DISEÑO Y REALIZACIÓN DE PRUEBAS

Índice

Introducción a las pruebas software

- Durante todo el proceso de desarrollo del software es necesario realizar un conjunto de pruebas que permiten verificar que el software que se está creando
  - **es correcto** y
  - **cumple con las especificaciones** impuestas por el usuario

Introducción a las pruebas software

- Las pruebas de software consisten en verificar y validar un producto software antes de su puesta en marcha
- La ejecución de pruebas de un sistema involucra una serie de etapas:
  - planificación de pruebas
  - diseño y construcción de los casos de prueba
  - definición de los procedimientos de prueba
  - ejecución de las pruebas
  - registro de resultados obtenidos
  - registro de errores encontrados
  - depuración de los errores
  - informe de los resultados obtenidos

Técnicas de diseño de Casos de Prueba

- Para llevar a cabo un caso de prueba es necesario:
  - definir las precondiciones y postcondiciones
  - identificar unos valores de entrada
  - conocer el comportamiento que debería tener el sistema ante dichos valores
- Tras realizar ese análisis e introducir dichos datos en el sistema se observa si su comportamiento es el previsto o no y porqué
- De esta forma se determina si el sistema ha pasado la prueba o no

Técnicas de diseño de Casos de Prueba

Para llevar a cabo el diseño de casos de prueba se utilizan **dos técnicas** o enfoques:

- **Prueba caja blanca:** se centran en validar la estructura interna del programa
  - se necesitan conocer los detalles procedimentales del código
- **Prueba caja negra:** se centran en validar los requisitos funcionales sin fijarse en el funcionamiento interno del programa
  - se necesita saber la funcionalidad que el código ha de proporcionar

Técnicas de diseño de Casos de Prueba

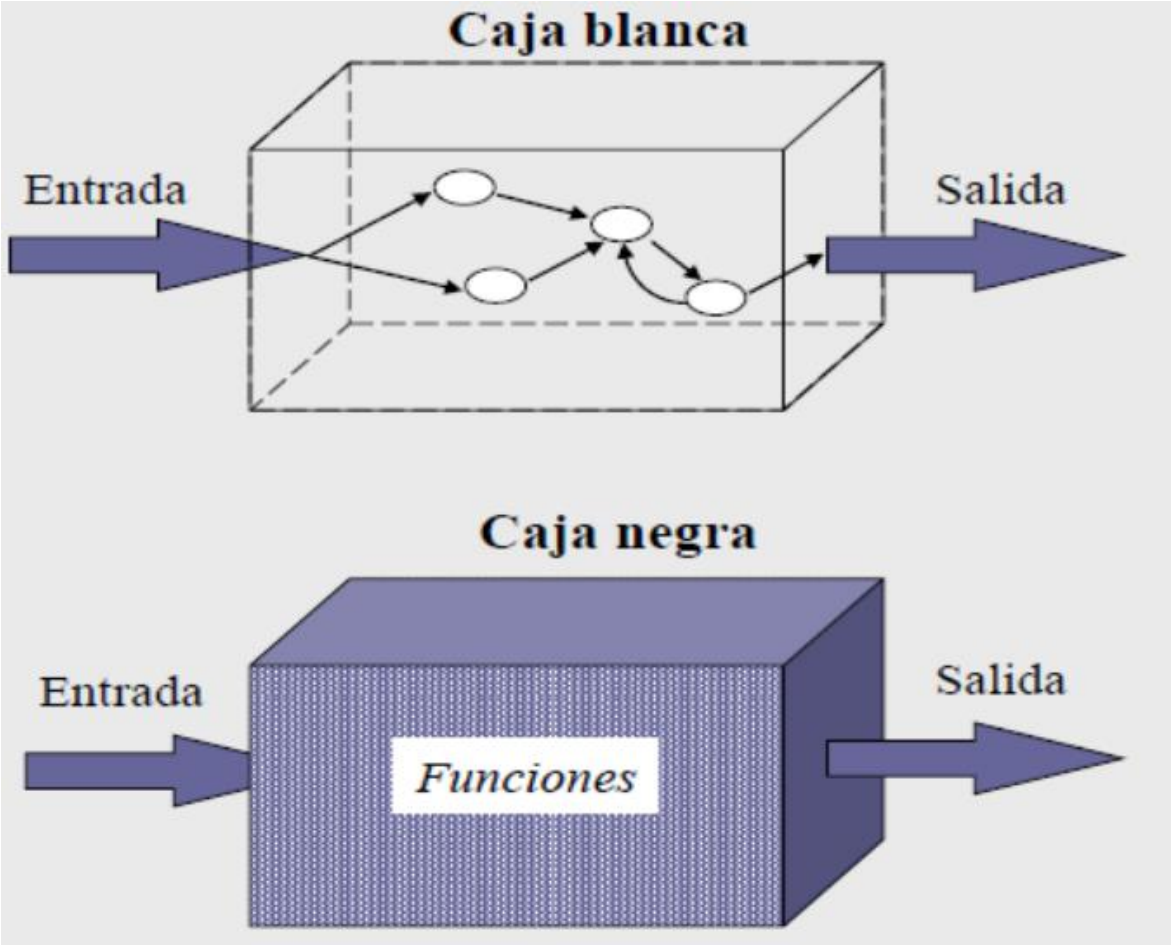
Prueba de la **CAJA BLANCA**
(Enfoque estructural)

