentencia ("statement coverage") (1/5)

El foco de la atención es la sentencia del código de un programa

- ¿Qué casos de prueba son necesarios con el objeto de ejecutar todas (o un porcentaje determinado) las sentencias del código existentes?

La base de este análisis es el gráfico (o diagrama) de flujo de control ("control flow graph")

- Todas las instrucciones están representadas por nodos y el flujo de control entre instrucciones está representado por una arista (flecha)
- Las instrucciones múltiples se combinan en un nodo independiente si solamente pueden ser ejecutados en una secuencia particular

El objetivo de la prueba (criterio de salida) es lograr la cobertura de un porcentaje específico de todas las sentencias, denominado cobertura de sentencia. (C_0 cobertura de código – "code coverage")

$$\text{Cobertura de sentencia } (C_0) = \frac{\text{Número de sentencias ejecutadas}}{\text{Número total de sentencias}} * 100\%$$

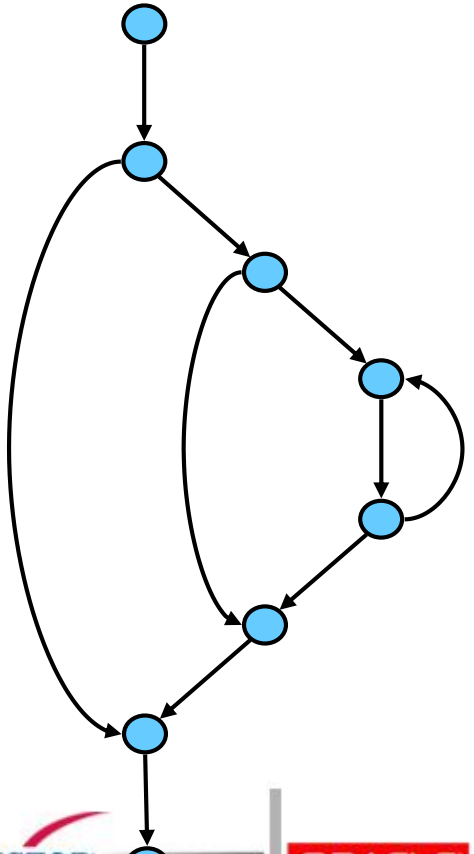
VI. Técnicas de diseño de pruebas

04. Técnicas basadas en la estructura o de caja blanca

Cobertura de sentencia ("statement coverage") (2/5)

Ejemplo A: 1/2

Se evalúa el siguiente segmento de código de un programa, que está representado por el diagrama del flujo de control (imagen a la derecha)



VI. Técnicas de diseño de pruebas

04. Técnicas basadas en la estructura o de caja blanca

Cobertura de sentencia ("statement coverage") (3/5)

Ejemplo A: 2/2

Considerar el programa representado por el gráfico (o diagrama) de flujo de control (imagen a la derecha).

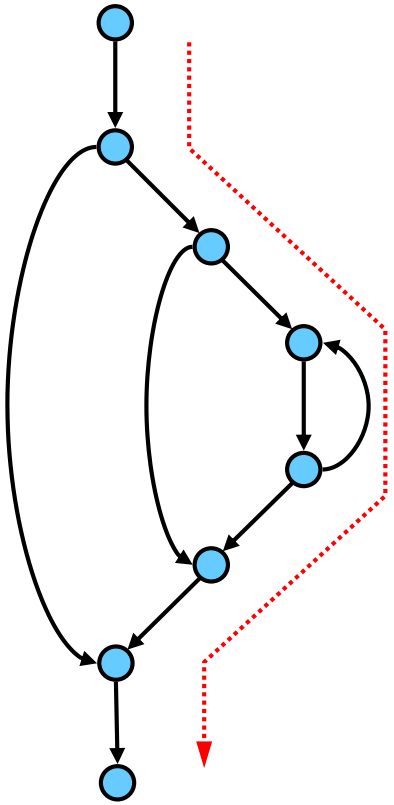
- Contiene dos sentencias "if" y un bucle "do-while" dentro de la segunda sentencia "if".

Hay tres "caminos" diferentes a través del segmento de programa.

- La primera sentencia "if" permite dos direcciones.
- La dirección de la derecha de la primera sentencia "if" se divide nuevamente a partir de una segunda sentencia "if".

Todas las sentencias de este programa pueden ser alcanzadas haciendo uso de este camino a la derecha.

