Curso de preparación
Tester Certificado
Nivel Básico

Agenda

- Capítulo 0 Introducción
- Capítulo 1 Fundamentos de Pruebas
- Capítulo 2 Pruebas a lo largo de Vida del desarrollo
- Capítulo 3 Técnicas Estáticas
- Capítulo 4 Técnicas de Diseño de Pruebas
- Capítulo 5 Gestión de Pruebas
- Capítulo 6 Herramientas de Soporte para el Proceso de Pruebas

Agenda

Capítulo 4. Técnicas de diseño de pruebas

- 4.1 El proceso de desarrollo de pruebas
- 4.2 Categorías de las técnicas de diseño de pruebas
- 4.3 Técnicas basadas en la especificación o de caja negra
- 4.4 Técnicas basadas en la estructura o de caja blanca
- 4.5 Técnicas basadas en la experiencia
- 4.6 Escoger las técnicas de prueba

4.1: El proceso de desarrollo de pruebas

Términos

- **Script de pruebas:**

"Test script"Comúnmente usado para referirse a una especificación de procedimiento de prueba, especialmente una automatizada.

- **Trazabilidad:**

"traceability"Capacidad de identificar elementos relacionados en la documentación y el software, tales como requisitos con las pruebas asociadas. Véase también trazabilidad horizontal y trazabilidad vertical.

Los casos de prueba pueden ser creados :

- **Muy Informal:** Con poca o sin ninguna documentación
- **Muy Formal:** Dependiendo del contexto de las pruebas, incluyendo la organización, la madurez de las pruebas y el proceso de desarrollo, restricción del tiempo y las personas involucradas.

Especificación de diseño de pruebas

Especificación de diseño de pruebas: "test design specification" Documento que especifica las **condiciones** de prueba (elementos de cobertura) para el elemento de prueba, el enfoque de pruebas de forma detallada e identifica los casos de prueba de alto nivel asociados. [Según IEEE 829]

El objetivo del documento de especificación, es identificar las características que se van **a probar** y deben ser plasmadas en el diseño de las pruebas.

Estructura definida como estándar en un diseño de prueba

- Identificador
- Características a probar de los elementos del Software
- Detalles del plan de pruebas de donde parten los diseños (técnicas de diseño de casos de pruebas y métodos de análisis de resultados).
- Criterios para que una pruebas sea exitosa o no

Especificación de caso de prueba

Especificación de casos de prueba: "test case specification" Documento que especifica un conjunto de casos de prueba (objetivos, entradas, acciones de prueba, resultados esperados y precondiciones de ejecución) para un elemento de prueba. [Según IEEE 829]

El objetivo del documento de especificación de caso de prueba**, es definir uno a uno los casos de prueba identificado por una especificación del diseño de prueba.**

Se debe tener en cuenta las necesidades del entorno (Hardware, software, recursos) y la dependencias entre casos de pruebas.

Especificación de procedimiento de prueba

El objetivo del documento de procedimiento de prueba, consiste en **especificar los pasos para la ejecución de un conjunto de casos de prueba. Estos se pueden combinar en test suites y escenarios de prueba**

Estructura definida como estándar

- **Identificador** único de la especificación y referencia a las correspondiente especificación de diseño de prueba.
- Objetivo del procedimiento y **lista de casos** que se van a ejecutar.
- **Pasos y la forma de registrar el resultado** y los defectos resultantes de la ejecución, para esto se debe especificar:
  - **La secuencia** necesaria de **acciones** , para preparar la ejecución
  - Acciones necesarias durante **la ejecución y suspensión** de la prueba
  - Quien y cuando deben realizar las pruebas.

- Puntos para reinicio de la ejecución y acciones necesarias.
- Acciones necesarias para restaurar el entorno y tratar los acontecimientos anómalos.
- Tener en cuenta el ambiente, las prioridades, disponibilidad de los recursos

Si las pruebas son ejecutadas con ayuda de **herramientas de ejecución**, la secuencia es especificada en el script de prueba (**procedimiento de prueba automatizado).**

Calendario de ejecución de pruebas:

"test execution schedule" Planificación **para la ejecución de procedimientos** de prueba. Los procedimientos de prueba están incluidos en el calendario de ejecución de pruebas en su contexto y en el orden en el cual deben ser ejecutados.

Incluyendo **los planes automatizados**, quien los va ejecutar y cuando.

Se toma en cuenta factores como **pruebas de regresión, priorización y dependencias técnicas y lógicas**

Objeto de prueba

- Elemento a ser revisado un documento o una pieza de software en el proceso de desarrollo de software.

Condición de prueba

- Elemento o evento de un componente o sistema que debería ser verificado por uno o más casos de prueba, por ejemplo una función, transacción, característica, atributo de calidad o elemento estructural.

Criterios de prueba

- El objeto de prueba debe cumplir los criterios de prueba con el objeto de superar la prueba.

Procedimiento de prueba

- Secuencia de acciones para la ejecución de una prueba.

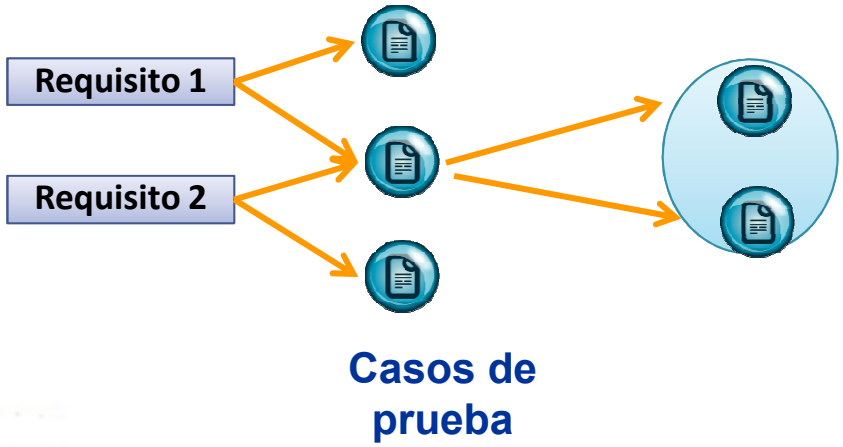
Trazabilidad

La trazabilidad o rastreabilidad es la "**aptitud para rastrear la historia, la aplicación o la localización** de una entidad mediante indicaciones registradas (norma ISO 8402).

La trazabilidad permite:

- Determinar el **origen y destino** de cada elemento.
- Relacionar a los **requisitos, diseño e implementación** y hacer un seguimiento efectivo **a las consecuencia de los cambios** que ocurran en estos de manera continua y como se ven afectados uno a otros.
- Determinar la **cobertura de los requerimientos en las prueba**, incluso cuando los proyectos son muy grandes o complejos.

Capítulo 4 - Técnicas de diseño de pruebas
Tema 4.1: El proceso de desarrollo de pruebas



Descripción de un caso de prueba según el estándar IEEE 829

- **Valores de entrada ("Input values"):** descripción de los datos de entrada de un objeto de pruebas.
- **Precondiciones ("preconditions"):** situación previa a la ejecución de pruebas o características de un objeto de pruebas antes de llevar a la práctica (ejecución) un caso de prueba.
- **Resultados esperados ("expected results"):** datos de salida que se espera que produzca un objeto de pruebas (resultado esperado - "expected result")
- **Poscondiciones ("post conditions"):** características de un objeto de pruebas tras la ejecución de pruebas, descripción de su situación tras la ejecución de las pruebas,
- **Dependencias ("dependencies"):** orden de ejecución de casos de prueba, razón de los dependencias.
- **Identificador distinguible ("distinct Identification"):** Identificador o código con el objeto de vincular, por ejemplo, un informe de errores al caso de prueba en el cual ha sido detectado.
- **Requisitos ("requirements"):** características del objeto de pruebas que el caso de prueba debe evaluar.

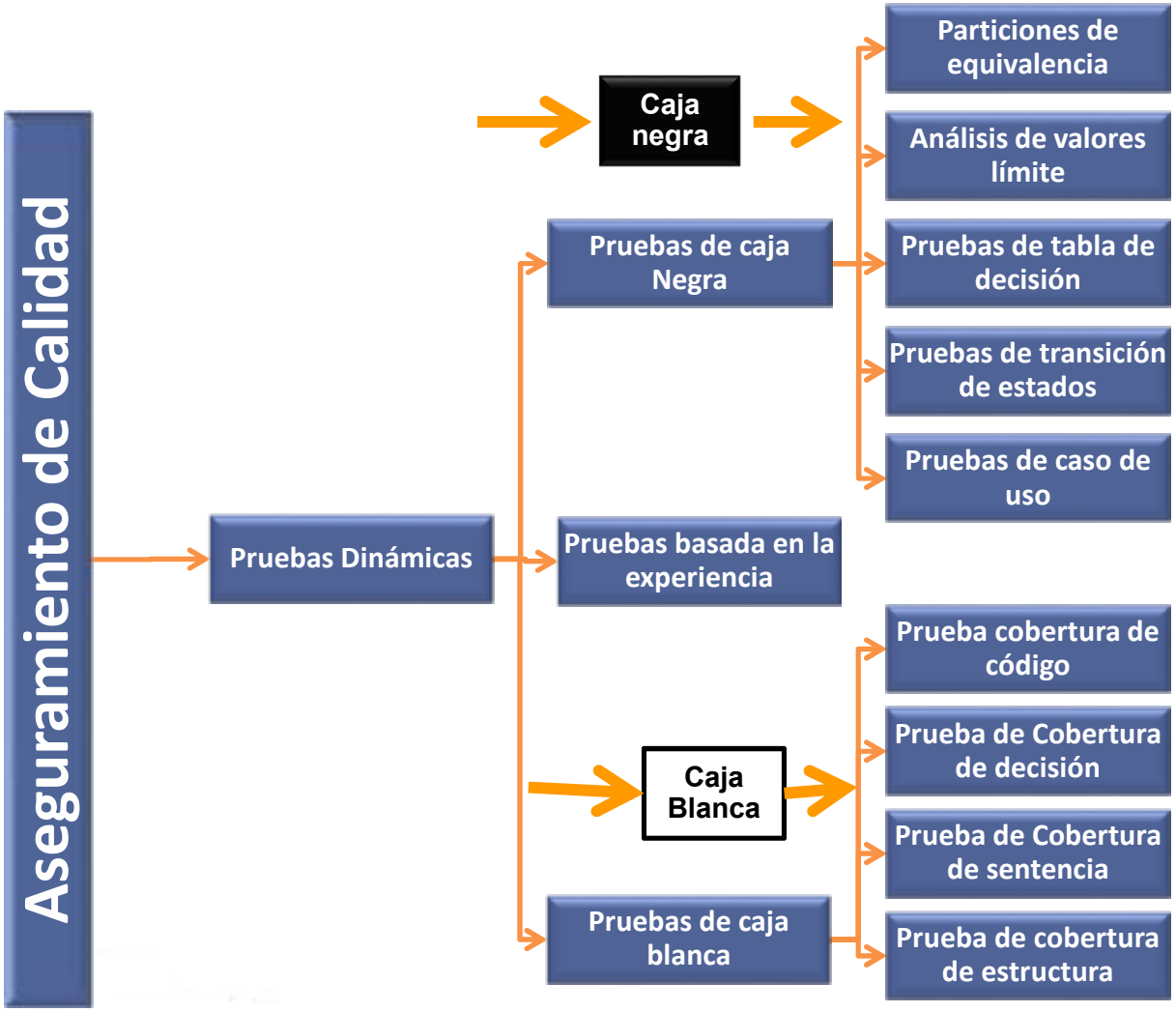
4.2: Categorías de las técnicas de diseño de pruebas

Término

- **Técnica de diseño de pruebas de caja negra:** "black-box test design technique" Procedimiento para obtener y/o **seleccionar casos de prueba basados en el análisis de la especificación**, tanto funcional como no funcional de un componente o sistema sin referencia a su estructura interna.
- **Técnica de diseño de pruebas basada en la experiencia:** "experienced-based test design technique" Procedimiento para derivar y/o seleccionar **casos de prueba basados en la experiencia**, conocimiento **e intuición** del probador.
- **Técnica de diseño de prueba de caja blanca:** "white-box test design technique" Procedimiento para obtener y/o **seleccionar los casos de prueba basados en la estructura interna** de un componente o sistema.

Capítulo 4 - Técnicas de diseño de pruebas
Tema 4.2: Categorías de las técnicas de diseño de pruebas

El propósito de una técnica de diseño de prueba es identificar las condiciones de prueba y los casos de prueba.



Técnicas Basada en la Especificación o de Caja Negra

Estas están basadas en la especificación (ya sea funcional o no funcional) y en la experiencia. Se basan en la documentación de base de pruebas, sin tener referencia de su estructura interna.

Algunas características comunes

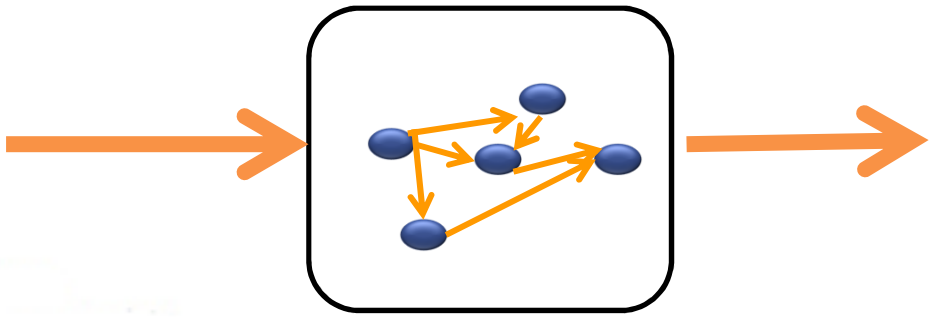
- Las funciones del software son operativas. Su objetivo es revisar la funcionalidad
- La entrada se acepta de forma adecuada y mira el comportamiento en la salida
- Se produce una salida correcta, y la integridad de la información externa se mantiene

Técnicas Basada en la Estructura o de Caja Blanca o Flujo de Control

Estas son basadas en un análisis de la estructura del componente o sistema. La información de como el software es construido es usado para generar los casos de prueba, por ejemplo código y diseño. Flujo de control, flujo de datos.

