Pruebas de software

Ingeniería de Software
Dpto. Ingeniería en Sistemas de Información

Universidad Tecnológica Nacional
Facultad Regional La Plata

Conceptos Generales

- Introducción
  - Conceptos Generales
- Pruebas Estáticas
  - Revisiones Gerenciales
  - Revisiones Técnicas
  - Inspecciones
  - Walkthorughs
- Pruebas dinámicas
  - Pruebas de Unidad
  - Pruebas de Integración
  - Pruebas de Validación
  - Pruebas de Sistema
- Enfoques de Pruebas
  - Pruebas de Caja Blanca
  - Pruebas de Caja Negra

¿Qué son las Pruebas de Software?

Son un conjunto de actividades planeadas previamente y llevadas adelante de manera sistemática, con el objetivo de demostrar que un programa hace lo que se intenta que haga, así como descubrir defectos en el programa antes de usarlo.

Motivación ¿Porqué realizar las pruebas de software?

?

Conceptos Generales
Clasificación de las pruebas de software

- **PRUEBAS ESTÁTICAS (REVISIONES)**
  - Revisiones Gerenciales
  - Revisiones Técnicas
  - Inspecciones
  - Walkthorughs
- **PRUEBAS DINÁMICAS (TESTEO)**
  - Pruebas de Unidad
  - Pruebas de Integración
  - Pruebas de Sistema
  - Pruebas de Aceptación

Pruebas Estáticas
Introducción y objetivos

- Este tipo de pruebas corresponden a las inspecciones y revisiones del software desde la mejora de la documentación. (Sin ejecución del software)
- La revisión es un proceso formal y estructurado.

OBJETIVOS

- Encontrar defectos del producto, en el punto más temprano posible del ciclo de desarrollo.
- Asegurar que las partes apropiadas llegan a un acuerdo técnico, sobre el producto.
- Verificar que el producto cumple con los criterios predefinidos.
- Proveer datos del producto y del proceso de revisión.

Unas pocas horas bien aprovechadas de pruebas estáticas sobre la documentación de un proyecto de desarrollo pueden ahorrar muchas horas de corrección y rediseño de software.

Pruebas Estáticas
Revisiones

GERENCIALES

- Tienen por objetivo asegurar el progreso del proyecto, uso de los recursos y recomendar acciones correctivas.
- Participan el responsable del proyecto, líderes técnicos e ingenieros. Generalmente el responsable del proyecto tiene el liderazgo de la revisión.
- Las decisiones se toman en la reunión y/o por las recomendaciones.

TÉCNICAS

- Tienen por objetivo evaluar la conformidad de un producto con respecto a especificaciones, planes y asegurar la integridad técnica y conceptual de los cambios.
- Participan líderes técnicos e ingenieros. Generalmente el líder técnico tiene el liderazgo de la revisión.
- El líder técnico debe verificar los cambios que resulten necesarios.

Pruebas Estáticas
Revisiones

INSPECCIONES

- Tienen por objetivo detectar e identificar defectos de un producto
- Participan ingenieros pares del responsable del producto.
- Los defectos deben ser removidos.
- La verificación de cambios y correcciones es obligatoria en el proceso.

WALKTHOUGHS

- Tienen por objetivo detectar defectos de un producto, examinar alternativas y generar un foro de aprendizaje.
- Participan colegas del responsable del producto.
- Generalmente el mismo productor tiene el liderazgo de la revisión.
- El productor decide si realizar los cambios y correcciones.
- La verificación de cambios y correcciones se deja para otros puntos de control del proyecto.

Pruebas Dinámicas
Introducción

- Todas aquellas pruebas que para su ejecución requieren la ejecución de la aplicación.
- El objetivo del testeo es descubrir bugs.
- Un buen caso de prueba es el que tiene altas probabilidades de detectar un bug.
- Es necesario describir el resultado esperado para los casos de pruebas.
- Los casos de prueba son para condiciones/entradas válidas e inválidas.
- Las pruebas dinámicas se basan en un marco de testeo llamado "***Modelo V***"
- Se caracterizan por tener distintos niveles
    - Pruebas de Unidad
    - Pruebas de Integración
    - Pruebas del Sistema
    - Pruebas de Validación

Pruebas Dinámicas
Verificación & Validación

Verificación y Validación

La V&V incluyen amplio conjuntos de actividades: revisiones técnicas, auditorías de calidad y configuración, monitoreo de rendimiento, simulación, estudio de factibilidad, análisis de algoritmos, pruebas de desarrollo, pruebas de usabilidad, pruebas de calificación, pruebas de aceptación entre otras.