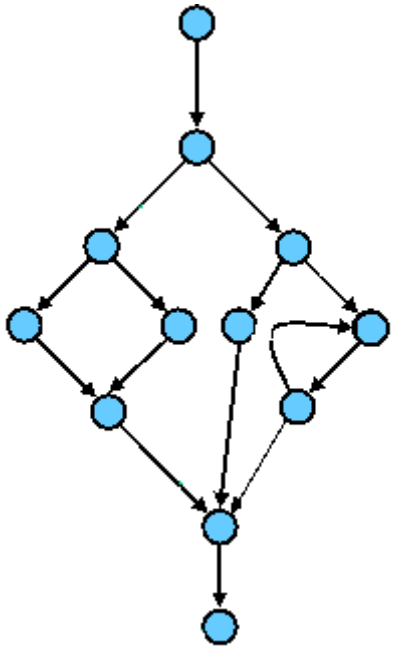
- Un solo caso de prueba será suficiente para alcanzar el 100% de cobertura de sentencia.



Cobertura de sentencia ("statement coverage") (4/5)

Ejemplo B:

En este ejemplo el gráfico ("diagrama") es ligeramente más complejo:



VI. Técnicas de diseño de pruebas

04. Técnicas basadas en la estructura o de caja blanca

Cobertura de sentencia ("statement coverage") (4/5)

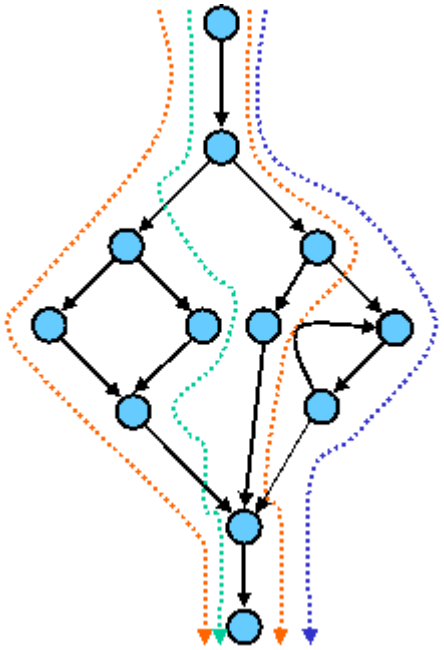
Ejemplo B:

En este ejemplo el gráfico ("diagrama") es ligeramente más complejo:

- El programa contiene las sentencias "if" y un bucle (dentro de una sentencia "if")

Cuatro caminos diferentes conducen a través de este segmento de programa

- La primera sentencia "if " permite dos direcciones
- En cada rama de la sentencia "if" otra sentencia "if" permite nuevamente dos direcciones diferentes.
- Para una cobertura de sentencia del 50% hacen falta **2 (dos)** casos de prueba.



VI. Técnicas de diseño de pruebas

04. Técnicas basadas en la estructura o de caja blanca

Cobertura de sentencia ("statement coverage") (5/5)

Puntos resaltantes sobre este tipo de técnica:

Será detectado el código muerto, es decir, código constituido por sentencias que nunca se ejecutan.

- Si hay código muerto en el programa, no se podrá lograr una cobertura del 100%

No podrán ser detectadas instrucciones faltantes, es decir, código que es necesario con el objeto de cumplir con la especificación.

- Estas pruebas solo buscan ¿si todo el código puede ser alcanzado/ejecutado?
- El código faltante no puede ser detectado utilizando técnicas de caja blanca (análisis de cobertura)

VI. Técnicas de diseño de pruebas

04. Técnicas basadas en la estructura o de caja blanca

Cobertura de decisión ("decision coverage") (1/5)

En lugar de las sentencias, la cobertura de decisión se centra en el flujo de control en un segmento de programa (no los nodos sino las aristas del diagrama de flujo de control)

- Todas las aristas del diagrama de flujo de control tienen que ser cubiertas al menos una vez
- ¿Qué casos de prueba son necesarios para cubrir cada arista del diagrama de flujo de control al menos una vez?