Algunas características comunes

- Su objetivo es revisar la estructura
- El porcentaje de cobertura se utiliza para la creación de casos de pruebas



Características comunes de las técnicas basadas en la experiencia

Los casos de prueba se derivan:

- La experiencia y conocimiento de los tester, desarrolladores , usuarios tanto funcional como no funcional y otros involucrados en el proyecto, su uso y su entorno.
- Conocimiento a cerca de probables defectos y su distribución
- El diseño de los casos de prueba se realiza a través de la exploración
- Se basan en la creatividad
  - Generar ideas y posibilidades
  - Se buscan los problemas que uno imagina que existen

4.3: Técnicas basadas en la especificación o de caja negra

Glosario de términos

- **Análisis de valores límite:** “boundary value analysis”Técnica de diseño de pruebas de caja negra en la cual los casos de prueba son diseñados basándose en los valores límite. Véase también valor límite.
- **Valor Límite:** “boundary value” Valor de entrada o de salida que se encuentra en la frontera de una partición de equivalencia o a la mínima distancia incremental a cualquier lado de la frontera, por ejemplo el valor mínimo o máximo de un rango.
- **Pruebas de tabla de decisión:** “decision table testing” Técnica de diseño de casos de prueba de caja negra en la que los casos de prueba se diseñan para ejecutar las combinaciones de entradas y/o estímulos (causas) representadas en una tabla de decisión.
- **Particiones de equivalencia:** “equivalence partition”Porción del dominio de una entrada o una salida para la cual se asume que el comportamiento de un componente o sistema, basado en la especificación, es el mismo.

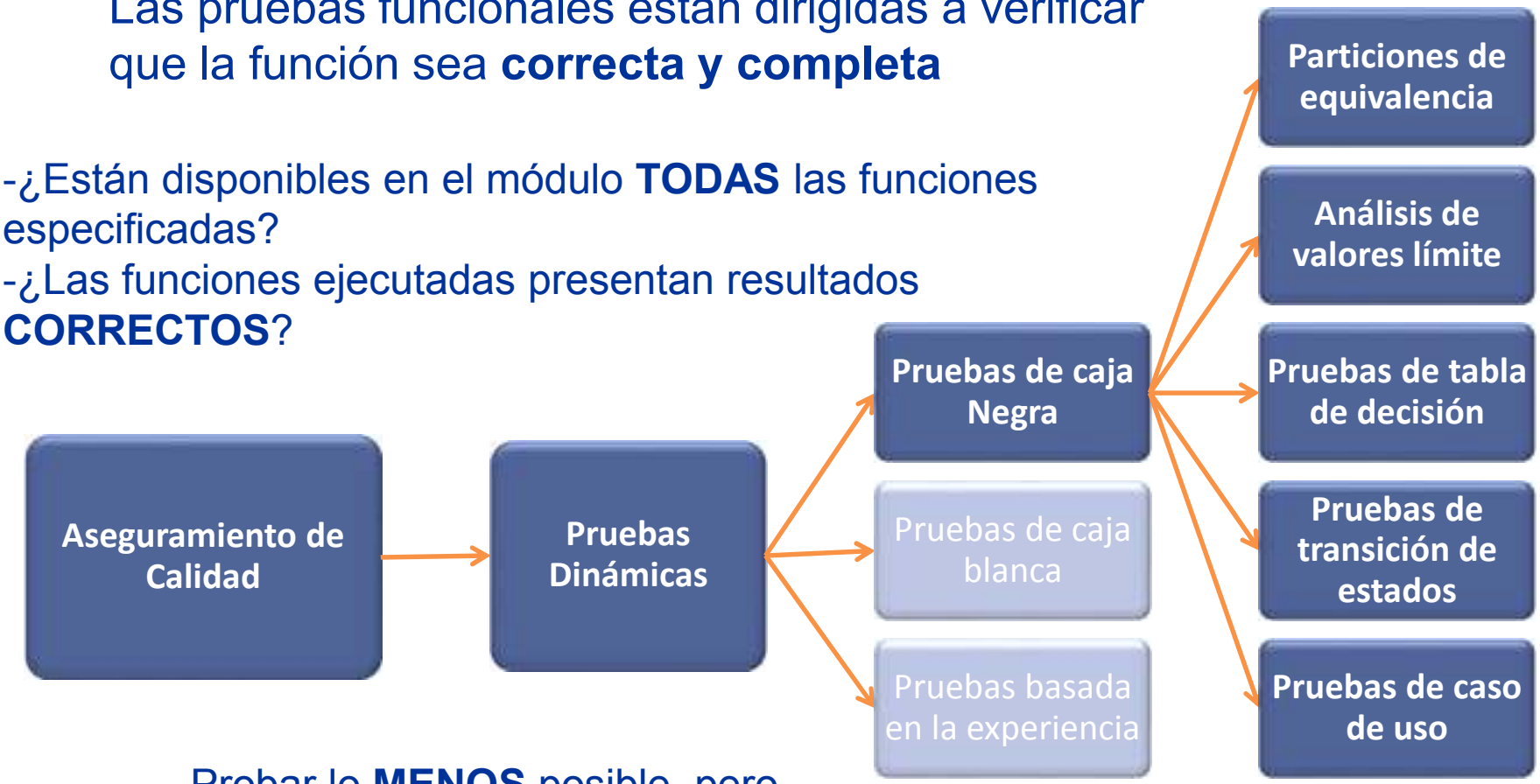
Glosario de términos

- **Pruebas de transición de estado:** "state transition testing" Técnica de diseño de pruebas de caja negra en la cual los casos de prueba son diseñados para ejecutar transiciones de estado válidas e inválidas. Véase también pruebas de conmutador de multiplicidad N.
- **Pruebas de caso de uso:** "use case testing" Técnica de diseño de prueba de caja negra en la que los casos de prueba están diseñados para ejecutar escenarios de usuario.

Capítulo 4 - Técnicas de diseño de pruebas
Tema 4.3: Técnicas basadas en la especificación o de caja negra

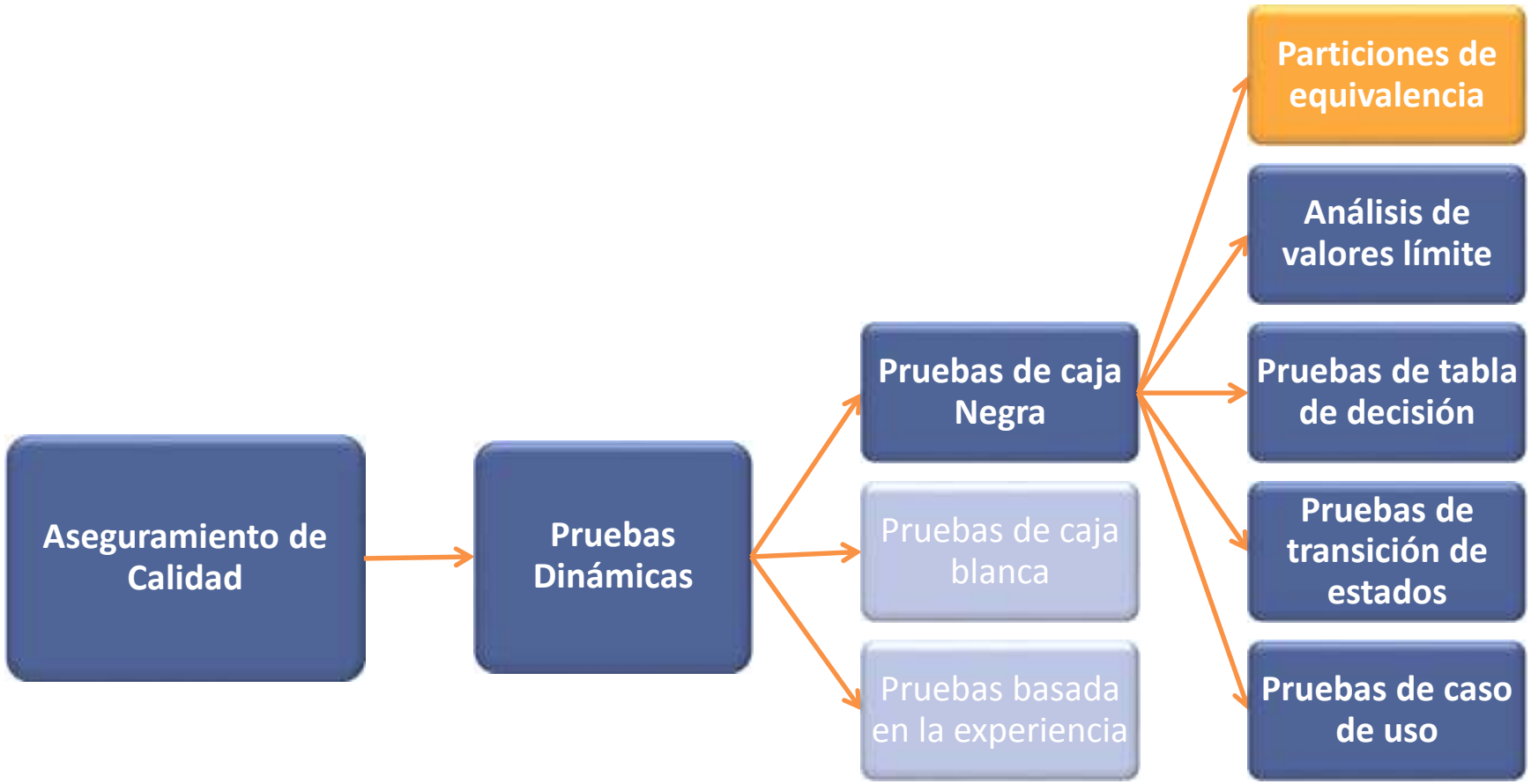
Las pruebas funcionales están dirigidas a verificar que la función sea **correcta y completa**

-¿Están disponibles en el módulo **TODAS** las funciones especificadas?
-¿Las funciones ejecutadas presentan resultados **CORRECTOS**?

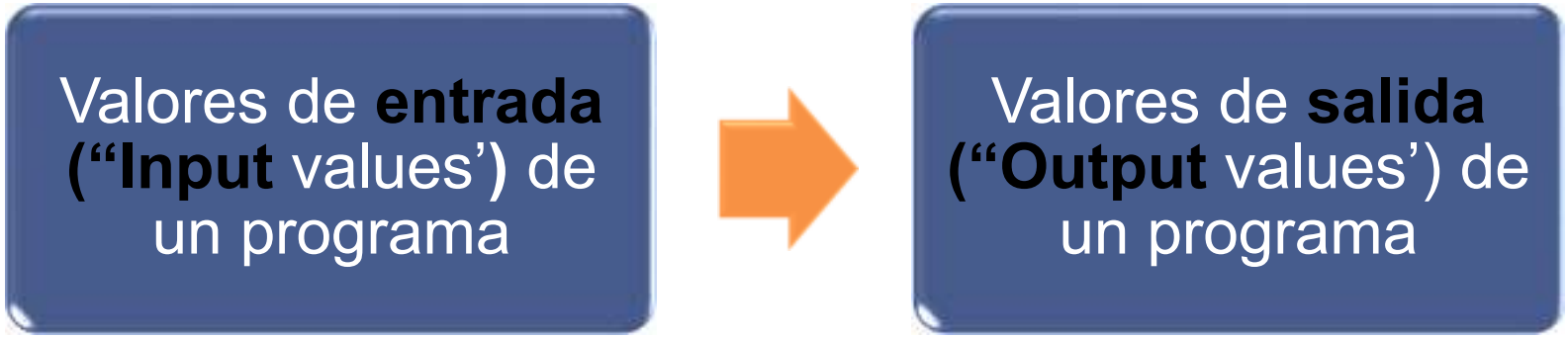


-Probar lo **MENOS** posible, pero
-Probar **TANTO** como sea posible.

Capítulo 4 - Técnicas de diseño de pruebas
Tema 4.3: Técnicas basadas en la especificación o de caja negra



Particiones de equivalencia o clases de equivalencia (CE)



- El rango de valores definido se agrupa en clases de equivalencia para las cuales se aplican las siguientes reglas
  - Todos los valores para los cuales **se espera** que el programa tenga un **comportamiento común** se agrupan en una clase de equivalencia (CE).
  - Las clases de equivalencia **no** pueden **superponerse** y **no** pueden **presentar ningún salto (discontinuidad).**
  - Las clases de equivalencia pueden consistir en un **rango** de valores (por ejemplo, 0<x< 10) o en un valor **aislado** (por ejemplo, x=Verdadero).

Particiones de equivalencia - Proceso

1. Todas las **variables de entrada** del objeto de prueba se identifican
   - Variable de entrada 1
   - Variables de entrada 2
2. Las variable de entradas identificadas, se clasifican de la de la siguiente manera:
   - Rango: Se define una clase de equivalencia válida y dos no válidas.
   - Valor específico: Se define una clase de equivalencia válida y una no válida.
   - Miembro de conjunto: Se define una clase de equivalencia válida y otra no válida.
   - Lógica: Se define una clase válida y otra no válida.

Rango de valores

- Si una condición de entrada especifica un **rango de valores** (entre 1 y 9999). Se define **una CE Valida** (1 >= valor <= 9999) y dos **CE Invalida** (valor < 1 y valor > 9999).

Valor especifico

- Si una CE requiere un **valor específico** (el primer carácter tiene que ser una letra), se define **una CE Valida** (una letra) y **una CE Invalida** (no es una letra).

Conjunto de valores de entrada

- Si una CE especifica un **conjunto de valores de entrada**, se define una CE Valido **para cada uno de los valores válidos**, y una **CE Invalido**

  EJ: (CEV para “Apartamento", “Casa" y “Finca", y CEI para “Hotel").

Número de valores

- Si una condición de entrada especifica el **número de valores** (una casa puede tener uno o dos cuartos), identificar una CE Valido (un cuarto o dos cuartos) y dos CE Invalido (0 cuartos y 3 cuartos).

3. Se clasifican los **dos tipos** de clases:

-Clases de equivalencia **válidas:** todos los valores pertenecientes al rango de definición se combinan en una única clase de equivalencia si son tratados de forma idéntica por el objeto de prueba.

Clases de equivalencia **no válidas:** *se distinguen dos casos que se encuentran fuera del rango de definición.*

- Valores con un formato correcto pero fuera del rango pueden ser combinaciones en una o más clases de equivalencia.
- Valores expresados en un formato erróneo pero, generalmente construido en una CE independiente.