Verificación

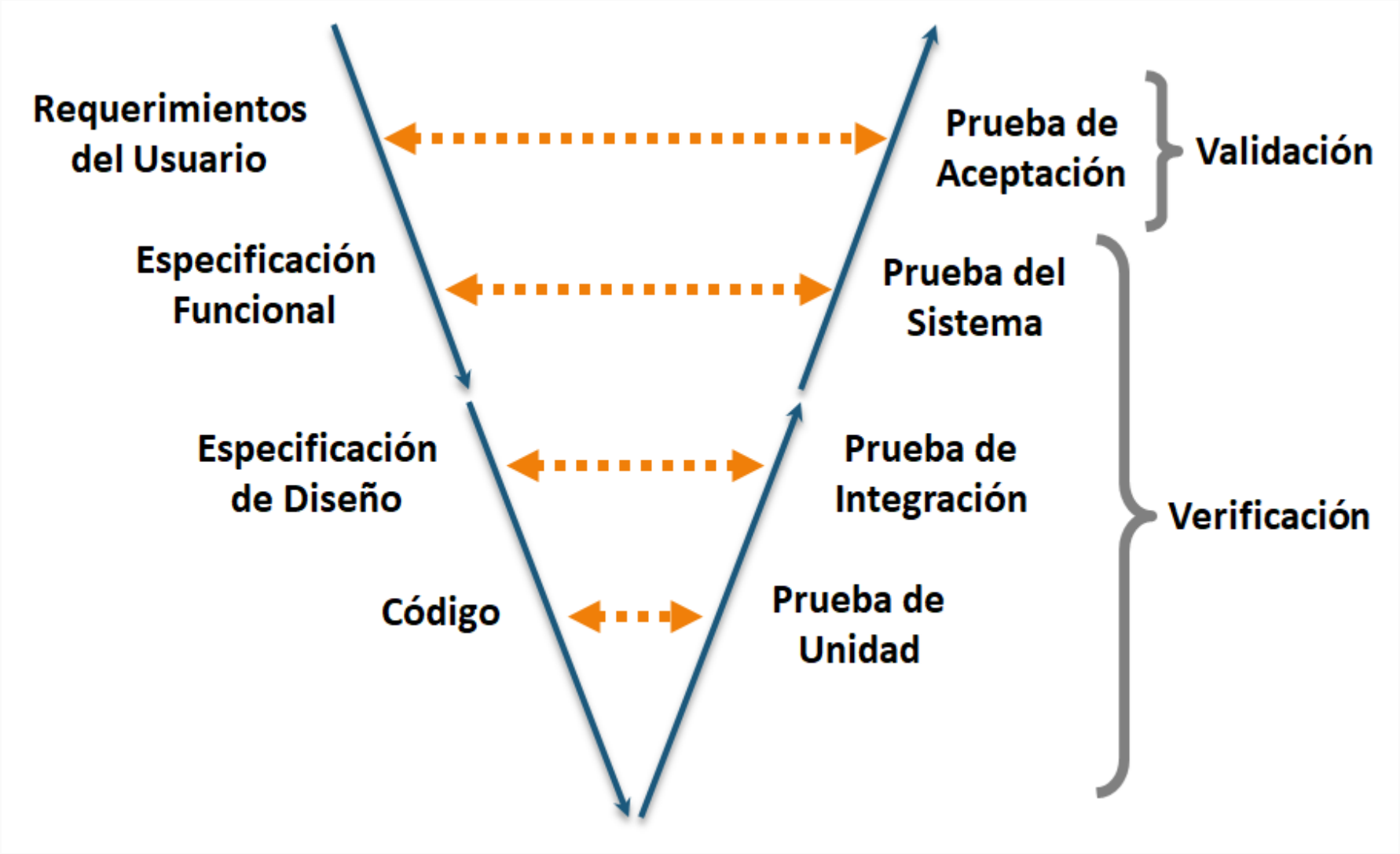
"¿Construimos el producto correctamente?"

Validación

"¿Construimos el producto correcto?"

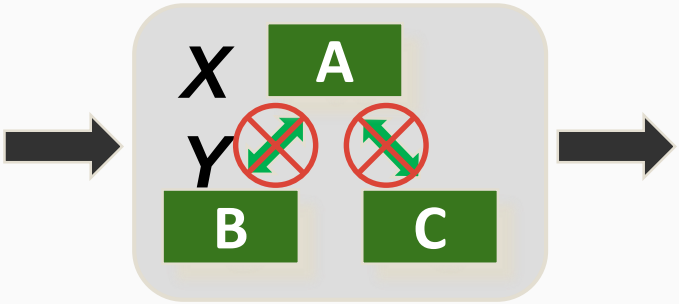
Pruebas Dinámicas

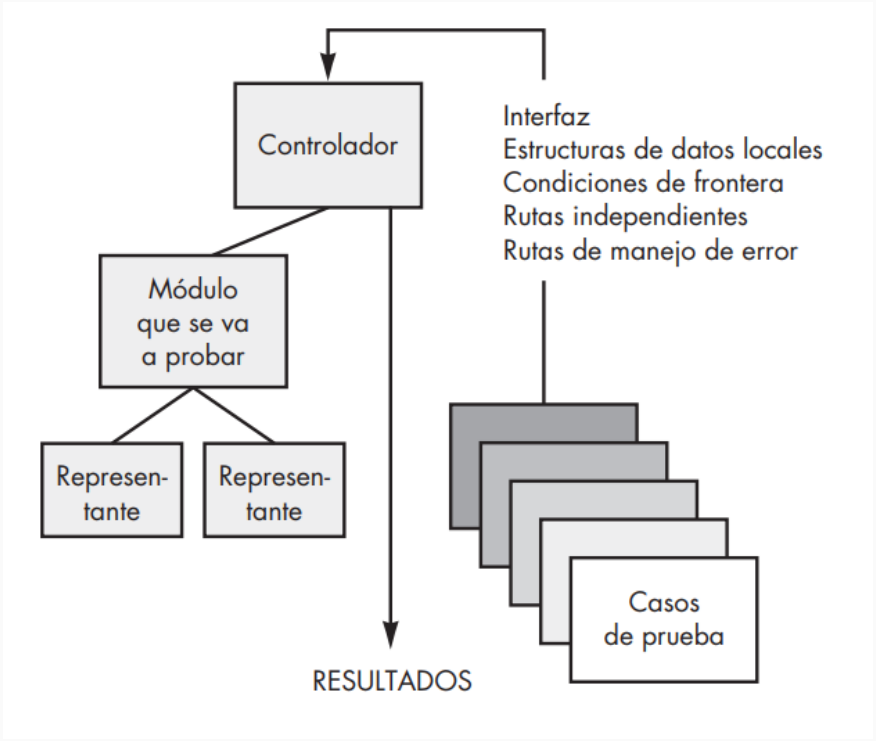
Marco del testeo - Modelo V



Pruebas de Software
Pruebas de Unidad

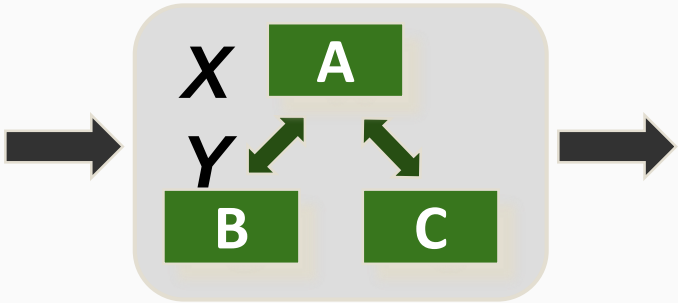
- Se centra en probar cada unidad del software acorde a la implementación. verificando programas o módulos individuales.
- Son ejecutadas, típicamente, en ambientes aislados o especiales.
- Generalmente son ejecutadas por la misma persona que programó el módulo o programa.
- Son los test que suele detectar la mayor cantidad de bugs.
- Gráficamente