El propósito de esta prueba (criterio de salida) es lograr la cobertura de un porcentaje específico de todas las decisiones, denominado cobertura de decisión (C_1 cobertura de decisión - "decision coverage"; cobertura de rama – "branch coverage")

$$\text{Cobertura de decisión}(\ C_1) = \frac{\text{Número de decisiones ejecutadas}}{\text{Número total de decisiones}} * 100\%$$

VI. Técnicas de diseño de pruebas

04. Técnicas basadas en la estructura o de caja blanca

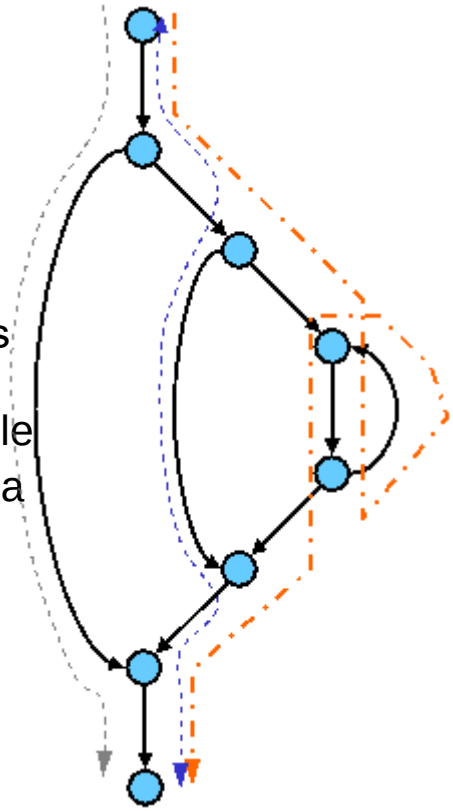
Cobertura de decisión ("decision coverage") (2/5)

Ejemplo C:

El gráfico ("diagrama") de flujo de control (imagen a la derecha) representa el segmento de un programa objeto de la evaluación

Tres caminos diferentes conducen a través del diagrama de este segmento de programa:

- La primera sentencia "if" conduce a dos direcciones diferentes
- Un camino de la primera sentencia "if" se divide nuevamente en dos caminos diferentes, uno de los cuales contiene un bucle
- Solamente se puede alcanzar todas las aristas a través de una combinación de los tres caminos posibles
- Son necesarios tres casos de prueba para alcanzar una cobertura de decisión del 100%
- Utilizando solamente las dos direcciones de la derecha, pueden ser cubiertas nueve de diez aristas (C_1 = 90%)

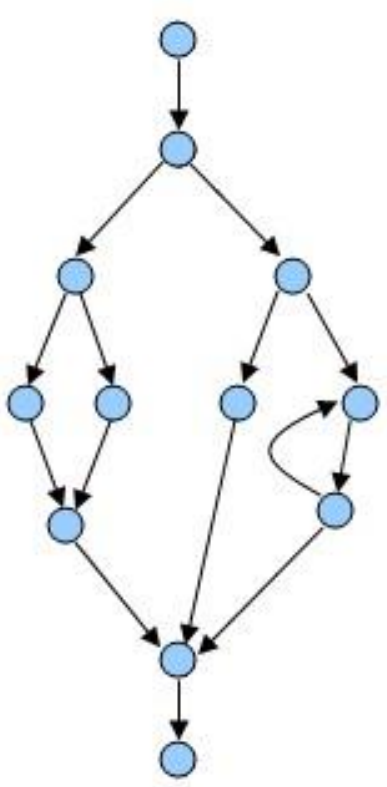


VI. Técnicas de diseño de pruebas

04. Técnicas basadas en la estructura o de caja blanca

Cobertura de decisión (“decision coverage”) (3/5)

Ejemplo D:

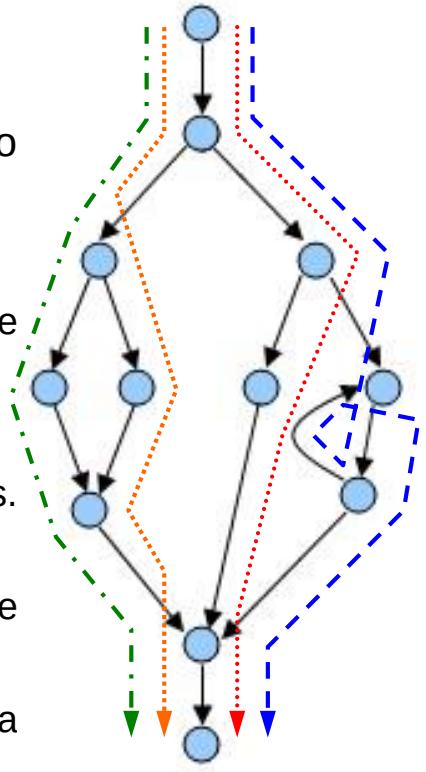


VI. Técnicas de diseño de pruebas

04. Técnicas basadas en la estructura o de caja blanca

Cobertura de decisión ("decision coverage") (3/5)