Resultado

Condición de entrada o Variable	Clase de Equivalencia	Estado	Representante	Casos de prueba

Ejemplo: Entero para identificar el día del mes.

- Valores posibles: [1..31]

1. Identificar la condición de entrada

Condición de entrada especifica un **rango de valores**

2. Variable de entrada

Día del mes

3. Identificar las clases de equivalencia:

Clase de equivalencia valida	Clase de equivalencia invalida
•Números entre 1 y 31	•Números mayores que 31 •Números menores que 1

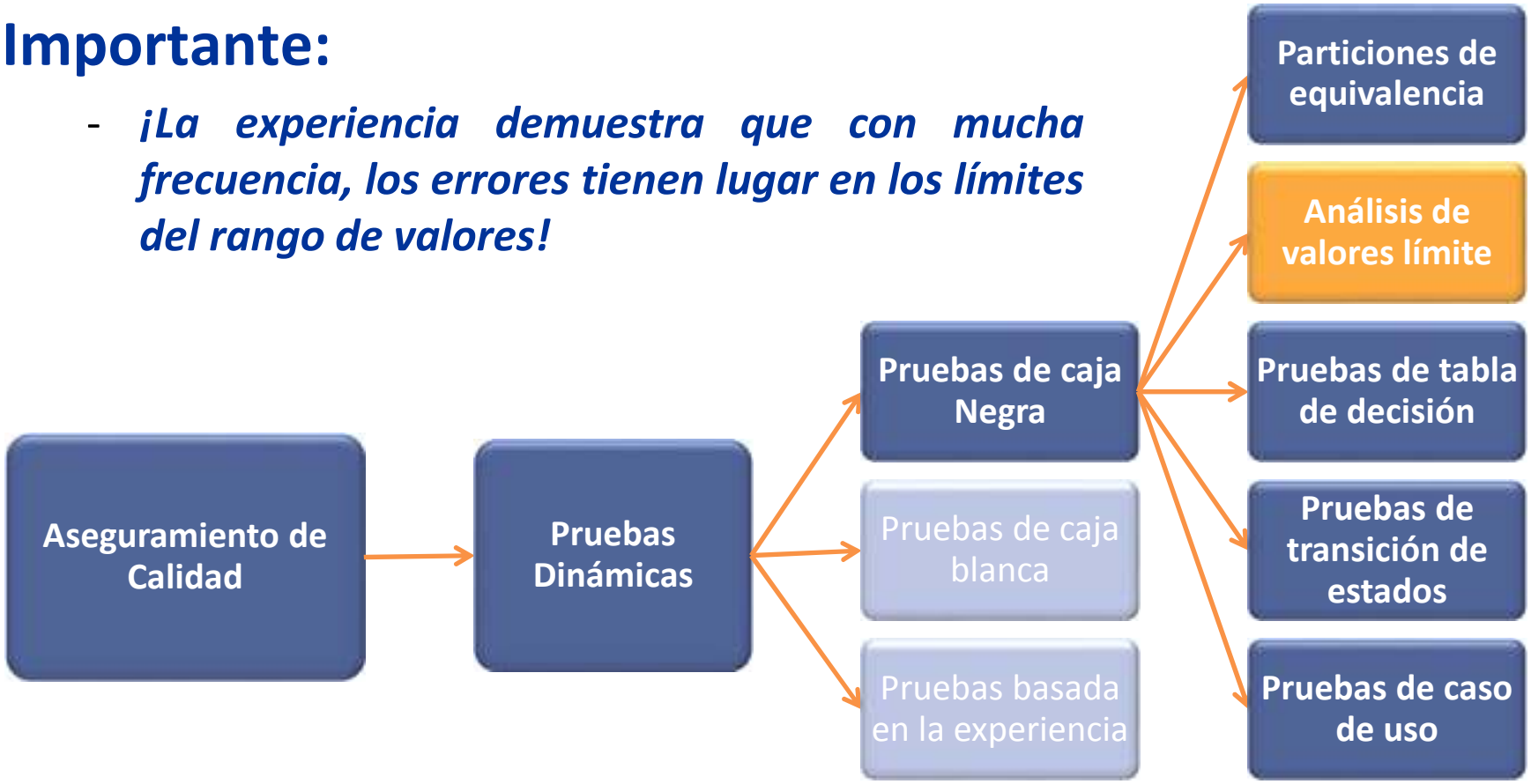
Resultado

Condición de entrada o Variable	Clase de Equivalencia	Estado	Representante	CP1	CP2	CP2
Días del mes	Números entre 1 y 31	Valido	5	X		
	Números menores que 1	No valido	0		X	
	Números mayores que 31	No valido	40			X

Análisis de valores limite

Importante:

- *¡La experiencia demuestra que con mucha frecuencia, los errores tienen lugar en los límites del rango de valores!*



Análisis de valores limite

Este análisis amplía la técnica de partición en clases de equivalencia, **introduciendo una regla para seleccionar a los representantes**.

Puede ser aplicado en todos los niveles de prueba.

Los **valores frontera** (valores límite) de la clase de equivalencia deben ser probados de **forma intensiva.**

El análisis de valores limite supone que:

- La clase de equivalencia está compuesta de un **rango continuo de valores,** no por un valor individual o un conjunto de valores discretos.
- Se pueden definir los limites para el rango de valores

Se utiliza el siguiente esquema:

Rango de valores Valor Mínimo <= X >= Valor Máximo		
Limite inferior – x	Limite inferior	Limite inferior + x
Limite superior – x	Limite superior	Limite superior + x
x es el menor incremento definido para el valor Por ejemplo 1 para valores enteros		

Ejemplo:

Software que permite realizar descuentos promocionales a los clientes en un rango del 15% al 50%.

Rango de valores permitidos para un descuento en %: 15 >= x <= 50

1. Definición de CE: 3 clases
   - CE: X<15
   - CE: 15,00 >= x <= 50,00
   - CE: x > 50
2. Análisis de valores limite:

Extiende los representantes a:

CE: 14,9%; 15%; 15,1%; 49,9%; 50%; 50.1%

En lugar de un representante para la CE valida, ahora hay seis representantes (cuatro validos y dos no validos)

Análisis de valores límite para CE no válida

- Los valores límite para clases de equivalencia no validas tienen poco sentido.
  - Los representantes de una CE no válida en la frontera de una CE válida ya se encuentran cubiertas a través del esquema básico.

1. Definición de CE: 3 clases
   - CE: X<15
   - CE: 15,00 >= x <= 50,00
   - CE: x > 50

- Para rangos de valores definidos como un conjunto de valores, en general, no *es* posible definir los valores límite.

  Por ejemplo: *Cundinamarca, Antioquia, Valle, Meta, Tolima.*

•La partición en clases de equivalencia y el análisis de valores límite tratan **entradas** en condiciones **aisladas**.

•Sin embargo, una condición de entrada puede tener efectos sólo en **combinación** con otras condiciones de entrada.

•Todos los métodos descritos previamente no tienen en cuenta el efecto de **dependencias y combinaciones.**

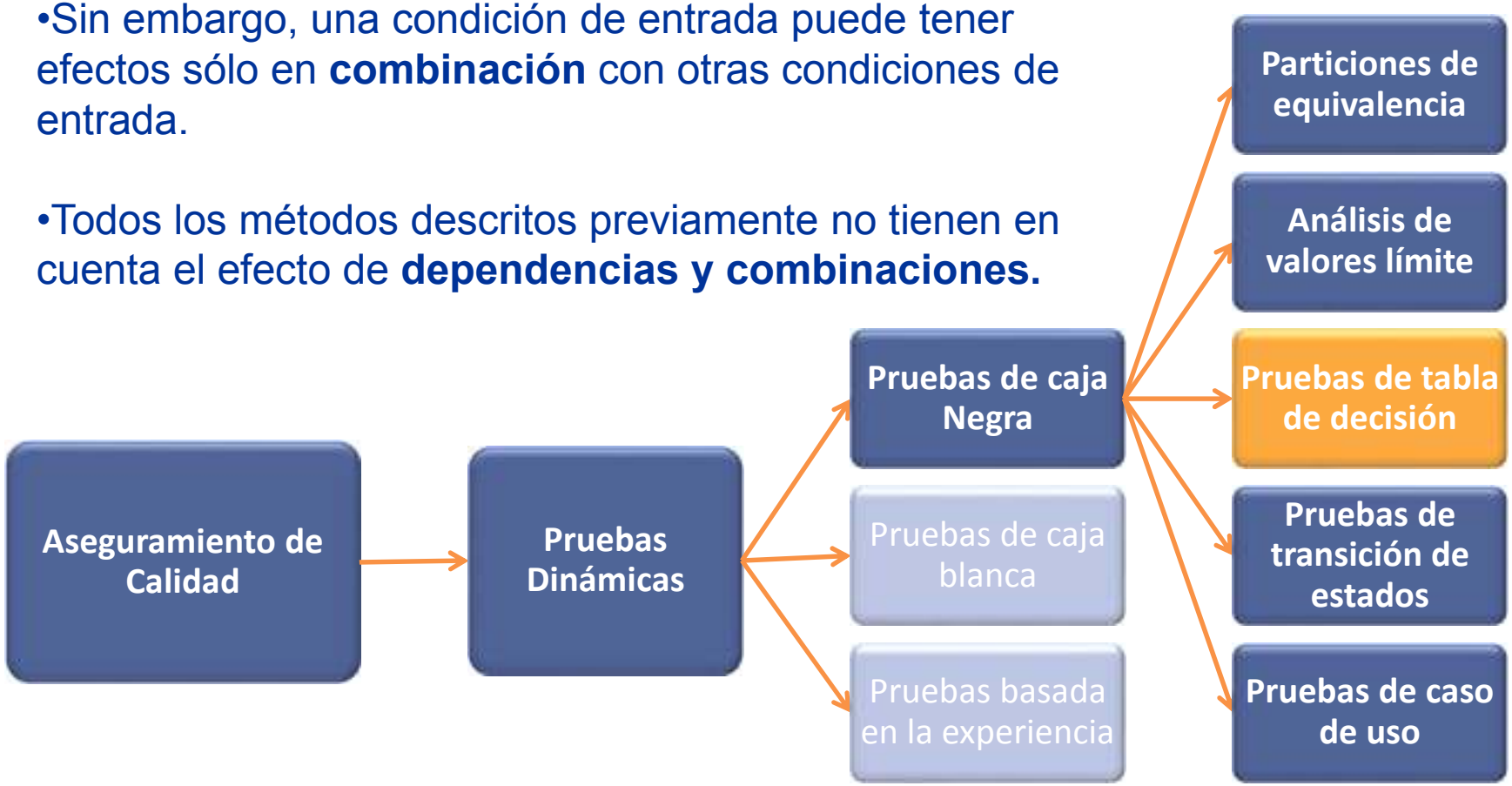
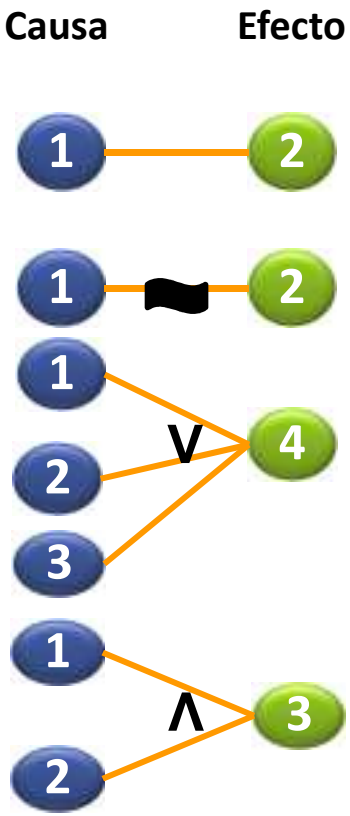


Diagrama causa y efecto

- Se genera **traduciendo la especificación** (lenguaje normalmente informal) a un **lenguaje formal**.
- El objeto de prueba esta sometido a una determinada cantidad de **efectos** que se remontan a sus respectivas **causas**.
- Se representa con los siguientes símbolos:



  - Aseveración (El efecto 2 ocurre si la causa 1 ocurre)
  - Negación (El efecto 2 ocurre si la causa 1 no ocurre)
  - O (El efecto 4 ocurre si ocurre alguna de las causas)
  - Y (El efecto 3 ocurre si ambas causas 1 y 2 ocurren)

Ejemplo: Se cuenta con la siguiente especificación de un módulo para buscar un carácter en una cadena:

- Se debe ingresar la longitud de la cadena y el carácter a buscar.
- Si la longitud de la cadena es mayor a 50 debe mostrarse un mensaje de error.
- Si el carácter se encuentra en la cadena, debe mostrarse su posición, de lo contrario debe mostrarse el mensaje "Carácter no encontrado".