- Estudio de la **estructura INTERNA** (implementación)
- Prueba ideal: Probar TODOS los posibles caminos de ejecución, a través de las instrucciones del código
- Estas pruebas solo las puede realizar el programador

Prueba de la **CAJA NEGRA**
(Enfoque funcional)

- Estudio de la **especificación de las funciones, la entrada y la salida** para derivar los casos
- Prueba ideal: probar todas las posibles entradas y salidas del programa
- Estas pruebas las puede realizar también el cliente como prueba de aceptación

Técnicas de diseño de Casos de Prueba



Pruebas de caja blanca

- También llamadas *pruebas estructurales* o *de caja de cristal*
- Se basan en el minucioso examen de los detalles procedimentales del código de la aplicación
- Mediante esta técnica se pueden obtener casos de prueba que:
  - Garanticen que se ejecutan al menos una vez todos los caminos independientes de cada módulo
  - Ejecuten todas las sentencias al menos una vez
  - Ejecuten todas las decisiones lógicas en su parte verdadera y en su parte falsa
  - Ejecuten todos los bucles en sus límites
  - Utilicen todas las estructuras de datos internas para asegurar su validez
- Una de las técnicas utilizadas para desarrollar los casos de prueba de caja blanca es la **prueba del camino básico**

Pruebas de caja negra

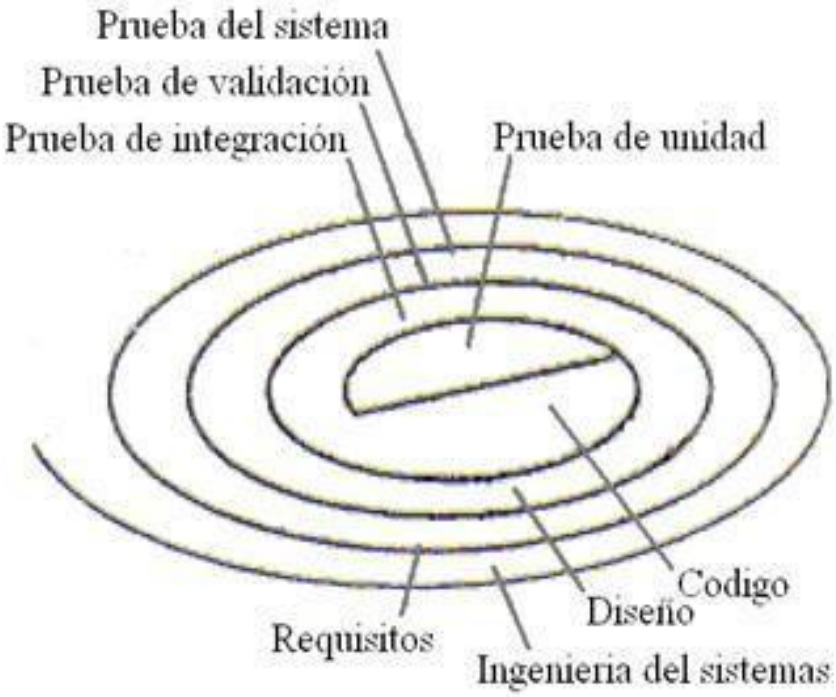
- También llamadas *pruebas de comportamiento*
- Se llevarán a cabo sobre la interfaz del software
  No es necesario conocer la estructura interna del programa
- Se pretende obtener casos de prueba que demuestren que las funciones del software son operativas:
  - Las salidas que devuelve la aplicación son las esperadas en función de las entradas

Pruebas de caja negra

- Con este tipo de pruebas se intenta encontrar errores de las siguientes categorías:
  - Funcionalidades incorrectas o ausentes
  - Errores de interfaz
  - Errores en estructuras de datos o en accesos a bases de datos externas
  - Errores de rendimiento
  - Errores de inicialización y finalización
- Existen diferentes técnicas para confeccionar los casos de prueba de caja negra, algunos son: Clases de equivalencia, Análisis de los valores límite, Métodos basados en grafos, Pruebas de comparación, etc.