Ejemplo D:



En este ejemplo el gráfico ("diagrama") es ligeramente más complejo. Cuatro "caminos" diferentes conducen a través del segmento de programa:

- La primera sentencia "if" permite dos direcciones
- En ambas ramas de la sentencia "if" otra sentencia "if" permite nuevamente dos direcciones diferentes
- En este ejemplo, el bucle no se cuenta como una decisión adicional
- Utilizando sólo un caso de prueba, pueden ser cubiertas 7 de 15 aristas. Esto resulta en un valor de C_1=46,67%
- Son necesarios cuatro casos de prueba para lograr una cobertura de decisión del 100%
- ¡En este ejemplo, son necesarios el mismo conjunto de casos de prueba para lograr una cobertura de sentencia del 100%!

VI. Técnicas de diseño de pruebas

04. Técnicas basadas en la estructura o de caja blanca

Cobertura de decisión (“decision coverage”) (4/5)

Puntos resaltantes sobre este tipo de técnica:

Lograr una cobertura de decisión del 100% requiere, al menos, los mismos casos de prueba que requiere la cobertura de sentencia con tendencia a una mayor cantidad en la mayoría de los casos

- Una cobertura de decisión del 100% siempre incluye una cobertura de sentencia del 100%

La mayoría de las aristas son cubiertas en múltiples ocasiones

Desventajas

- No se pueden detectar sentencias faltantes
- No es suficiente para probar condiciones complejas
- No es suficiente para probar bucles de forma extensiva
- No se consideran las dependencias entre bucles

VI. Técnicas de diseño de pruebas

04. Técnicas basadas en la estructura o de caja blanca

Cobertura de decisión ("decision coverage") (5/5)

Pruebas de sentencia Vs Pruebas de decisión

Ambos métodos se refieren a **caminos a través** del **diagrama de flujo de control**

- **Difieren** en la **cantidad de casos de prueba** necesarios para lograr el 100% de cobertura

Sólo se considera el resultado final de una condición, a pesar de que la condición resultante puede estar constituida por múltiples condiciones atómicas

- El camino (del programa) a ejecutar depende solamente del resultado final de la condición combinada
- Aquellos fallos debidos a una implementación errónea de las partes de una decisión combinada pueden no ser detectados

VI. Técnicas de diseño de pruebas

04. Técnicas basadas en la estructura o de caja blanca

Cobertura de condición ("condition coverage") (1/4)

Se tiene en cuenta la complejidad de una condición que esté constituida por múltiples condiciones atómicas

- Una condición atómica no puede ser dividida en sentencias condicionales más pequeñas

Éste método tiene por objetivo detectar defectos que resulten de la implementación de condiciones múltiples (condiciones combinadas)

- Las condiciones múltiples están constituidas por condiciones atómicas, que se combinan con el uso de operadores lógicos como: OR, AND, XOR, etc.
  - Ejemplo: ((a>2) OR (b<6))
- Las condiciones atómicas no contienen operadores lógicos, sólo contienen operadores relacionales y el operador NOT (=, >, <, etc.).

Cobertura de condición (“condition coverage”) (1/4)

Hay dos tipos de cobertura de condición.

- Cobertura de condición simple (“simple condition coverage”).
- Cobertura de condición múltiple (“multiple condition coverage”).

VI. Técnicas de diseño de pruebas

04. Técnicas basadas en la estructura o de caja blanca

Cobertura de condición ("condition coverage") (2/4)

Cobertura de condición simple ("simple condition coverage")

Cada sub-condición atómica de una sentencia condicional combinada tiene que tomar, al menos una vez, los valores lógicos verdadero ("true) así como falso ("false")

Considerar la siguiente condición:

X > 4 OR Y < 8

Los casos de prueba para la cobertura de condición simple, podrían ser, por ejemplo:

X =7 (true)	Y =10 (false)	X>4 OR Y<8 (true)
X=2 (false)	Y =5 (true)	X>4 OR Y<8 (true)

- Este ejemplo se utiliza para explicar la cobertura de condición utilizando una expresión con una condición múltiple
- Con sólo dos casos de prueba se puede lograr una cobertura de condición simple
  - Cada sub-condición ha tomado los valores verdadero ("true) y falso ("false")
- Sin embargo, el resultado combinado es verdadero ("true")en ambos casos
  - **true OR false = true**
  - **false OR true = true**