1. Identificación de las condiciones de entrada (Causas)

Causa 1: Entero positivo entre 1 y 50

Causa 2: Carácter a buscar

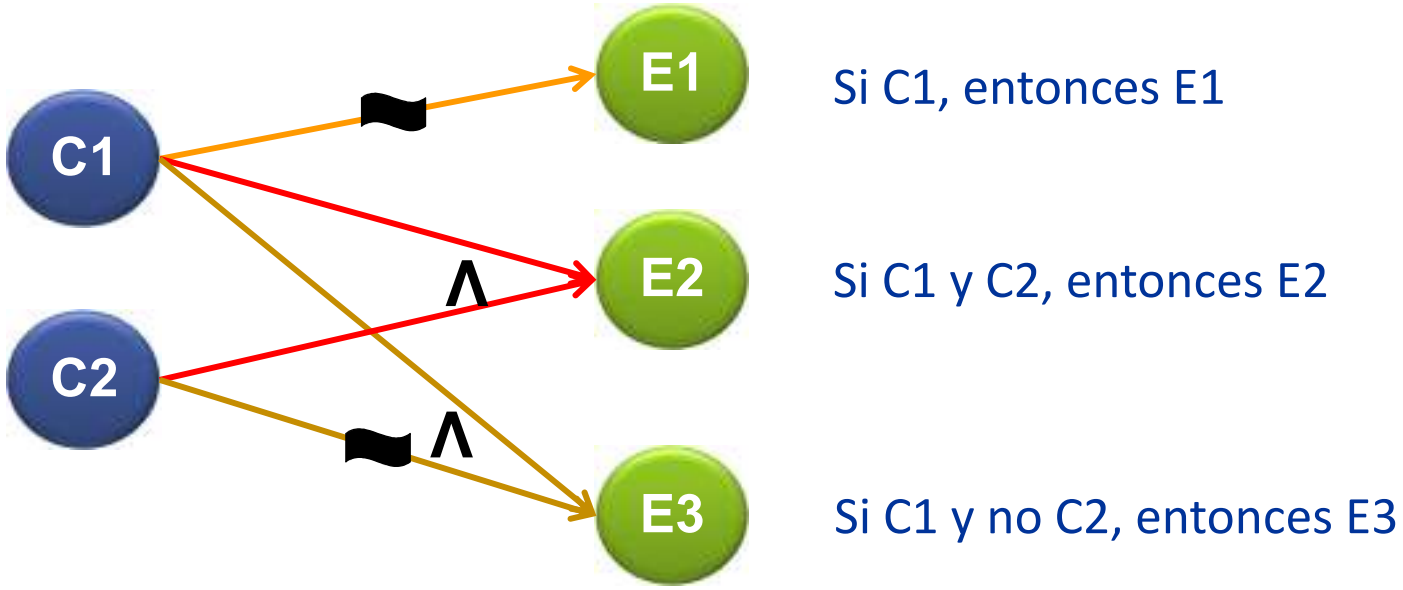
2. Identificación de los efectos o condiciones de salida:

Efecto 1: Longitud fuera de rango

Efecto 2: Posición del carácter

Efecto 3: Carácter no encontrado

3. Elaboración del gráfico de causa y efecto



4. Elaboración de tabla de decisión

Se sigue el siguiente procedimiento ("1" representa la presencia de la causa o efecto, "0" representa su ausencia, y "-" indica que no se considera).

	Prueba 1	Prueba 2	Prueba 3
Causa 1	1	1	0
Causa 2	1	0	-
Efecto 1	0	0	1
Efecto 2	1	0	0
Efecto 3	0	1	0

La fortaleza de las pruebas de tabla de decisión es que crea combinaciones de condiciones que de otra manera no pudiesen haber sido ejecutadas durante las pruebas.

- **Uso práctico.**
  - La especificación está dividida en partes fáciles de gestionar, por lo que conlleva a una tabla de decisión con un tamaño práctico.
  - Es difícil deducir valores límite a partir del diagrama causa-y-efecto o de la tabla de decisión.
  - Es recomendable combinar casos de prueba obtenidos a partir de tablas de decisión con los obtenidos a partir de un análisis de valores límite.
  - El número de causas y efectos analizados determinarán la complejidad del diagrama causa-y-efecto: para n precondiciones cuyos posibles valores puedan ser **verdadero o falso, se pueden generar 2^n casos de prueba**,
  - Para sistemas de **mayor tamaño este método sólo es controlable con el apoyo de herramientas**.

Beneficios.

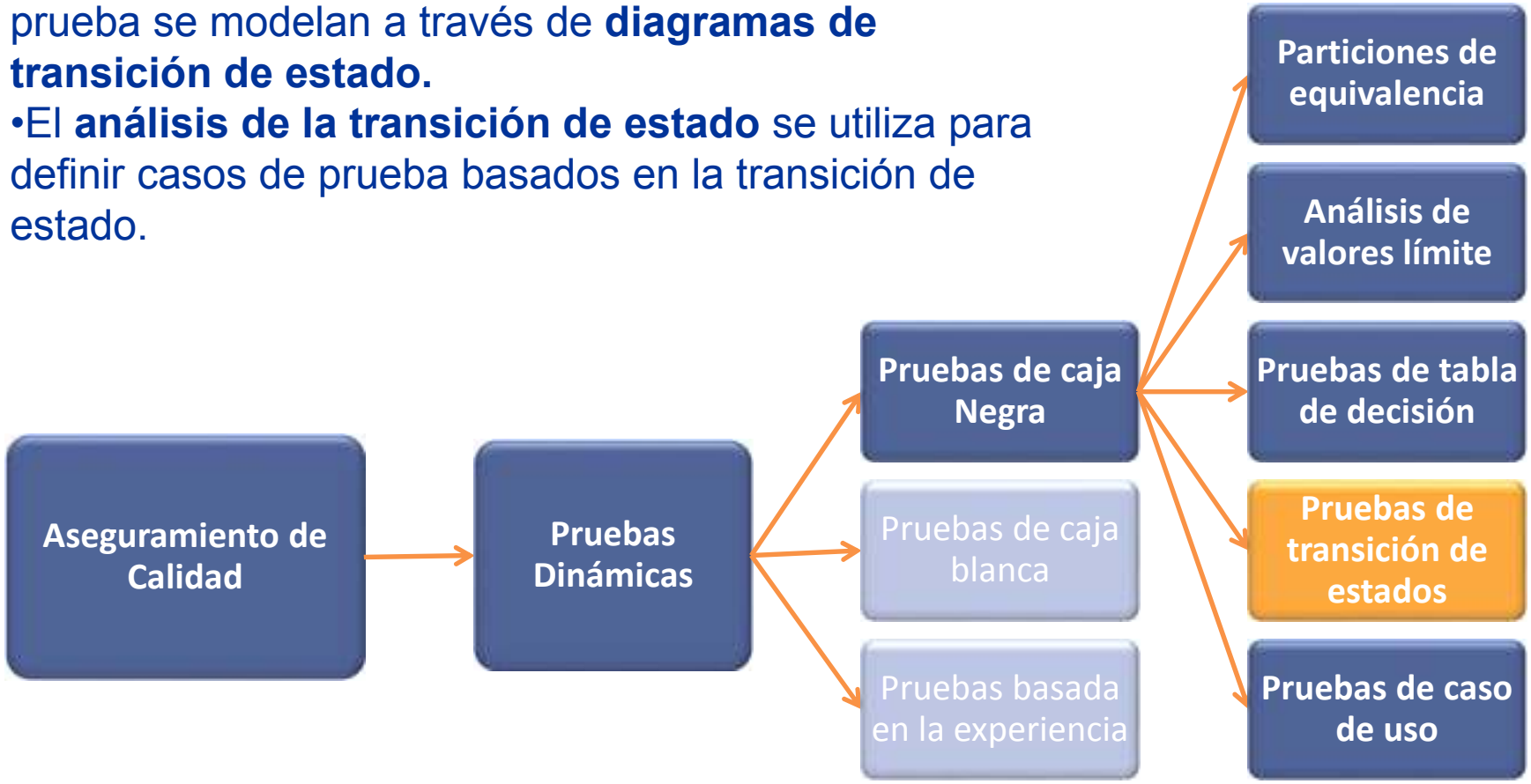
- Identificación sistemática de combinaciones de entradas (combinaciones de causas) que no podrían ser identificadas utilizando otros métodos.
- Los casos de prueba son fáciles de obtener a partir de la tabla de decisión.
- Facilidad de determinar una cobertura suficiente de casos de prueba, por ejemplo, por lo menos un caso de prueba por cada columna de la tabla de decisión.
- El número de casos de prueba se puede reducir por la fusión sistemática de columnas de la tabla de decisión.

Desventajas.

- El establecimiento de un gran número de causas conduce a resultados complejos y extensos.
- Por lo tanto, se puede incurrir en muchos errores en la aplicación de este método.
- Esto hace necesario el uso de una herramienta.

Capítulo 4 - Técnicas de diseño de pruebas
Tema 4.3: Técnicas basadas en la especificación o de caja negra

•Los distintos estados que puede tomar un objeto de prueba se modelan a través de **diagramas de transición de estado.**
•El **análisis de la transición de estado** se utiliza para definir casos de prueba basados en la transición de estado.



Pruebas de transición de estados

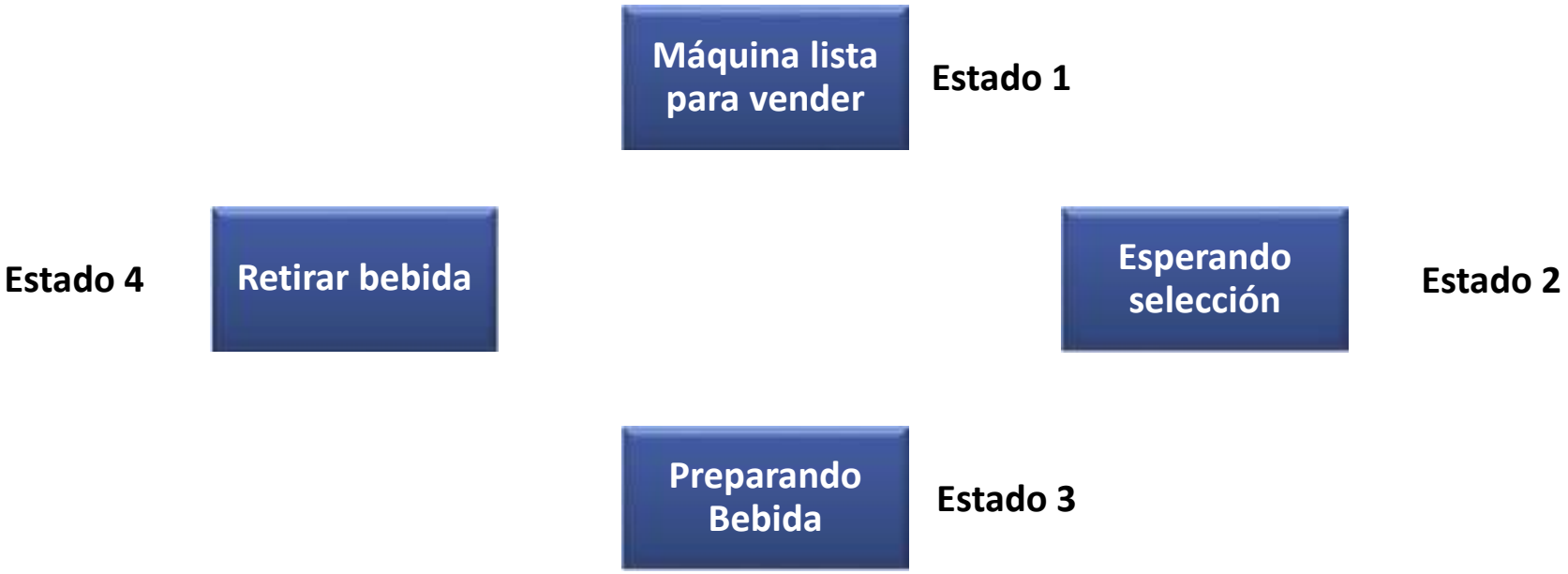
- Esta técnica se basa en el diagrama de transición de estados, que se construye para los objetos relevantes del sistema objeto de la prueba.
- El diagrama de transición de estados, presenta los diferentes estados de un objeto y los eventos que generan las transiciones.
- Se debe tabular las combinaciones posibles que deben ocurrir para que un objeto pase de un estado a otro.
- Las pruebas de transición de estados son las mas utilizadas en las pruebas de sistemas particulares con flujos de negocio basados en los estados.

Proceso

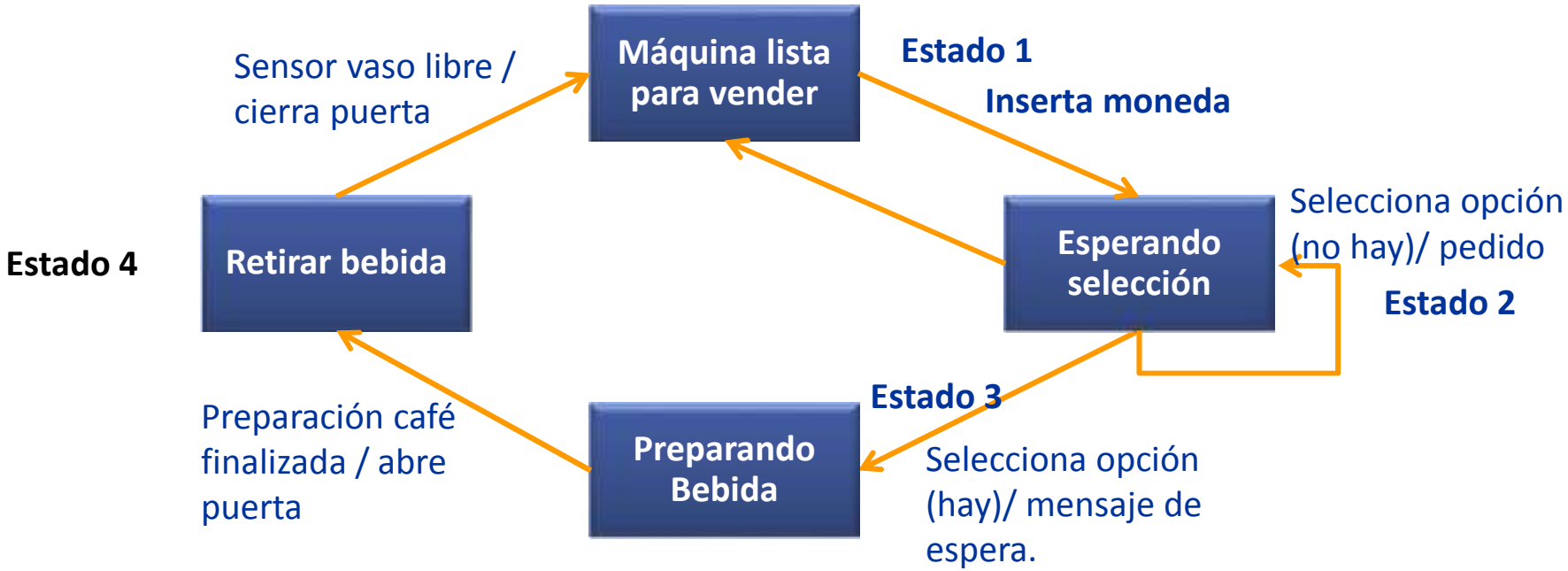
1. Elegir una perspectiva desde donde observar y modelar el sistema.
2. Identificar los estados en los que puede estar el sistema, mediante una diagrama de estados.
3. Agregar las transiciones, eventos, condiciones y acciones que pueden aplicarse a cada estado.
4. Utilizar el grafo (o una tabla equivalente) para predecir el comportamiento del sistema.
5. Validarlo con los requerimientos del sistema.
6. Para cada transición (u otro criterio más exigente) verificar que la acción y el siguiente estado ocurren

Ejemplo: Máquina dispensadora de café

1. Elegir una perspectiva desde donde observar y modelar el sistema.
2. Identificar los estados en los que puede estar el sistema.