Estrategias de pruebas del software

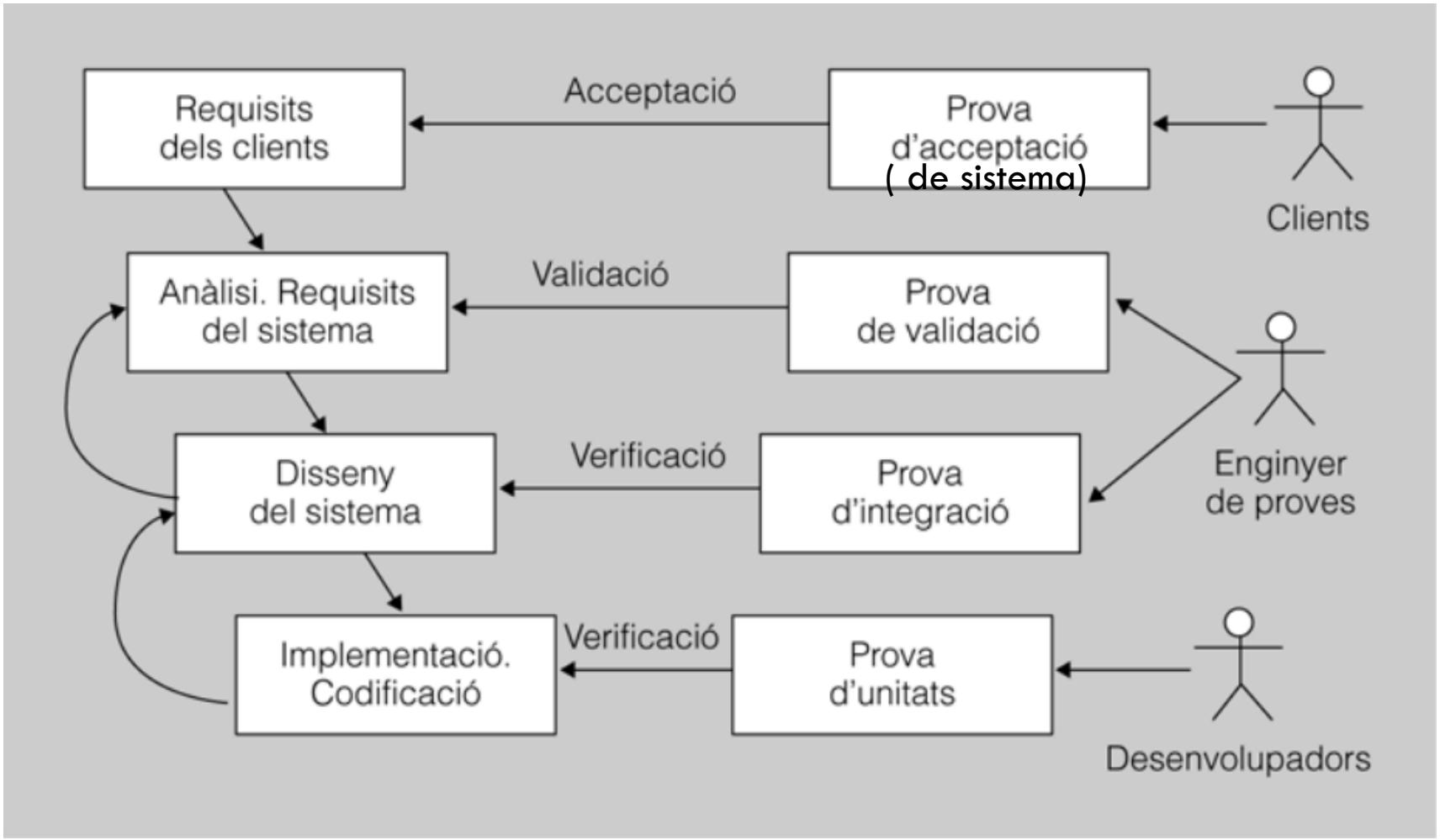
- La estrategia de prueba del software se puede ver en el contexto de una espiral:



Estrategias de pruebas del software

- En el vértice de la espiral comienza la **prueba de unidad**
  - Se centra en la unidad más pequeña del software, el módulo tal como está implementado en código fuente
- La prueba avanza para llegar a la **prueba de integración**
  - Se toman los módulos probados mediante la prueba de unidad y se construye una estructura de programa que esté de acuerdo con lo que dicta el diseño
- La espiral avanza llegando a la **prueba de validación**
  - Prueba del software en el entorno real de trabajo con intervención del usuario final
  - Se validan los requisitos establecidos en el análisis comparándolos con el sistema que ha sido construido
- Finalmente se llega a la **prueba del sistema**
  - Verifica que cada elemento encaja de forma adecuada y se alcanza la funcionalidad y rendimiento total
  - Se prueba como un todo el software y otros elementos del sistema

Estrategias de pruebas del software



Prueba de unidad

- En este nivel se prueba cada unidad o módulo con el objetivo de eliminar errores en la interfaz y en la lógica interna
- Esta actividad utiliza técnicas de caja negra y caja blanca, según convenga para lo que se desea probar
- Se realizan pruebas sobre:
  - La interfaz del módulo, para asegurar que la información fluye adecuadamente
  - Las estructuras de datos locales, para asegurar que mantienen su integridad durante todos los pasos de ejecución del programa
  - Las condiciones límite, para asegurar que funciona correctamente en los límites establecidos durante el proceso
  - Todos los caminos independientes de la estructura de control, con el fin de asegurar que todas las sentencias se ejecutan al menos una vez
  - Todos los caminos de manejo de errores

Prueba de unidad

- Algunas de las herramientas que se utilizan para pruebas unitarias son: Junit, CPPUnit, PHPUnit, etc.

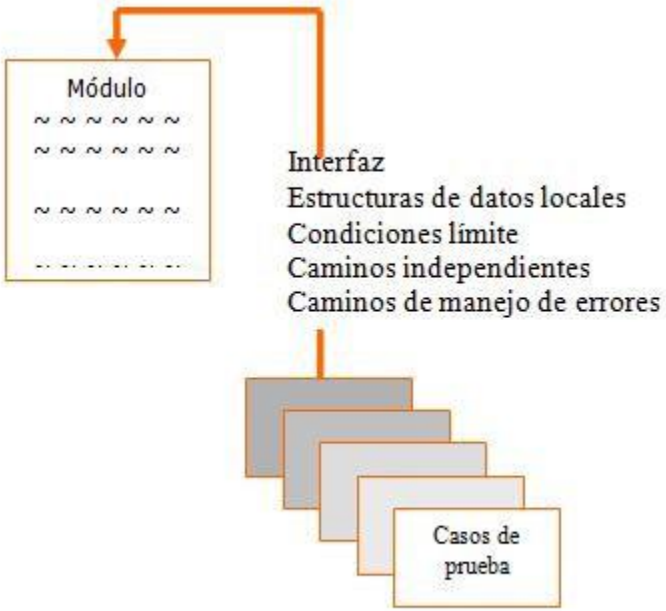


Figura. Prueba de Unidad. (Fuente: Roger Pressman, 2002)

Prueba de Unidad

- Para las pruebas de unidad (o de código) se van a mostrar diferentes técnicas que dependerán del tipo de enfoque utilizado:
  - De caja blanca
    - **Prueba del camino básico**
  - De caja negra
    - **Partición o clases de equivalencia**
    - **Análisis de los valores límite**

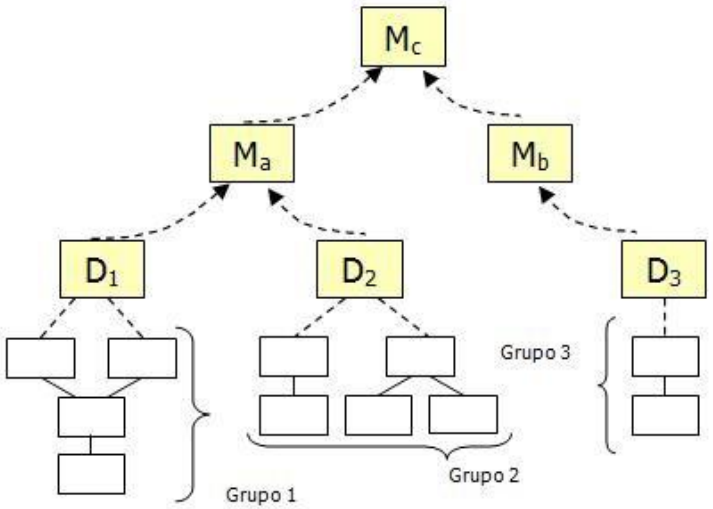
Pruebas de Integración

- En este tipo de prueba se observa cómo interaccionan los distintos módulos
- Se ha probado que todos los módulos funcionan por separado, ahora se valida la integración, se trata de comprobar si funcionan juntos
- Existen dos enfoques:
  - **Integración no incremental o *big bang*:**
    - Se prueba cada módulo por separado y luego se combinan todos de una vez y se prueba todo el programa completo
    - En este enfoque se encuentran gran cantidad de errores y la corrección se hace difícil