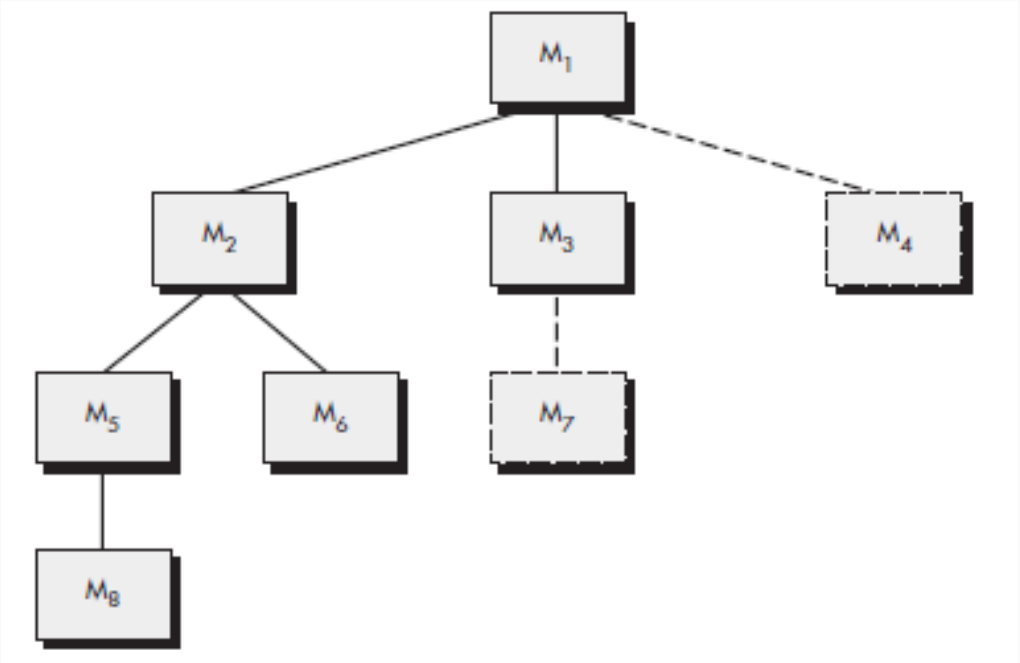
Pruebas de Software
Pruebas de Integración

- Se centra en probar el diseño y construcción de la arquitectura del software.
- Verifican las interfaces entre partes de un sistema (módulos, componentes o subsistemas).
- La integración puede realizarse de 2 maneras:
  - **Integración Total (Big Bang) - No Incremental:** Todos los componentes se combinan por adelantado. El programa se prueba como un todo.
  - **Integración Gradual - Incremental:** El programa se construye y prueba en pequeños incrementos (Integración Ascendente, Integración Descendente, Pruebas de Regresión,...)
- Gráficamente



INTEGRACIÓN DESCENDENTE

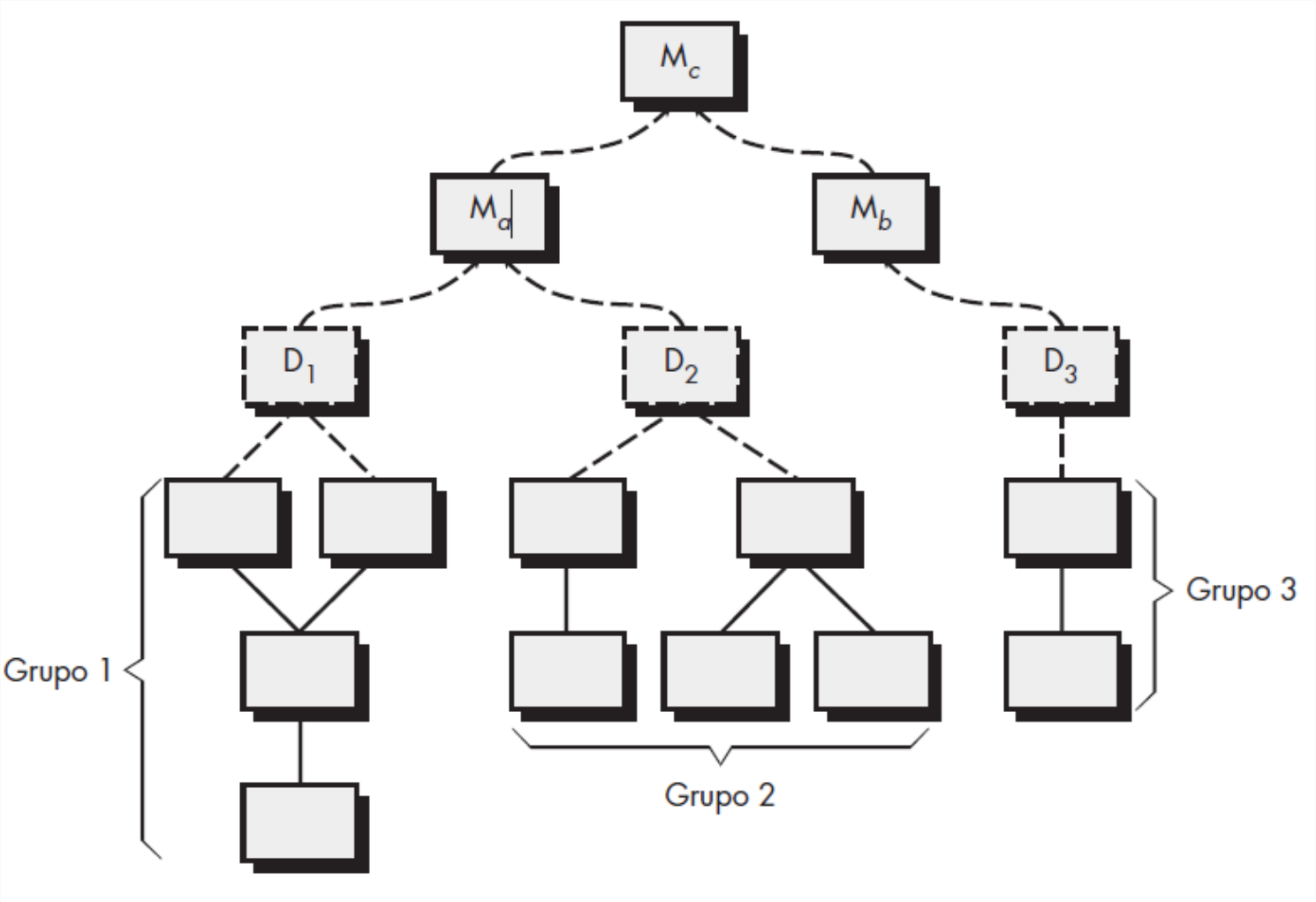
- Los módulos se integran al moverse hacia abajo a en la jerarquía de control, comenzando con el módulo de control principal (programa principal).
- Los módulos subordinados al módulo de control principal se incorporan en la estructura en una forma de primero en profundidad o primero en anchura.
- Estrategias:
  - **Primero en Profundidad**
  - **Primero en Anchura**