3. Agregar las transiciones, eventos, condiciones y acciones que pueden aplicarse a cada estado.



Árbol de transición de estado

- Para determinar los casos de prueba utilizando un diagrama de transición de estado se puede construir un **árbol de transición**.
- Para todos los estados, las posibles transiciones de un estado a otro se representan como ramas.
- El estado Inicial es la raíz del árbol.
- Para cada estado que puede ser alcanzado desde el estado inicial, se crea un nodo que está conectado a la raíz a través de una rama.
- Esta operación se repite para todos los estados sucesivos,
- L a creación de ramas finaliza si:
  - El estado correspondiente al nodo es un estado final (hoja del árbol), o
  - El mismo nodo con el mismo estado ya es parte del árbol.
- **Cada camino**, desde la raíz hasta una hoja, representa un caso de prueba para las pruebas de transición de estado.

Recordar que los diagramas de estado contienen los siguientes elementos:

- **Estado**: abstracción que representa una situación del sistema que afecta su comportamiento.
- **Transición**: indica la posibilidad de cambio de un estado a otro, esta asociado a un evento. La transición puede disparar acciones.
- **Evento**: situación que provoca el cambio en un estado. Puede tener una condición asociada para diferenciar distintas transiciones.
- **Acción**: salida, comportamiento u otro resultado esperado.

Lista de estado	Lista de acciones	Lista de eventos
Lista para vender	Devuelve moneda	Inserta moneda Cancela
Esperando Selección	Pedido	Selecciona bebida
Preparando bebida	Mensaje de espera	Preparación finalizada
Retirar bebida	Abre puerta Cierra puerta	Sensor vaso libre

La tabla se construye con base en la combinación de estados y eventos. Ciertas combinaciones estado - evento producen acciones.

- Criterio de salida de la prueba.
  - -**Cada estado** debe haber sido alcanzado al menos una vez.
  - -**Cada transición** debe haber sido ejecutada al menos una vez.

- Beneficios / Desventajas de este método.

  - Buen método de pruebas para aquellos objetos de prueba que pueden ser descritos como una **máquina de estado**.

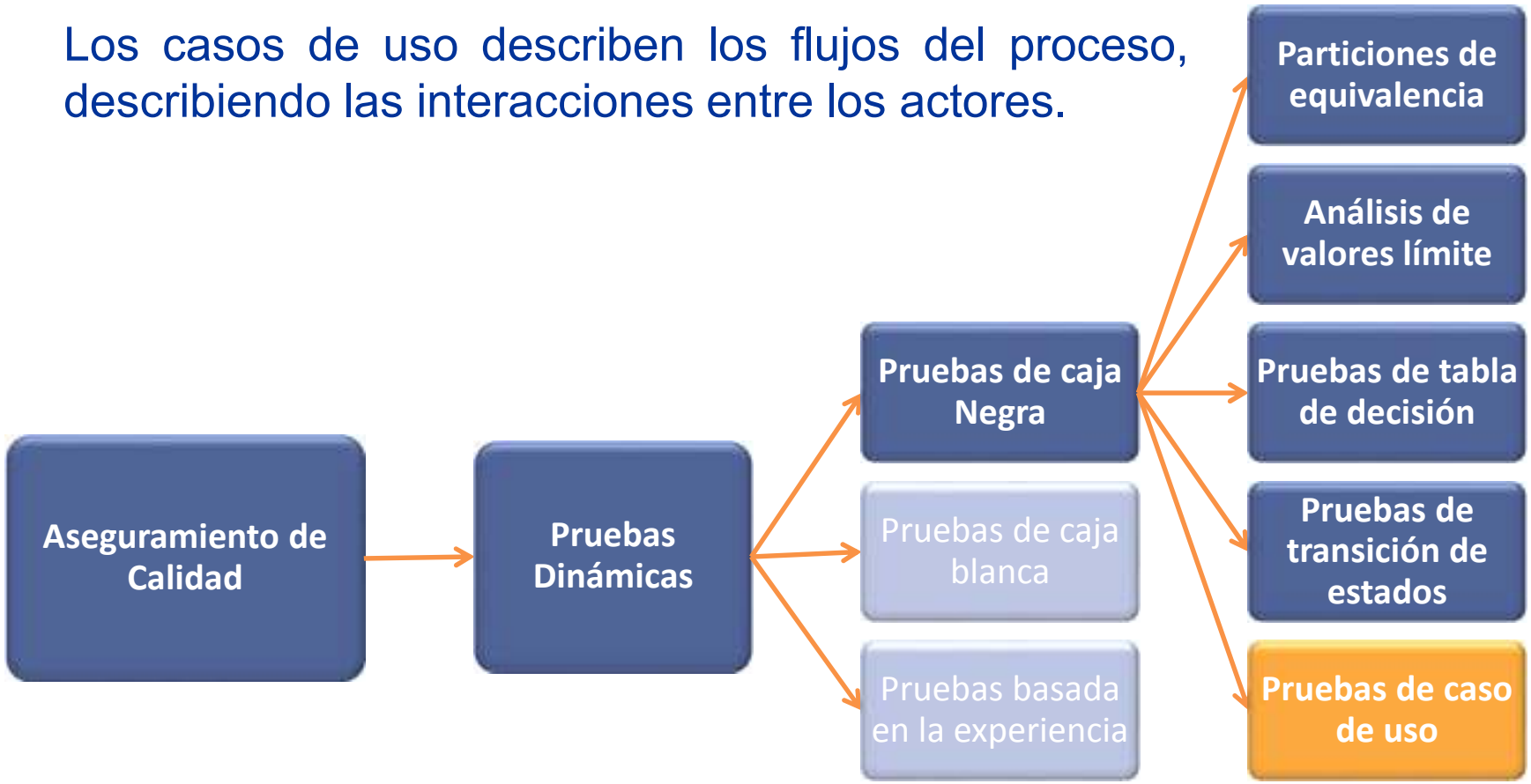
  - Buen método de pruebas para probar clases, sólo en el caso de disponer del **ciclo de vida del objeto**.

  - Con frecuencia, los estados son más bien **complejos**, es decir, es necesario un gran número de parámetros para describir el estado,

    - En estos casos, diseñar casos de prueba y analizar los resultados de las pruebas puede ser difícil y puede implicar un consumo de tiempo considerable.

  - La sola cobertura de todos los estados no garantiza una cobertura completa de las pruebas.

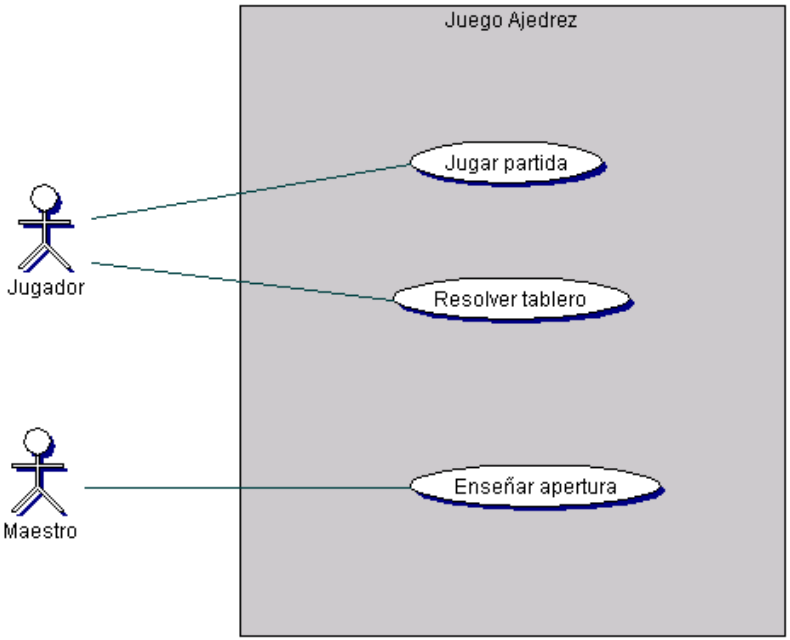
Los casos de uso describen los flujos del proceso, describiendo las interacciones entre los actores.



Pruebas basadas en casos de uso

- Los casos de prueba se obtienen directamente a partir de los casos de uso del objeto de prueba.
  - El objeto de prueba es visto como un sistema reaccionando con actores.
  - Un caso de uso describe la interacción de todos los actores involucrados conduciendo a un resultado final por parte del sistema.
  - Todo caso de uso llena precondiciones que deben ser cumplidas con el objeto de ejecutar el caso de uso (caso de prueba) de forma satisfactoria.
  - Todo caso de uso tiene post-condiciones que describen el sistema tras la ejecución del caso de uso (caso de prueba).

Ejemplo de un diagrama de casos de uso sencillo (Fuente: http://www.chuidiang.com/)



El diagrama de la izquierda descubre la funcionalidad de un Sistema " Restaurant" sencillo.

- Los casos de uso están representados por óvalos y los actores están representados por figuras de palo.
- El actor 'Jugador' puede Jugar partida y Resolver tablero.
- El actor Maestro sólo puede enseñar apertura.
- La caja define los límites del sistema "Juego de Ajedrez", es decir, los casos de uso representados son parte del sistema a modelar y no los actores

¿Qué es un escenario?

Un escenario es una instancia de un caso de uso, el cual describe **un camino a través del flujo** de eventos.

•La descripción de un **caso de uso** incluye, pero no está limitado a:
- -Precondiciones.
- -Resultados esperados / comportamiento del sistema.
- -Pos condiciones.

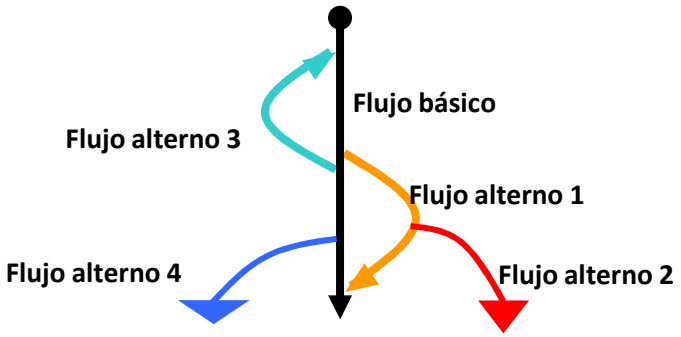
•Estos elementos descriptivos también son utilizados para definir el **caso de prueba** correspondiente.

•Cada **caso de uso** puede ser utilizado como la fuente para un caso de prueba.

•**Cada alternativa** en el diagrama corresponde a un caso de prueba separado.

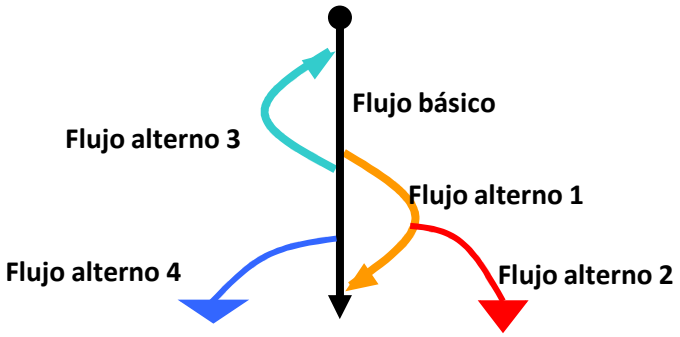
Flujos de eventos

- **Flujos normales o básicos:** Entradas validas (happy path)
- **Flujos alternos:** Otras entradas validas o invalidas



Identificando escenarios

Escenario 1	Flujo básico			
Escenario 2	Flujo básico	Flujo alterno 1		
Escenario 3	Flujo básico	Flujo alterno 1	Flujo alterno 2	
Escenario 4	Flujo básico	Flujo alterno 3		
Escenario 5	Flujo básico	Flujo alterno 3	Flujo alterno 1	
Escenario 6	Flujo básico	Flujo alterno 3	Flujo alterno 1	Flujo alterno 2
Escenario 7	Flujo básico	Flujo alterno 4		
Escenario 8	Flujo básico	Flujo alterno 3	Flujo alterno 4	



Beneficios

- Pruebas apropiadas para **pruebas de aceptación y pruebas de sistema**, dado que cada caso de uso describe un escenario a probar.
- Pruebas apropiados si las especificaciones del sistema se encuentran disponibles en UML.

Desventajas

- **Nula obtención** de casos de prueba adicionales mas allá de la información aportada por el caso de uso.
- Por lo tanto este método debería ser utilizado solo en **combinación con otros métodos** de diseño de casos de prueba.

Conclusiones generales

-Las pruebas de caja negra se centran en lo que se espera de un módulo, es decir, intentan encontrar casos en que el módulo no se atiene a su especificación. Por ello se denominan pruebas funcionales, y el probador se limita a suministrarle datos como entrada y estudiar la salida, sin preocuparse de lo que pueda estar haciendo el módulo por dentro.

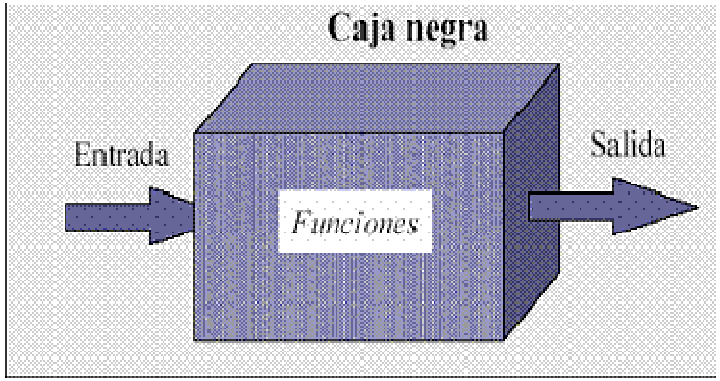
-Las pruebas de caja negra están especialmente indicadas en aquellos módulos que van a ser interfaz con el usuario (en sentido general: teclado, pantalla, ficheros, canales de comunicaciones, etc) Este comentario no obsta para que sean útiles en cualquier módulo del sistema.