VI. Técnicas de diseño de pruebas

04. Técnicas basadas en la estructura o de caja blanca

Cobertura de condición ("condition coverage") (3/4)

Cobertura de condición múltiple ("multiple condition coverage")

Todas las combinaciones que puedan ser creadas utilizando permutaciones de las sub condiciones atómicas deben formar parte de las pruebas

Considerar la siguiente condición:

X > 3 OR Y < 7

Los casos de prueba para la cobertura de condición múltiple, podrían ser, por ejemplo:

X=6 (true)	Y=9 (false)	X>3 OR Y<7 (true)
X=8 (true)	Y=5 (true)	X>3 OR Y<7 (true)
X=2 (false)	Y=6 (true)	X>3 OR Y<7 (true)
X=1 (false)	Y=8 (false)	X>3 OR Y<7 (false)

- Este ejemplo se utiliza para explicar la cobertura de condición utilizando una expresión con una condición múltiple
- Con cuatro casos de prueba se puede lograr una cobertura de condición múltiple
  - **Se han creado todas las combinaciones de los valores verdadero ("true") y falso ("false")**
  - **Se han logrado todos los posibles resultados de la condición múltiple**
- El número de caso de prueba se incrementa de forma potencial:
  - **n = número de condiciones atómicas**
  - **2^n = número de casos de prueba**

VI. Técnicas de diseño de pruebas

04. Técnicas basadas en la estructura o de caja blanca

Cobertura de condición ("condition coverage") (4/4)

Puntos resaltantes sobre este tipo de técnica:

La **cobertura de condición simple** es un instrumento débil para probar condiciones múltiples.
La **cobertura de condición múltiple** es un método mucho más efectivo.

- Asegura cobertura de sentencia y decisión
- Sin embargo, tiene como resultado un alto número de casos de prueba: 2^n
- La ejecución de algunas combinaciones no es posible
- Por ejemplo "*x > 5 AND x < 10*" ambas sub condiciones no pueden ser falsas al mismo tiempo.

VI. Técnicas de diseño de pruebas

05. Técnicas basadas en la experiencia

Fundamentos

- Las pruebas basadas en la experiencia también se denominan **pruebas intuitivas (“intuitive testing ”)** e incluyen: predicción de errores (“error guessing”) (pruebas orientadas a puntos débiles) y pruebas exploratorias (pruebas iterativas basadas en el conocimiento adquirido respecto del sistema)
- Principalmente aplicadas con el objeto de **complementar** otros **casos de prueba generados** con un mayor **formalismo**
- No cumple los criterios de un proceso de prueba **sistemático**
- Frecuentemente produce **casos de prueba adicionales** que no podrían ser creados con otras prácticas, por ejemplo:
  - Pruebas de un año bisiesto posterior al año 2060
    (Problemas conocidos del pasado)
  - Conjuntos vacíos en valores de entrada
    (Una aplicación similar ha tenido errores en estas circunstancias)

VI. Técnicas de diseño de pruebas

05. Técnicas basadas en la experiencia

Predicción de errores

- **Procedimiento:**
- Comprobar lista de defectos
  - Enumerar posibles errores
  - Factores **ponderados** dependientes del riesgo y probabilidad de ocurrencia
- Diseño de caso de prueba
  - Creación de casos de prueba dirigidos a reproducir los defectos de la lista
  - **Aumentar la prioridad** a aquellos casos de prueba considerando el **valor de su riesgo**
- Actualizar la lista de defectos durante las pruebas
  - Procedimiento iterativo
  - Es útil una **colección** estructurada **de experiencias** cuando se repite el procedimiento en futuros proyectos

VI. Técnicas de diseño de pruebas

05. Técnicas basadas en la experiencia

Pruebas exploratorias ("exploratory testing")

- Es un procedimiento de diseño de casos de prueba especialmente apropiado cuando la **información base** se encuentra **poco** estructurada
- También es útil cuando el **tiempo** disponible para pruebas es **escaso**
- **Procedimiento:**
  - **Revisar** las partes constituyentes (individuales/identificables) del objeto de prueba
  - **Ejecutar** un número reducido de **casos de prueba**, exclusivamente sobre aquellas partes que deben ser probadas, aplicando predicción de errores ("error guessing")
  - **Analizar los resultados**, desarrollar un modelo preliminar ("rough model") de cómo funciona el objeto de prueba
  - **Iteración:** Diseñar nuevos objetos de prueba aplicando el conocimiento adquirido recientemente
  - Por lo tanto concentrándose en las áreas relevantes y explorando características adicionales del objeto de prueba
  - Las herramientas de captura pueden ser útiles para registrar las actividades de prueba