INTEGRACIÓN ASCENDENTE

- La prueba de integración ascendente, comienza con la construcción y la prueba con módulos atómicos (componentes en los niveles inferiores dentro de la estructura del programa).
- Los componentes se integran de abajo hacia arriba, la funcionalidad que proporcionan los componentes subordinados en determinado nivel siempre está disponible y se elimina la necesidad de stubs (representantes).

INTEGRACIÓN ASCENDENTE



PRUEBAS DE REGRESIÓN

- Consisten en ejecutar de algún subconjunto de pruebas ya realizadas a fin de asegurar que los cambios no propagaron efectos colaterales indeseados.
- Cada vez que se agrega un nuevo módulo como parte de las pruebas de integración, el software cambia.
- Las pruebas exitosas (de cualquier tipo) dan como resultado el descubrimiento de errores, y los errores deben corregirse.
- Ayudan a garantizar que los cambios realizados (Por pruebas u otra razón) pruebas o por otras razones) no introducen comportamiento no planeado o errores adicionales.

Pruebas de Software

Pruebas de Sistema

- Son una serie de diferentes pruebas cuyo propósito principal es ejercitar por completo el sistema basado en computadora. Verifican el sistema global contra sus objetivos iniciales y verifican que los elementos del sistema se hayan integrado de manera adecuada
- El software se incorpora con otros elementos del sistema (por ejemplo, hardware, personas, información), y se lleva a cabo una serie de pruebas de integración y validación del sistema.
- No se llevan a cabo exclusivamente por parte de ingenieros de software.
- Existen diferentes tipos: Pruebas de Recuperación, Pruebas de Performance (Stress), Pruebas de Seguridad, Pruebas de Rendimiento, Pruebas de Despliegue, Pruebas de Operabilidad

Pruebas de Software
Pruebas de Validación

- Proporciona la garantía final de que el software cumple con todos los requerimientos informativos, funcionales, de comportamiento y de rendimiento.
- Las pruebas se enfocan en las acciones visibles para el usuario y las salidas del sistema reconocibles por el usuario.
- La validación del software se logra a través de una serie de pruebas que demuestran conformidad con los requerimientos.
- Las desviaciones o errores descubiertos en esta etapa en un proyecto rara vez pueden corregirse antes de la entrega calendarizada (Negociar con el cliente)

Pruebas de Software
Pruebas de Validación

- Si el software se desarrolla como un producto que va a ser usado por muchos clientes, no es práctico realizar pruebas de aceptación formales con cada uno de ellos.

PRUEBAS ALFA

- Se lleva a cabo en el sitio del desarrollador por un grupo representativo de usuarios finales.
- Las pruebas alfa se realizan en un ambiente controlado

PRUEBAS BETA

- Se realiza en uno o más sitios del usuario final
- Es una aplicación “en vivo” del software en un ambiente que no puede controlar el desarrollador.

Pruebas de Software

Pruebas Unitarias - Enfoques: Pruebas de Caja Negra y Caja Blanca

- Para Pruebas Unitarias existen dos enfoques:
  - **Pruebas de Caja Blanca** (o Pruebas Estructurales): se basan en el examen cercano de los detalles de procedimiento. Las rutas lógicas a través del software y las colaboraciones entre componentes se ponen a prueba al revisar conjuntos específicos de condiciones y/o bucles.
  - **Pruebas de Caja Negra** (o Pruebas Funcionales): se llevan a cabo en la interfaz del software. Una prueba de caja negra examina algunos aspectos fundamentales de un sistema sin entrar en estructura lógica interna del software
- Combinar ambos enfoques permite lograr mayor fiabilidad

Pruebas Unitarias

Pruebas de Caja Blanca y Caja Negra

Pruebas Unitarias
Pruebas de Caja Blanca

- La prueba de la caja blanca usa la estructura de control del diseño procedural para derivar los casos de prueba
- Idea: confeccionar casos de prueba que garanticen que se verifican todos los caminos independientes
- Verificaciones para cada camino independiente:
    - Probar sus dos facetas desde el punto de vista lógico, es decir, verdadera y falsa
    - Ejecutar todos los bucles en sus límites operacionales
    - Ejercitar las estructuras internas de datos

Pruebas Unitarias

Pruebas de Caja Blanca - Prueba de la ruta básica

- La idea es derivar casos de prueba a partir de un conjunto dado de caminos independientes
- Camino independiente es aquel que introduce por lo menos una sentencia de procesamiento (o valor de condición) que no estaba considerada
- Para obtener el conjunto un conjunto de caminos independientes se construirá el Grafo de Flujo asociado y se calculará su Complejidad Ciclomática

Pruebas Unitarias

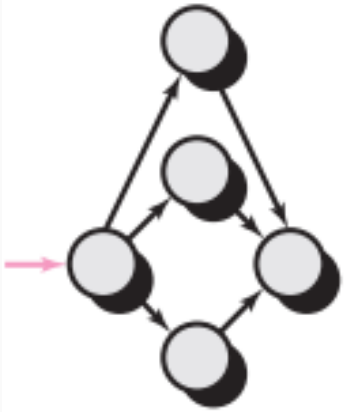
Pruebas de Caja Blanca - Grafo de Flujo

- Los grafos de flujo son la notación utilizada para mostrar el flujo de control lógico.