-Las pruebas de caja negra se apoyan en la especificación de requisitos del módulo. De hecho, se habla de "cobertura de especificación" para dar una medida del número de requisitos que se han probado. Es fácil obtener coberturas del 100% en módulos internos, aunque puede ser más laborioso en módulos con interfaz al exterior. En cualquier caso, es muy recomendable conseguir una alta cobertura en esta línea.

Conclusiones generales

-Los métodos de caja negra (black box) siempre son utilizados en el proceso de pruebas.

-Las desventajas pueden ser compensadas con el uso de métodos adicionales de diseño de casos de prueba, por ejemplo, pruebas de caja blanca o pruebas basadas en la experiencia.



4.4 Técnicas basadas en la estructura o de caja blanca

Glosario de términos

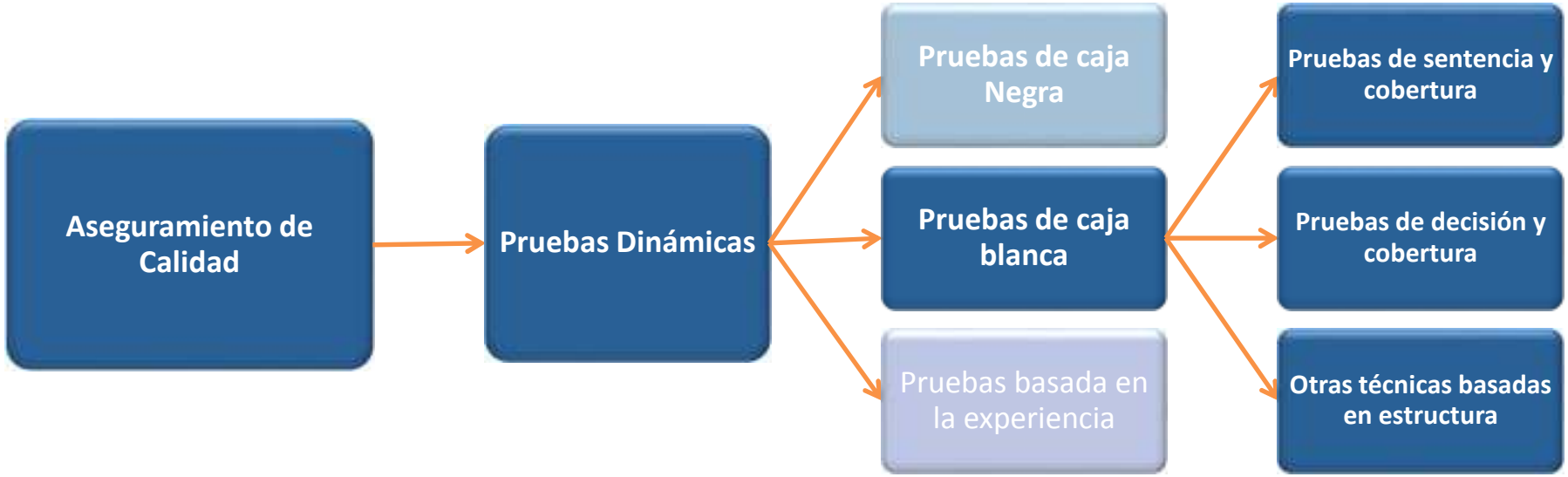
Cobertura de código: "Code coverage" Método de análisis que determina qué partes del software han sido ejecutadas (cubiertas) por el juego de pruebas y qué partes no han sido ejecutadas, ejemplo cobertura de sentencia, cobertura de decisión o cobertura de condición.

Cobertura de decisión: "decision coverage" Porcentaje de resultados de decisión que han sido practicados por un juego de pruebas. El 100% de cobertura de decisión implica tanto un 100% de cobertura de rama como un 100% de cobertura de sentencia.

Cobertura de sentencia: "statement coverage" Porcentaje de sentencias ejecutables que han sido practicadas por un juego de pruebas.

Pruebas basadas en la estructura: "structure-based testing" Pruebas basada en el análisis de la estructura interna de un componente o sistema.

Capítulo 4 - Técnicas de diseño de pruebas
Tema 4.4 Técnicas basadas en la estructura o de caja blanca



Técnicas basadas en la estructura o de caja blanca

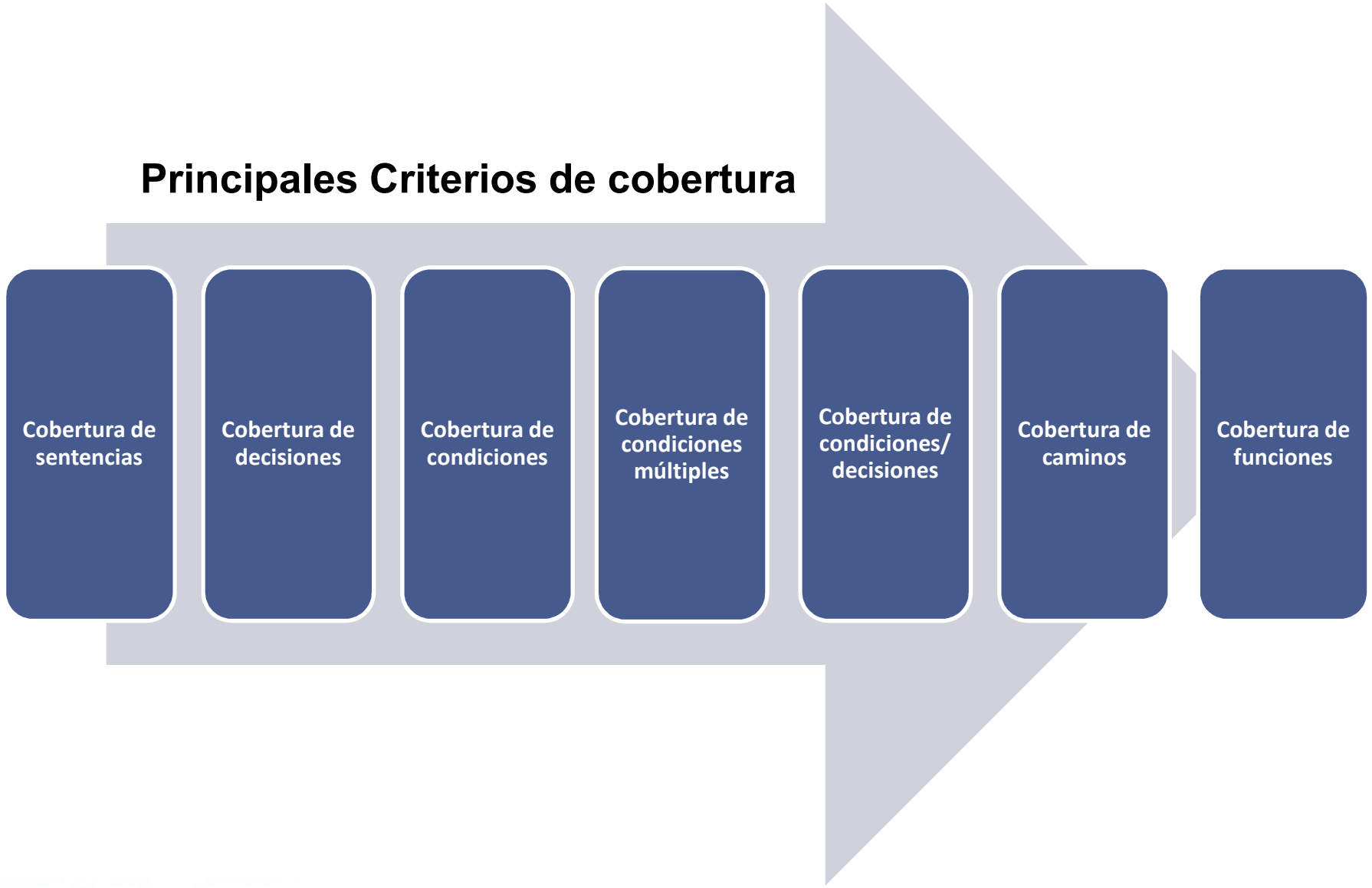
- Conocidas también como caja de cristal
- El diseño de los casos de prueba parten del comportamiento interno y la estructura del programa
- No tiene en cuenta el rendimiento
- Su objetivo es diseñar los casos de prueba, con el fin de que se ejecute al menos una vez las sentencias del programa y todas las condiciones tanto verdaderas como falsas
- Examina la lógica interna

Criterios de cobertura

De acuerdo con (Cornett 2002), el análisis de cobertura del código es el proceso de:

- Encontrar fragmentos del programa que no son ejecutados por los casos de prueba.
- Crear casos de prueba adicionales que incrementen la cobertura.
- Determinar un valor cuantitativo de la cobertura (que es, de manera indirecta, una medida de la calidad del programa).

Adicionalmente, el análisis de cobertura también permite la identificación de casos de prueba redundantes, que no incrementan la cobertura.


Principales Criterios de cobertura
Cobertura de sentencias
Cobertura de decisiones
Cobertura de condiciones
Cobertura de condiciones múltiples
Cobertura de condiciones/ decisiones
Cobertura de caminos
Cobertura de funciones

Cobertura de sentencia

- Se alcanza cuando se ejecuta el 100% de las sentencias del programa objeto de prueba
- Puede entenderse que una sentencia es aquello que termina en un punto y coma.
- Permite la identificación de los casos de prueba, teniendo en cuenta que toda sentencia ejecutable en un programa sea invocada al menos una vez.

Prueba de cobertura de decisión o de rama

Permite la identificación de casos de prueba, de tal forma que se ejecutan las vertientes verdaderas y falsas de cada decisión en un programa. Asegurando la prueba completa de las estructuras de control del programa, teniendo en cuenta que las decisiones tienen que ser simples (No formada por combinación lógica de decisiones

Cobertura de condiciones

Se trata de diseñar tantos casos de prueba como sea necesario, para que cada condición de cada decisión adopte el valor verdadero al menos una vez y el falso al menos una vez.

Cobertura de condiciones múltiples

En el caso de que se considere que la evaluación de las condiciones de cada decisión no se realiza de forma simultánea, se puede considerar que cada decisión multicondicional se descompone en varias condiciones unicondicionales.

Cobertura de condiciones/decisiones

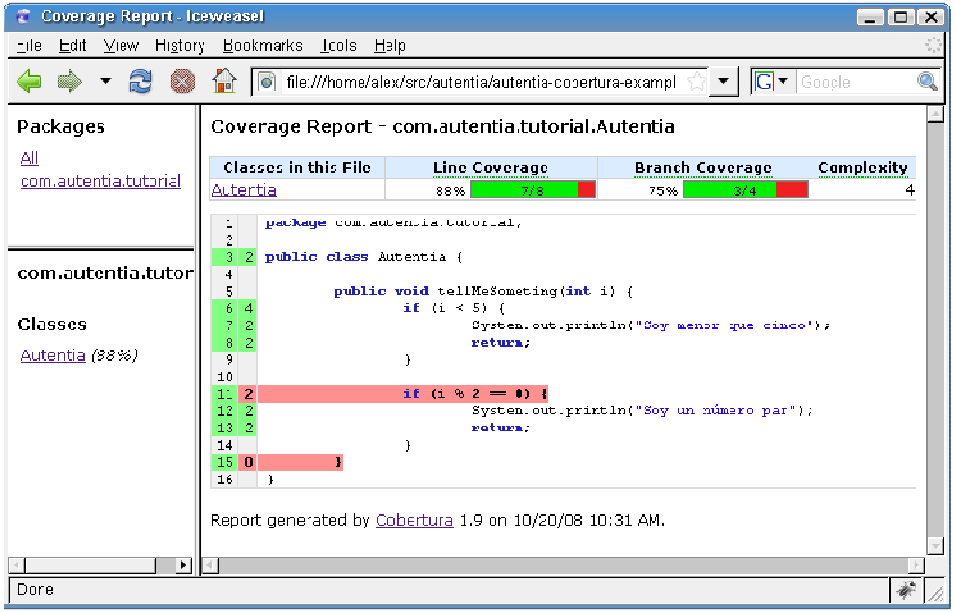
Consiste en exigir el criterio de cobertura de condiciones, obligando a que se cumpla también el criterio de decisiones.

Cobertura de caminos

Se recorren todos los caminos

Cobertura de funciones

Comprueba el número de funciones y procedimientos que han sido llamados.