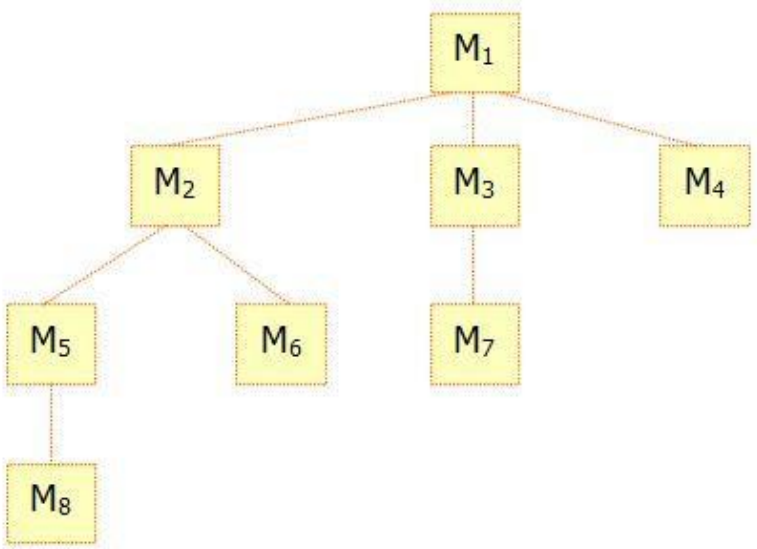
Pruebas de Integración

- **Integración incremental:**
  - el programa completo se va construyendo y probando en pequeños segmentos
  - los errores son más fáciles de localizar
  - Se dan dos estrategias:
    - **Ascendente:** la construcción y prueba del programa empieza desde los módulos de los niveles más bajos de la estructura del programa
    - **Descendente:** la integración comienza en el módulo principal moviéndose hacia abajo por la jerarquía de control
      - Dos tipos: en profundidad y en anchura

Pruebas de Integración



Integración ascendente



Integración descendente

Pruebas de Validación

- La validación se consigue cuando el software funciona de acuerdo con las expectativas razonables del cliente definidas en el documento de especificación de requisitos del software
- Se llevan a cabo una serie de pruebas de caja negra para comprobar se cumplen requisitos:

Pruebas de Validación

- **Prueba Alfa**
  - Se lleva a cabo por el cliente o usuario en el lugar de desarrollo
  - El cliente utiliza el software de forma natural bajo la observación del desarrollador que irá registrando errores y problemas de uso
- **Prueba Beta**
  - Se lleva a cabo por los usuarios finales del software en su lugar de trabajo. El desarrollador no está presente
  - El usuario registra todos los problemas que encuentra e informa al desarrollador
  - Como resultado de los problemas el desarrollador lleva a cabo las modificaciones y prepara una nueva versión del producto

Pruebas del Sistema

- La prueba del sistema está formada por un conjunto de pruebas cuya misión es ejercitar profundamente el software
- Son las siguientes:
  - **Pruebas de recuperación**
    - En este tipo de prueba se fuerza el fallo del software y verifica que la recuperación se lleva a cabo apropiadamente
  - **Pruebas de seguridad**
    - En esta prueba se intenta verificar que el sistema está protegido contra accesos ilegales
  - **Pruebas de resistencia (Stress)**
    - Trata de enfrentar el sistema con situaciones que demandan gran cantidad de recursos.
    - Ejemplo: Diseñar caso de prueba que demande gran cantidad memoria, frecuencia de datos...

Documentación para las pruebas

- El estándar IEEE 829-1998 describe el conjunto de documentos que pueden producirse durante el proceso de prueba
- Son los siguientes:
- **Plan de pruebas**
  - Describe el alcance, el enfoque, los recursos y el calendario de las actividades de prueba. Identifica los elementos a probar, las características que se van a probar, las tareas que se van a realizar, el personal responsable de cada tarea y los riesgos asociados al plan
- **Especificaciones de prueba**
  - Incluye diseño de la prueba, especificación de los casos de prueba y especificación de los procedimientos de prueba
- **Informes de pruebas**
  - Incluye informe que identifica los elementos que están siendo probados, registro de pruebas, informe de incidencias de prueba e informe resumen de las actividades de prueba