Secuencia

Si

Caso

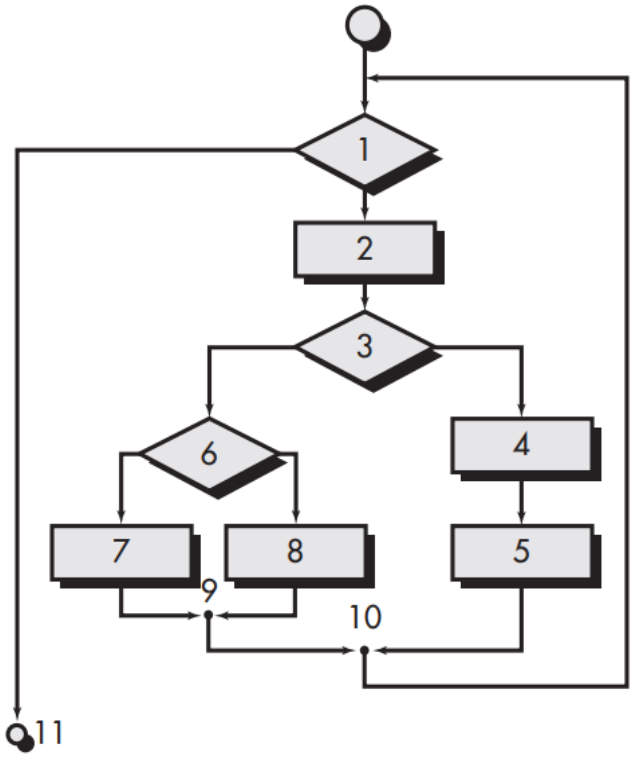


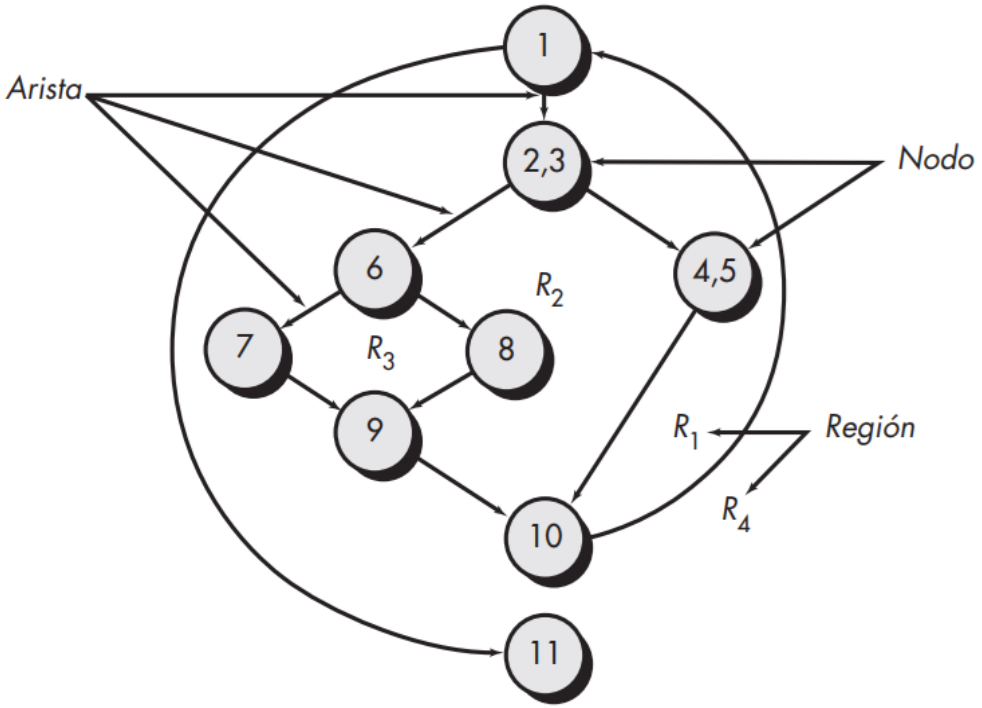
Mientras

Hasta

Pruebas Unitarias

Pruebas de Caja Blanca - Grafo de Flujo



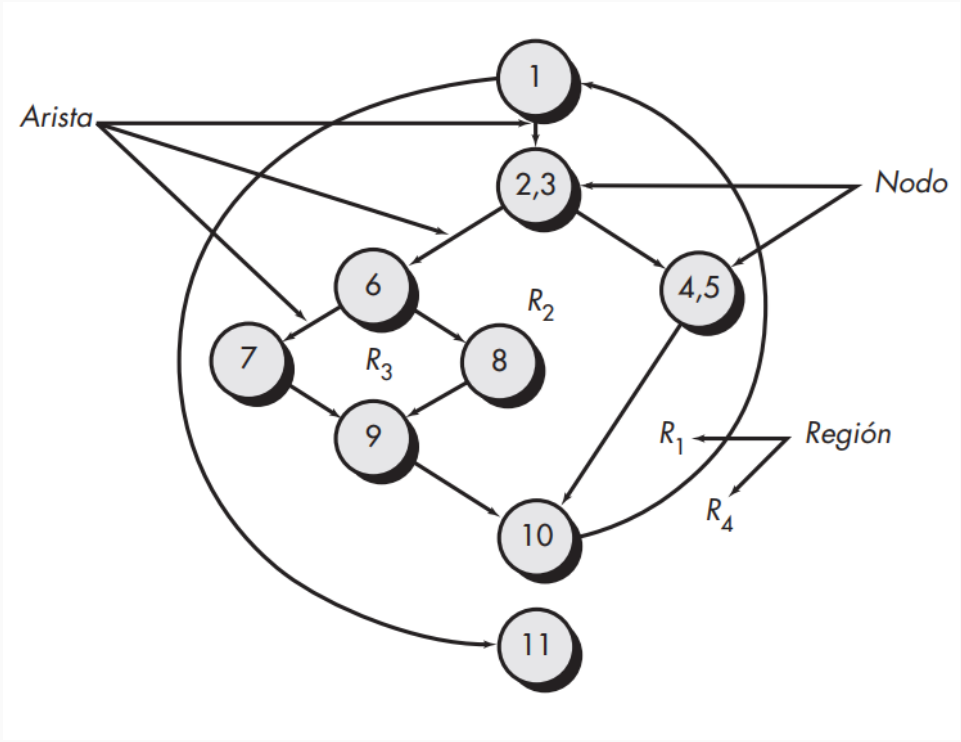


COMPLEJIDAD CICLOMÁTICA

- Complejidad ciclomática de un grafo de flujo V(G) establece el número de caminos independientes
- Puede calcularse de tres formas alternativas:
  - El número de regiones del grafo de flujo
  - V(G) = A - N + 2, donde A es el número de aristas y N es el número de nodos
  - V(G) = P + 1, donde P es el número de nodos predicado (Nodos que contienen una condición)