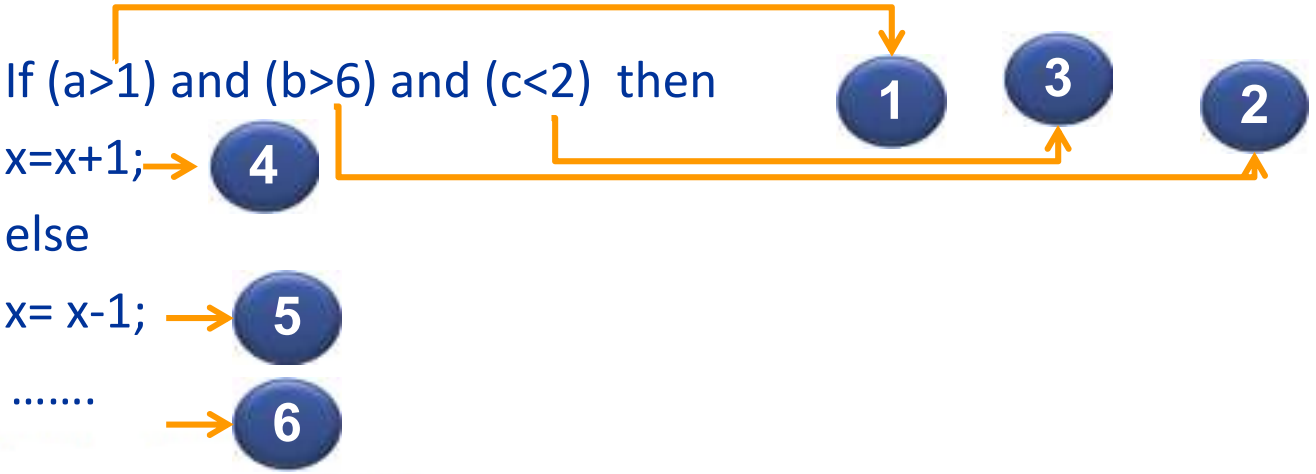


http://cobertura.sourceforge.net/mavenCobertura.html

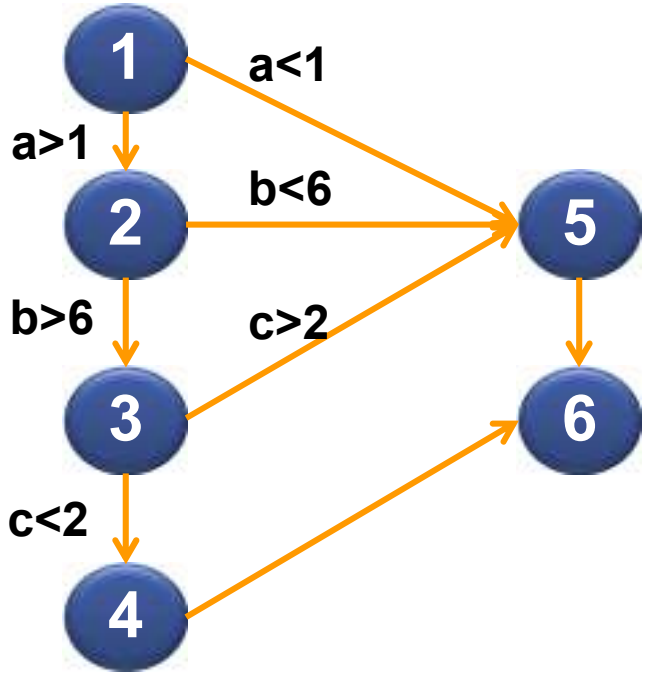
Ejemplo 1: Considerar el siguiente fragmento de programa

```
If (a>1) and (b>6) and (c<2)  then

x=x+1;

else

x= x-1;
```

1. Identificamos el grafo de flujo y la complejidad ciclomática (Capítulo 3 pruebas estáticas)
   - En el fragmento de código, se señalan los nodos



2. Dibujar el grafo de flujo



3. Calcular la complejidad ciclomática

V(G)= 8 – 6 + 2 = 4 caminos independientes

- Camino 1 ➔ 1 – 5 – 6
- Camino 2 ➔ 1- 2 – 5 – 6
- Camino 3 ➔ 1 – 2 – 3 – 5 –6
- Camino 4 ➔ 1 – 2 –3 – 4 – 6

4. Cobertura de sentencias:

Se trata de ejecutar con los casos de prueba cada sentencia e instrucción al menos una vez.

Ejecutando los casos de prueba que da en el enunciado para ejecutar cada instrucción al menos una vez: si se fijan en los caminos independientes se dan cuenta de que el flujo de ejecución pasa por todos los nodos del código.

5. Cobertura de decisiones

- Se escriben los casos suficientes para que cada condición tenga al menos una resultado verdadero y otro falso. En este caso bastaría con ejecutar el interior del if y el interior del else, sin tener en cuenta las condiciones. Cuando se tienen decisiones multicondicionales puede ejecutarse el bloque else con diferentes combinaciones.
- En este caso se cumple la cobertura de condiciones ya que se ejecutan el nodo 4 y 5, que son las sentencias del interior del if y del else respectivamente.

Ejemplo 2: Considerar el siguiente fragmento de programa

```
Procedure responsabilidad (edad, sexo, casado, premium);

Begin
   Premium = 500;
   if ((edad < 25) and (sexo = masculino) and (not casado)) then
      premium = premium + 1500;
      else if ((casado) or (sexo = femenino)) then
               Premium = premium - 200;
               else if ((edad > 45) and (edad < 65)) then
               Premium = premium – 100;

End;
```

Cobertura de sentencia

Solo hay dos sentencias en el programa y cualquier combinación de entrada cubre las sentencias.

GENERALMENTE ES UNA CONDICIÓN DEMASIADO DEBIL

Cobertura de decisión

Cobertura de decisiones	edad	sexo	casado	Caso de prueba
If – 1	<25	Masculino	FALSE	(1) 23 M F
If – 1	<25	Femenino	FALSE	(2) 23 F F
If – 2	*	Femenino	TRUE	(2)F T
If – 2	>=25	Masculino	FALSE	(3) 50 M F
If – 3	<=45	Femenino	*	(2)
If - 3	>45, <65	*	*	(3)

Cobertura de decisión ("decision coverage")

•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, cobertura de código "code coverage"),

Suficiente en la gran mayoria de los casos
Exigida al 100% en algunas normas de productos
Hay herramientas automáticas de apoyo

Cobertura de condiciones

Cobertura de condiciones	Edad	Sexo	Casado	Caso de prueba
If – 1	<25	FEMENINO	FALSE	(1) 23 F F
If – 1	>=25	MASCULINO	TRUE	(2) 30 M T
If – 2	*	MASCULINO	TRUE	(2)
If – 2	*	FEMENINO	FALSE	(1)
If – 3	<=45	*	*	(1)
If – 3	>45	*	*	(3) 70 F F
If – 3	<65	*	*	(2)
If – 3	>=65	*	*	(3)

Cobertura de Condiciones

•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 con condicionales más pequeñas.

Interesante si existen condiciones complejas con riesgo de defectos
Superconjunto de "decisiones": se puede partir de los caminos
Independientes y expandirlos a las condiciones

•Hay tres tipos de cobertura de condición.

-Cobertura de condición simple ("simple condition coverage").
-Cobertura de condición múltiple ("múltiple condition coverage").
-Mínima cobertura de condición múltiple ("mínimum múltiple condition coverage").

Cobertura de decisión / condición

Cobertura de decisiones / condiciones	Edad	Sexo	Casado	Casos de prueba
If – 1	<25	Masculino	FALSE	(1) 23 M F
If – 1	<25	Femenino	FALSE	(2) 23 F F
If – 1	<25	Femenino	FALSE	(2)
If – 1	>=25	Masculino	TRUE	(3) 70 M T
If – 2	*	Femenino	*	(2)
If – 2	>=25	Masculino	FALSE	(4) 50 M F
If – 2	*	Masculino	TRUE	(3)
If – 2	*	Femenino	FALSE	(2)
If – 3	<=45	*	*	(2)
If – 3	>45, <65	*	*	(4)
If – 3	<=45	*	*	(2)
If – 3	>45	*	*	(4)
If – 3	<65	*	*	(4)
If – 3	>=65	*	*	(3)

Cobertura de condiciones múltiples

Cobertura de condiciones múltiples	edad	sexo	casado	Caso de prueba
If – 1	<25	Masculino	TRUE	(1) 23 M T
If – 1	<25	Masculino	FALSE	(2) M F
If – 1	<25	Femenino	TRUE	(3) 23 F T
If – 1	<25	Femenino	FALSE	(4) 23 F F
If – 1	>=25	Masculino	TRUE	(5) 30 M T
If – 1	>=25	Masculino	FALSE	(6) 7 M F
If – 1	>=25	Femenino	TRUE	(7) 50 F T
If – 1	>=25	Femenino	FALSE	(8) 30 F F
If – 2	*	Masculino	TRUE	(5)
If – 2	*	Masculino	FALSE	(6)
If – 2	*	Femenino	TRUE	(7)
If – 2	*	Femenino	FALSE	(8)
If – 3	<=45 >=65	*	*	IMPOSIBLE
If – 3	<=45, <65	*	*	(8)
If – 3	>45, >=65	*	*	(7)
If – 3	>45, <65	*	*	(6)

Cobertura de condición ("condition coverage") conclusiones generales

•La **cobertura de condición simple** es un instrumento débil para probar condiciones múltiples.

•La **cobertura de condición múltiple** es un método mucho mejor.
- -Asegura cobertura de sentencia y decisión.
- -Sin embargo, tiene como resultado un alto número de casos de prueba: 2n.
- -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.

•La **mínima cobertura de condición múltiple** es incluso mejor, debido a:
- -Reduce el número de casos de prueba (de (n+1) a (2n)).
- -Las coberturas de sentencia y decisión también son cubiertas.
- -Tiene en cuenta la complejidad de las sentencias de decisión.

Todas las decisiones complejas deben ser probadas la mínima cobertura de condición múltiple es adecuada para lograr este objetivo.

Cobertura de camino ("path coverage ")

Se escriben casos de prueba suficientes para que se ejecuten todos los posibles caminos de un programa. Entendiendo camino como una secuencia de sentencias encadenadas desde la entrada del programa hasta su salida.

Los pasos a realizar para aplicar esta técnica son:

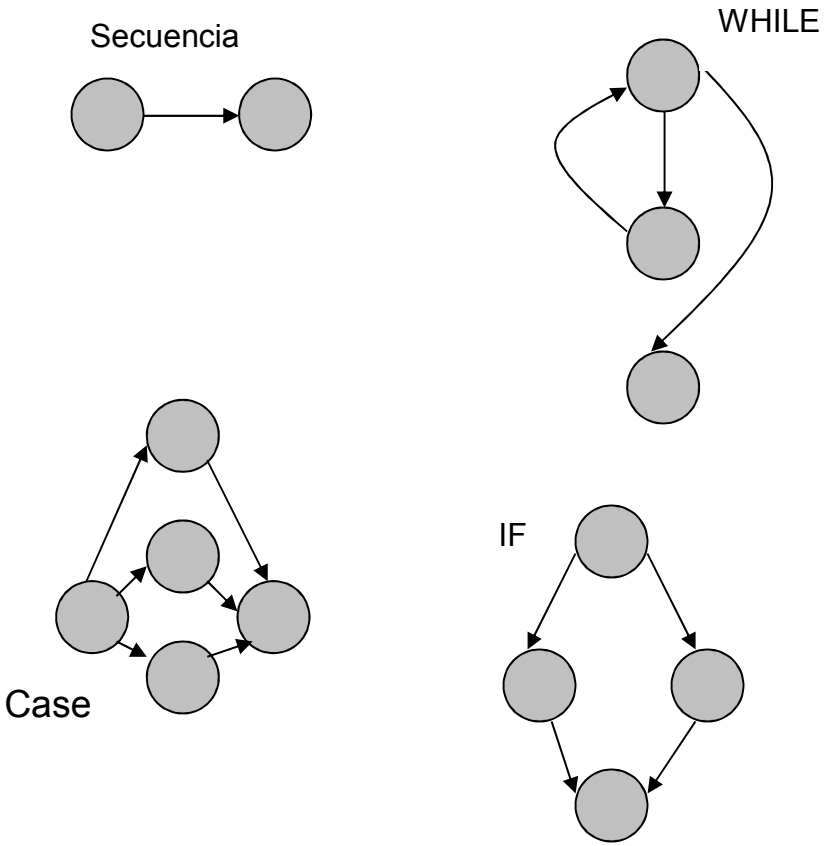
- Representar el programa en un grafo de flujo
- Calcular la complejidad ciclomática
- Determinar el conjunto básico de caminos independientes
- Derivar los casos de prueba.

$$\text{Cobertura de camino} = \frac{\text{Número de caminos cubiertos}}{\text{Numero Total de caminos}} * 100\%$$

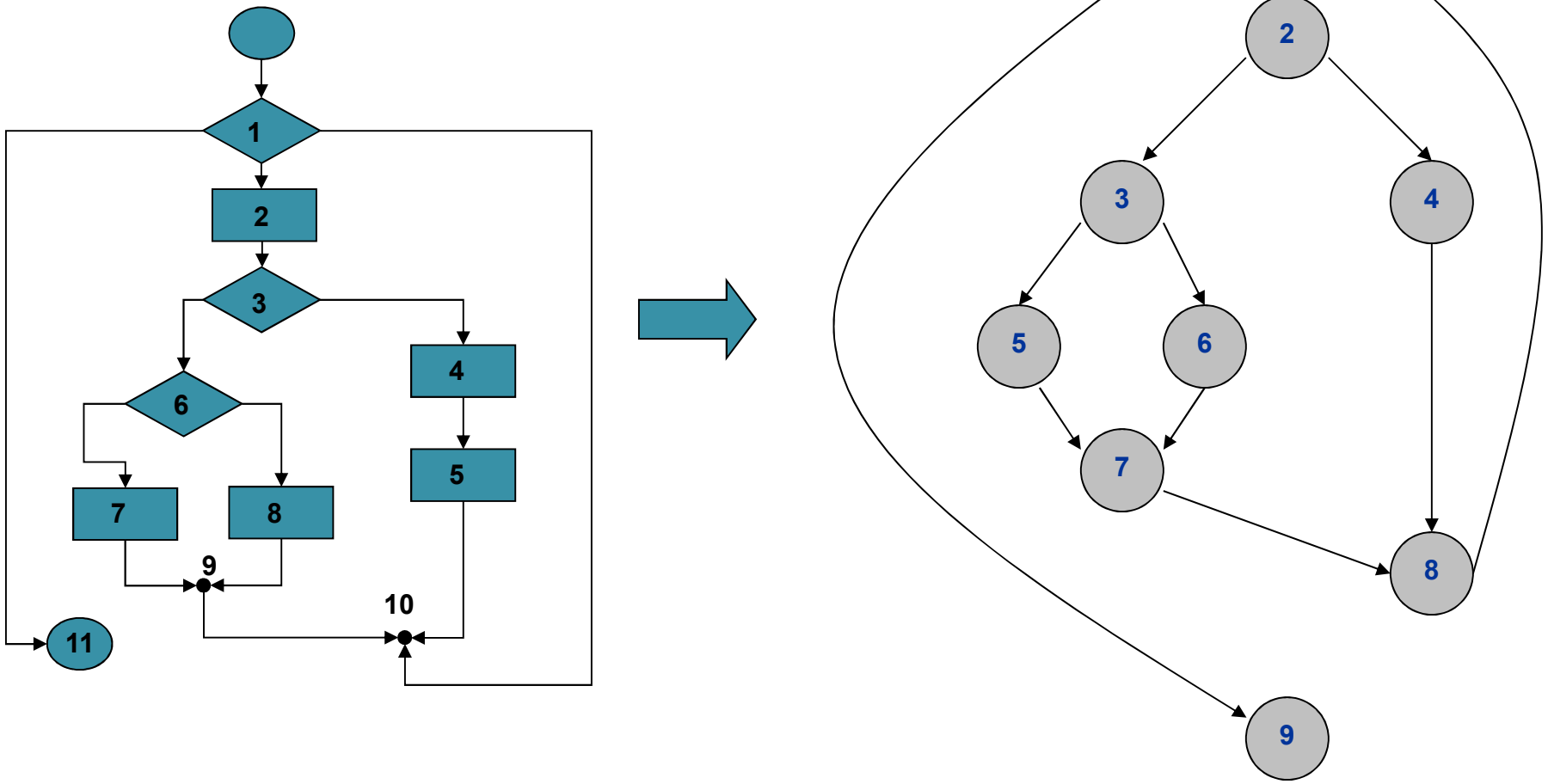
Notación

- **Nodos**: Representan cero, una o varias sentencias.
- **Aristas**: líneas que unen dos nodos.
- **Regiones**: áreas delimitadas por aristas y nodos. Cuando se contabilizan las regiones de un programa debe incluirse el área externa como una región más

Nodos Predicado: Cuando en una condición aparecen uno o más operadores lógicos (AND, OR, XOR, ...) se crea un nodo distinto por cada una de las condiciones simples. Cada nodo generado de esta forma se denomina nodo predicado.