VI. Técnicas de diseño de pruebas

05. Técnicas basadas en la experiencia

Beneficios

- Técnica fácil de implementar hasta para probadores novatos.
- Las técnicas basadas en la experiencia complementan las técnicas sistemáticas para determinar casos de prueba
- Las técnicas basadas en la experiencia dependen en gran medida de la habilidad individual del probador
- La predicción de errores y las pruebas exploratorias son dos de las técnicas más ampliamente utilizadas de pruebas basadas en la experiencia

VI. Técnicas de diseño de pruebas

06. Selección de las técnicas de prueba

Criterios para seleccionar el diseño apropiado de caso de prueba (1)

- Estado de la **información** del **objeto de prueba**
  - ¿Se pueden realizar pruebas de **caja blanca** de alguna manera (**código fuente** disponible)?
  - ¿Hay suficiente material de **especificación** para definir pruebas de **caja negra**, o son necesarias pruebas exploratorias para comenzar?
- Objetivos de prueba predominantes
  - ¿Las pruebas funcionales han sido **solicitadas de forma explícita**?
  - ¿Qué pruebas no funcionales son necesarias?
  - ¿Son necesarias pruebas estructurales para lograr los objetivos del proceso de prueba?
- Riesgos
  - ¿Se espera un **daño/perjuicio serio** proveniente de defectos ocultos?
  - ¿Es muy alta la **frecuencia de uso** del objeto de prueba?
  - ¿Hay algún **estándar legal o contractual**, respecto de la ejecución de pruebas y cobertura de pruebas, que deba ser cumplido?

VI. Técnicas de diseño de pruebas

06. Selección de las técnicas de prueba

Criterios para seleccionar el diseño apropiado de caso de prueba (2)

- Precondiciones del **proyecto**
  - ¿Cuanto **tiempo** y quien está asignado a las actividades de prueba?
  - ¿Cuál es el **riesgo** de que el proceso de prueba no sea finalizado según la planificación?
  - ¿Qué **métodos de desarrollo** son utilizados?
  - ¿Cuales son los punto débiles del **proceso** asociado al proyecto?
- Características del **objeto de prueba**
  - ¿Que **posibilidades** de prueba ofrece el objeto de prueba?
  - ¿Cuál es la **disponibilidad** del objeto de prueba?
- Requisitos **contractuales** y del cliente
  - ¿Ha habido algún **acuerdo específico** entre el cliente/iniciador del proyecto respecto de los procedimientos de prueba?
  - ¿Qué **documentos** deben ser entregados en el momento del despliegue del sistema?

VI. Técnicas de diseño de pruebas

06. Selección de las técnicas de prueba

Criterios para seleccionar el diseño apropiado de caso de prueba (3)

- Mejor (buena) práctica
  - ¿Qué enfoques han **demostrado ser apropiados** para estructuras similares?
  - ¿Qué **experiencias** han sido adquiridas con qué enfoques en el pasado?
- Niveles de prueba
  - ¿En qué niveles de prueba se deben realizar pruebas?

¡Se deben aplicar criterios adicionales dependiendo de la situación específica!

VI. Técnicas de diseño de pruebas

06. Selección de las técnicas de prueba

Intereses distintos causan enfoques diferentes en el diseño de pruebas

- Interés del jefe de proyecto:
  - Para desarrollar un producto software de la calidad requerida
  - Cumplir las restricciones de tiempo y presupuesto
- Intereses del cliente/iniciador del proyecto:
  - Recibir software de la más alta calidad (funcionalidad, fiabilidad, usabilidad, eficiencia, portabilidad y mantenibilidad)
  - Cumplir las restricciones de tiempo y presupuesto
- Intereses del jefe de prueba:
  - Prueba suficientes e intensivas / despliegue adecuado de las técnicas necesarias desde el punto de vista del proceso de prueba
  - Evaluar/valorar el nivel de calidad alcanzado por el proyecto
  - Asignar y utilizar los recursos planificados para las pruebas de forma óptima