Pruebas Unitarias

Pruebas de Caja Blanca - Prueba de la ruta básica



COMPLEJIDAD CICLOMÁTICA

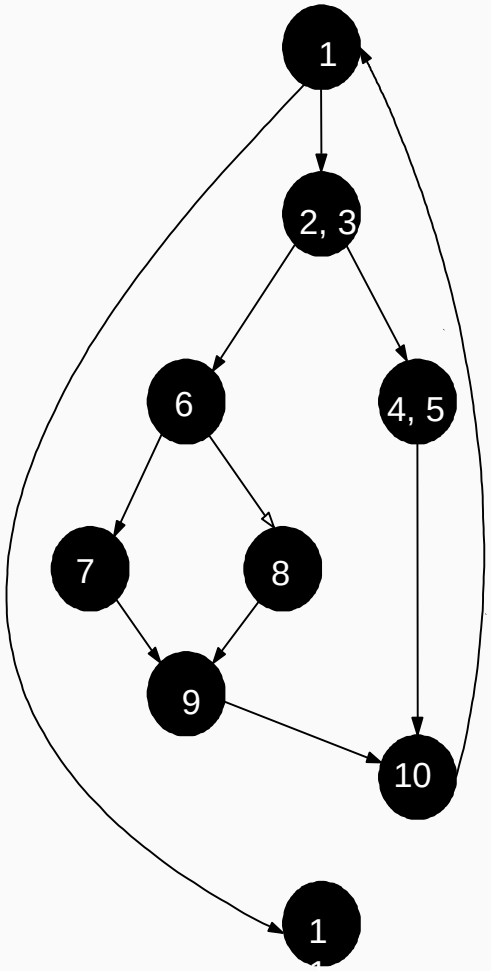
V(G) = 4

- El grafo de la figura tiene cuatro regiones.
- 11 aristas - 9 nodos + 2 = 4
- 3 nodos predicado + 1 = 4

Pruebas Unitarias

Pruebas de Caja Blanca - Prueba de la ruta básica

CONJUNTO BÁSICO



- Un conjunto de caminos independientes

  Camino 1: 1-11
  Camino 2: 1-2-3-4-5-10-1-11
  Camino 3: 1-2-3-6-8-9-10-1-11
  Camino 4: 1-2-3-6-7-9-10-1-11

- El camino

  1-2-3-4-5-10-1-2-3-6-8-9-10-1-11

  No se considera un camino independiente, ya que es simplemente una combinación de caminos ya especificados

- Los cuatro caminos anteriores constituyen un conjunto básico para el grafo

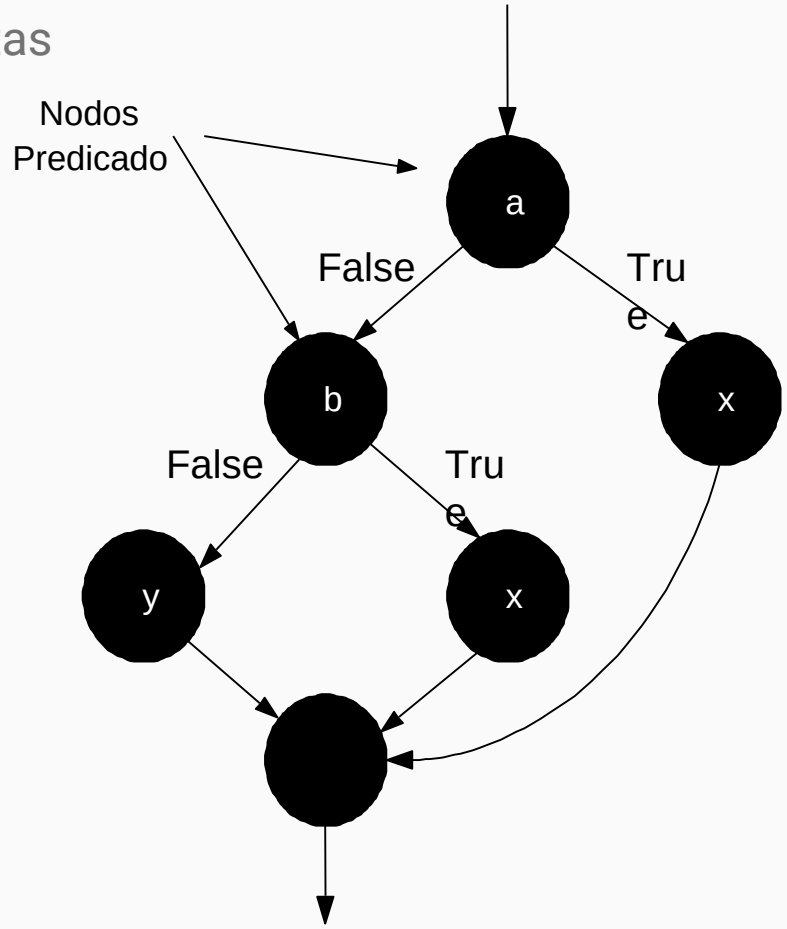
Pruebas Unitarias

Pruebas de Caja Blanca - Prueba de la ruta básica

CONJUNTO BÁSICO

- Tratamiento de Condiciones Compuestas
- Ejemplo :

```
IF a OR b THEN
        procedimiento x
ELSE
    procedimiento y
    ENDIF
```