Pasar un diagrama de flujo a un grafo

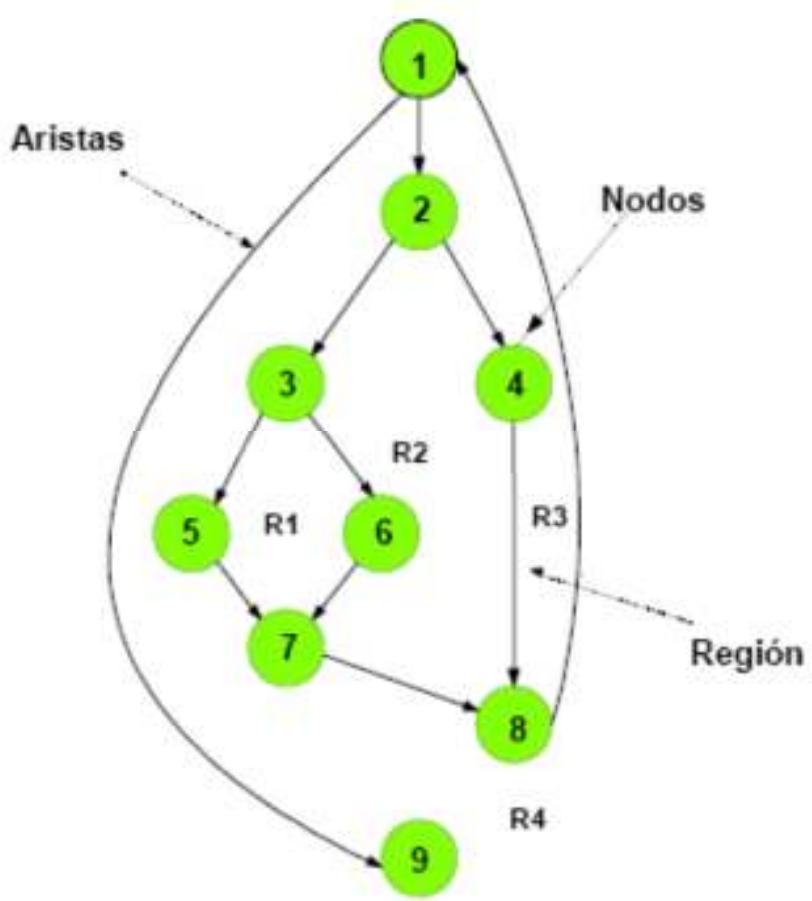


Complejidad Ciclomática

La técnica de camino básico se basa en la medida de complejidad ciclomática que es una métrica de software que provee una medición cuantitativa de la complejidad lógica de un programa.

Usada en el contexto testing, define el número de caminos independientes en el conjunto básico y entrega un limite superior para el número de casos necesarios para ejecutar todas las instrucciones al menos una vez.

El inconveniente que presenta es que no da idea de la complejidad de los datos.



- Considerando el grafo, encontramos el siguiente conjunto de caminos independientes:
  - Camino 1: 1-9
  - Camino 2: 1-2-4-8-1-9
  - Camino 3: 1-2-3-6-7-8-1-9
  - Camino 4: 1-2-3-5-7-8-1-9
- Cada nuevo camino introduce un arco nuevo.
- No se consideran caminos independientes aquellos que resulten de la combinación de otros caminos.
- El conjunto básico no es único.
- Se debe elegir como primer camino aquel que atraviese el mayor número de decisiones en el grafo.

Calculo de la complejidad ciclomática

- **V(G) = Número de regiones**
- **V(G) = Aristas - Nodos + 2p**
- **V(G) = Número de nodos predicado + 1**

Para el caso del grafo anterior, el conjunto básico calculado en todos los casos da 4.

- V(G) = Número de regiones = 4
- V(G) = Aristas – Nodos + 2 =11-9 + 2 = 4
- V(G) = Nodos Predicado + 1 = 3 +1 = 4

Resumen

•Los métodos de caja blanca y caja negra son métodos dinámicos, el objeto de prueba es ejecutado durante las pruebas.

•**El método de caja blanca (white-box) comprende:**
- **-Cobertura de sentencia ("statement coverage").**
- **-Cobertura de decisión ("decision coverage").**
- **-Cobertura de camino ("path coverage").**
- **-Cobertura de condición ("condition coverage")** (simple ('single'), múltiple-("multiple"), mínimo múltiple (" minimum multiple")).

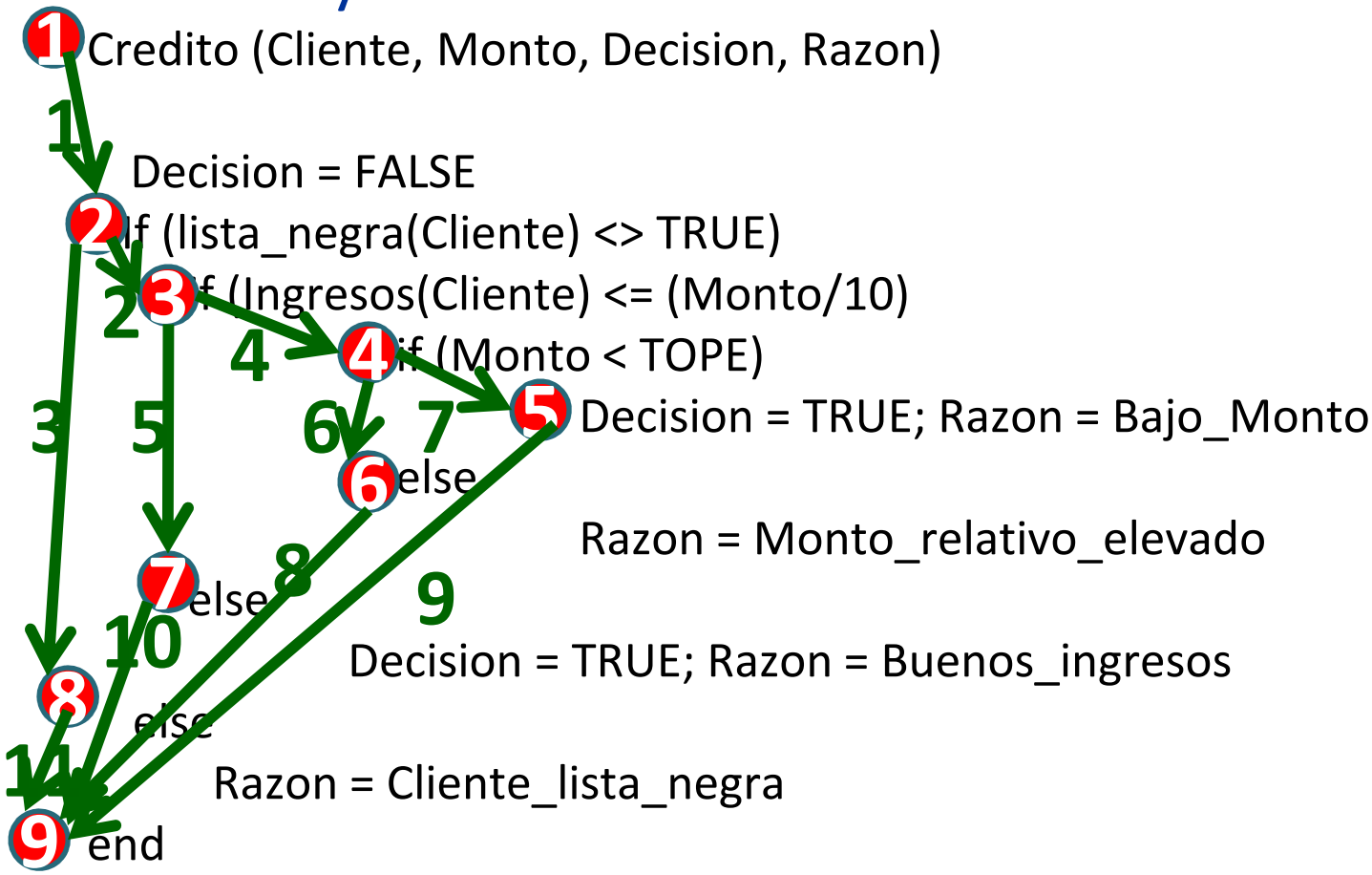
•Sólo se puede probar código existente. Habiendo funciones faltantes, este hecho no puede ser detectado. Sin embargo el **código muerto o superfluo** puede ser detectado con las pruebas de caja blanca.

•Principalmente, los métodos de caja blanca son utilizados en pruebas de bajo nivel como **pruebas de componente** o pruebas de integración.

•Los métodos difieren en la intensidad de las pruebas (profundidad de la prueba).
- -Dependiendo del método, el número de casos de prueba es distinto.

Ejercicio:

Nodos y Arcos orientados



Tema 4.5 Técnicas basadas en la experiencia

Glosario de términos

Pruebas exploratorias: Técnica informal de diseño de pruebas donde quien prueba controla activamente el diseño de las pruebas a medida que las pruebas son realizadas y utiliza la información obtenida durante las pruebas para diseñar unas nuevas y mejores. [Según Bach]

- No tienen claro ningún enfoque metodológico, se basa en la intuición y experiencia del probador con programas similares.
- Por lo general siempre para la pruebas se tienen en cuenta lo siguiente:
  - ¿Dónde se han acumulados los defectos en el pasado?
  - ¿Dónde falla el software con frecuencia?
- El aprendizaje de la aplicación se obtiene en el diseño y ejecución de las pruebas
- Las pruebas se diseñan, ejecutan y modifican de forma dinámica y sobre la marcha
- Se diseñan casos de prueba que no pueden ser creados con otras técnicas, por ejemplo:
  - Campos vacíos
  - Caracteres inválidos

Capítulo 4 - Técnicas de diseño de pruebas
Tema 4.5 Técnicas basadas en la experiencia

Diseño intuitivo de casos de prueba

- Resultados de pruebas y experiencia práctica con sistemas similares
- Experiencia de usuario en el sistema
- Que parte del sistema serán utilizados con mayor frecuencia

Predicción de errores "Error Guessing"

- En una lista se enumeran posibles errores y su probabilidad de ocurrencia
- Se crean casos de prueba dirigidos a producir los errores de la lista
- Actualizar lista de errores durante las pruebas

Pruebas exploratorias

- Se utiliza cuando la información base se encuentra poco estructurada o el tiempo para pruebas es escaso
- Revisar las partes que constituyen el objeto de prueba
- Aplicar predicción de errores únicamente sobre las partes que deben ser probadas
- Analizar resultados y desarrollar un modelo preliminar del funcionamiento del objetos de prueba
- Utilizar herramientas de captura para registrar actividades de prueba

Resumen

•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.

4.6 Escoger las técnicas de prueba

Criterios para seleccionar las técnicas de pruebas a utilizar:

- **Estado de la información** respecto al objeto de pruebas ¿hay suficiente material de especificación o es necesario realizar una prueba exploratoria?
- **Objetivo de las pruebas** ¿Son pruebas funcionales o no funcionales? ¿Hay alguna estructura para lograr los objetivos del proceso de pruebas?
- **Riesgos** ¿Es muy alta la frecuencia de uso del objeto de prueba?¿se espera un daño o perjuicio serio proveniente de defectos ocultos?
- **Precondiciones del proyecto** ¿Cuánto **tiempo** esta planificado para las pruebas y cual es el **riesgo** de no finalizar según lo estimado?¿que **método de desarrollo** es utilizado?
- **Características del objeto de prueba** ¿Cuál es la **disponibilidad** del objeto de prueba? ¿Qué **posibilidades** para ser probado ofrece el objeto de pruebas?

- **Requisitos contractuales y del cliente** ¿Hay algún **acuerdo especifico** entre el cliente y los analistas respecto de los procedimientos de prueba?¿**Qué documentos** deben ser entregados en el momento del despliegue del sistema?
- Intereses distintos causan **enfoques diferentes** en el diseño de los casos de prueba.

- 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!

Intereses distintos causan enfoques diferentes en el diseño de pruebas

•Interés del jefe de proyecto:
-Desarrollar 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 pruebas:
-Prueba suficientes e intensivas / despliegue adecuado de las técnicas necesarias desde el punto de vista del proceso de pruebas.
-Evaluar/valorar el nivel de calidad alcanzado por el proyecto.
-Asignar y utilizar los recursos planificados para las pruebas de forma óptima.

Resumen

La selección de cuales técnicas de prueba usar, depende de varios factores, incluyendo el tipo de sistema, estándares normativos, requerimientos contractuales o del cliente, nivel de riesgo, tipo de riesgo, objetivo de prueba, documentación disponible, conocimiento de los testers, tiempo y presupuesto.

Referencias bibliográficas

- Norma IEEE 1028
- Norma ISO 14598
- Norma IEEE 610
- Programa de estudios básicos de ISQTB
- Standard glossary of terms used in Software Testing, Version 2.1 (dd. April 1st, 2010)
- Libro “Análisis y diseño de aplicaciones informáticas de gestión – Una perspectiva de ingeniería del software” Mario Piattini, José Calvo, Joaquín Cervera y Luis Fernández editorial Ra-MA.