Pruebas Unitarias

Pruebas de Caja Blanca - Prueba de la ruta básica

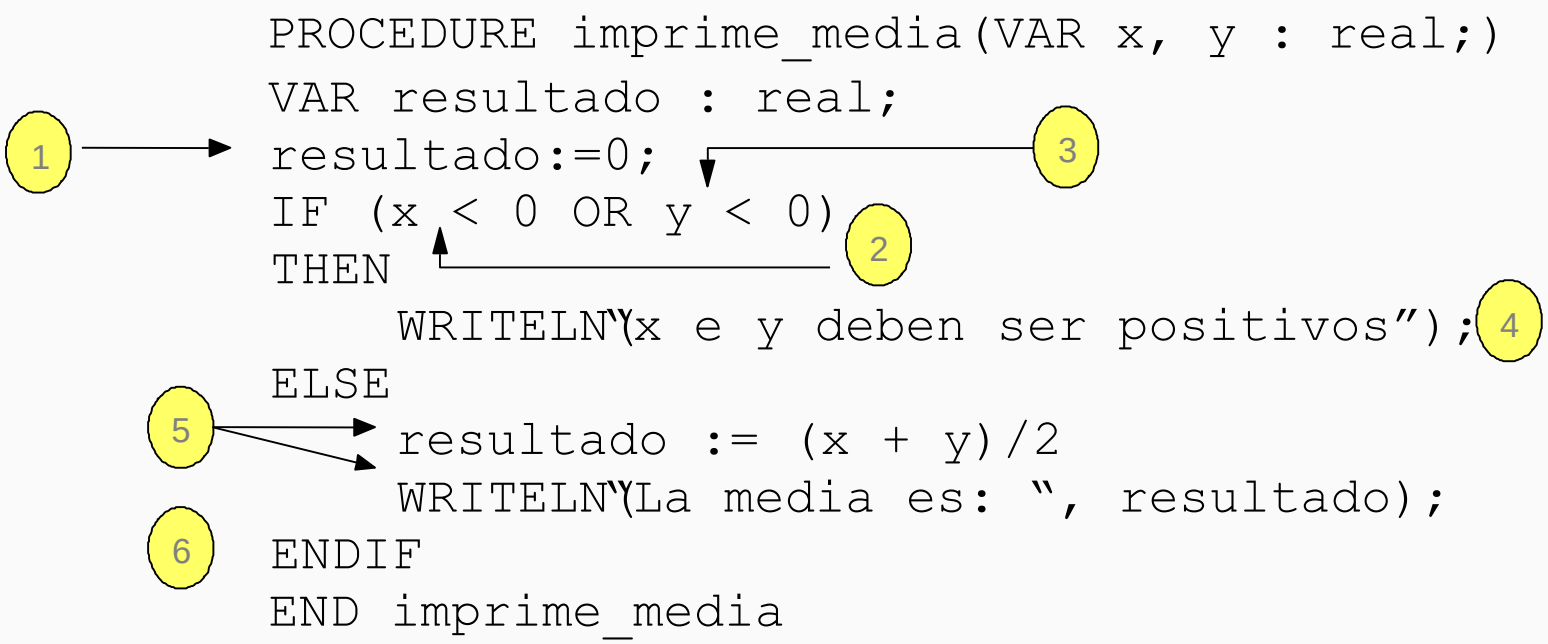
DERIVACIÓN DE CASO DE PRUEBA

- Pasos para realizar las pruebas:
  a. A partir del diseño o del código fuente, dibujar el grafo de flujo asociado
  b. Se calcula la complejidad ciclomática del grafo
  c. Se determina un conjunto básico de caminos independientes
  d. Se preparan los casos de prueba que obliguen a la ejecución de cada camino del conjunto básico

Pruebas Unitarias

Pruebas de Caja Blanca - Prueba de la ruta básica

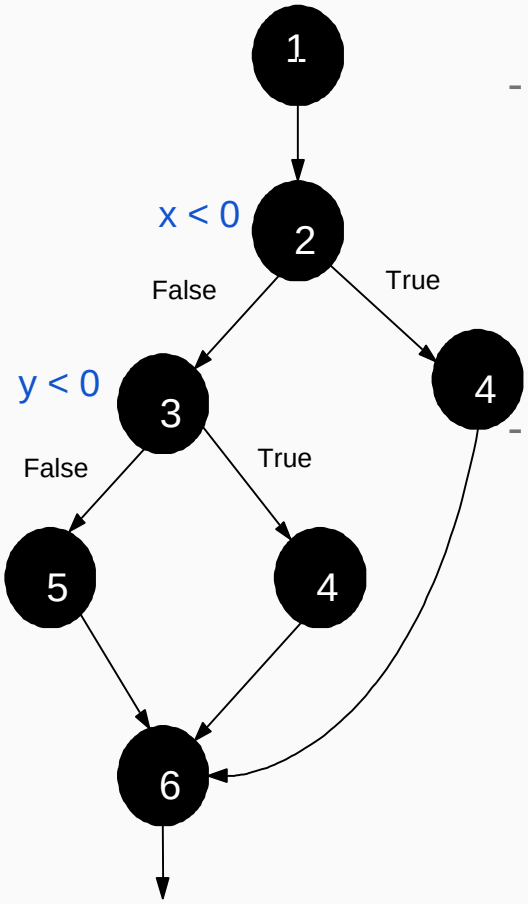
DERIVACIÓN DE CASO DE PRUEBA



Pruebas Unitarias

Pruebas de Caja Blanca - Prueba de la ruta básica

COMPLEJIDAD CICLOMÁTICA



- V(G) = 2+1 = 3. Determinar 3 caminos independientes.
- Por ejemplo:

  Camino 1: 1-2-3-5-6
  Camino 2: 1-2-4-6
  Camino 3: 1-2-3-4-6

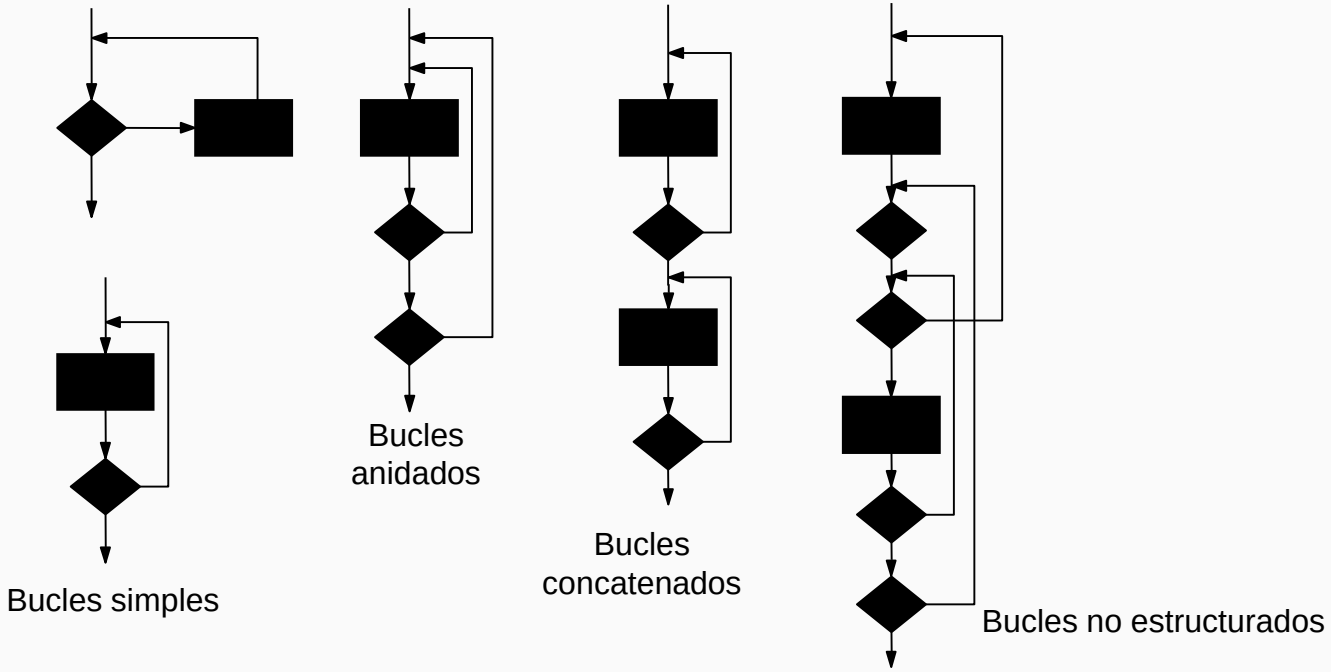
- Casos de prueba para cada camino:
  - Camino 1: Escoger algún x e y tales que se cumpla x >= 0 AND y >= 0
  - Camino 2: Escoger algún x tal que se cumpla x < 0
  - Camino 3: Escoger algún x e y tales que se cumpla x >= 0 AND y < 0

Pruebas Unitarias

Pruebas de Caja Blanca - Prueba de Condiciones

Tipos de errores que pueden aparecer en una condición:

- Existe un error en un operador lógico
- Existe un error en un paréntesis lógico
- Existe un error en un operador relacional
- Existe un error en una expresión aritmética



Pruebas Unitarias

Pruebas de Caja Blanca - Prueba de Condiciones

BUCLES SIMPLES

- n es el número máximo de iteraciones permitidos por el bucle
- Pasar por alto totalmente el bucle
- Pasar una sola vez por el bucle
- Pasar dos veces por el bucle
- Hacer m pasos por el bucle con m < n
- Hacer n-1, n y n + 1 pasos por el bucle

Pruebas Unitarias

Pruebas de Caja Blanca - Prueba de Condiciones

BUCLES ANIDADOS

- Comenzar en el bucle más interior estableciendo los demás bucles en sus valores mínimos.
- Realizar las pruebas de bucle simple para el más interior manteniendo los demás en sus valores mínimos.
- Avanzar hacia fuera confeccionando pruebas para el siguiente bucle manteniendo todos los externos en los valores mínimos y los demás bucles anidados en sus valores típicos
- Continuar el proceso hasta haber comprobado todos los bucles

BUCLES CONCATENADOS

- Siempre que los bucles concatenados sean independientes se puede aplicar lo relativo a bucles simples/anidados. En caso de ser dependientes se evaluarán como bucles anidados

BUCLES NO ESTRUCTURADOS

- Siempre que se usen los mecanismos que aporta la programación estructurada, este tipo de bucles no estarán presentes

Pruebas Unitarias
Pruebas de Caja Negra

- También llamadas Pruebas Funcionales o Pruebas de Comportamiento.
- Se enfocan en los requerimientos funcionales del software.
- Permiten derivar conjuntos de condiciones de entrada que revisarán por completo todos los requerimientos funcionales para un programa.
- Tiende a aplicarse durante las últimas etapas de la prueba
- Intentan encontrar errores de los siguientes tipos:
    - Funciones incorrectas o inexistentes
    - Errores relativos a las interfaces
    - Errores en estructuras de datos o en accesos a bases de datos externas
    - Errores debidos al rendimiento
    - Errores de inicialización o terminación

Pruebas Unitarias

Pruebas de Caja Negra - Partición Equivalente

- La partición equivalente es un método que divide el campo de entrada de un programa en clases de datos que pueden derivarse en casos de pruebas.
- Una condición de entrada es un valor numérico específico, un rango de valores, un miembro de un conjunto de valores o lógica.
- Una clase de equivalencia representa un conjunto de estados válidos y no válidos para una condición de entrada.
- La prueba de partición equivalente se basa en evaluar las clases de equivalencia para una condición de entrada.

Ejemplo:

http://www.lsi.us.es/docencia/get.php?id=401

Pruebas Unitarias

Pruebas de Caja Negra - Análisis de Valores Límite

- Selecciona casos de prueba que ejerciten los valores límite
- Complementa la prueba de partición equivalente. En lugar de realizar la prueba con cualquier elemento de la partición equivalente, se escogen los valores en los bordes de la clase
- Se derivan tanto casos de prueba a partir de las condiciones de entrada como con las de salida

Referencias Bibliográficas

- Ingeniería del software - Un enfoque práctico, Roger S. Pressman, 7° Edición
  - Capitulo 17 - Estrategias de prueba de software
  - Capitulo 18 - Pruebas de aplicaciones convencionales
  - Capitulo 19 - Prueba de aplicaciones orientadas a objetos
  - Capitulo 20 - Prueba de aplicaciones web
- Ingeniería de Software , Ian Sommerville, 9° Edición
  - Capitulo 8 - Pruebas del Software
- Ejemplo “Pruebas Equivalencia”: http://www.lsi.us.es/docencia/get.php?id=401