05BM - Fundamentos de Ingeniería del Software.

Bloque 2.- Desarrollo de Sw.

Tema 5. Prueba de Software.

Departamento de Informática y Sistemas
Facultad de Informática
Campus Universitario de Espinardo - Murcia

Asignatura: Fundamentos de Ingeniería del Software
Titulación: Ingeniería Técnica de Informática de Gestión
Curso Académico: 2005-2006
Curso: 3º
Cuatrimetres: Primero
Créditos: 6(3+3)
Página Web: dis.um.es/~lopezquesada
Profesor: Juan Antonio López Quesada
Departamento: Informática y Sistemas

Tema 5. Prueba de Software. Estructura.

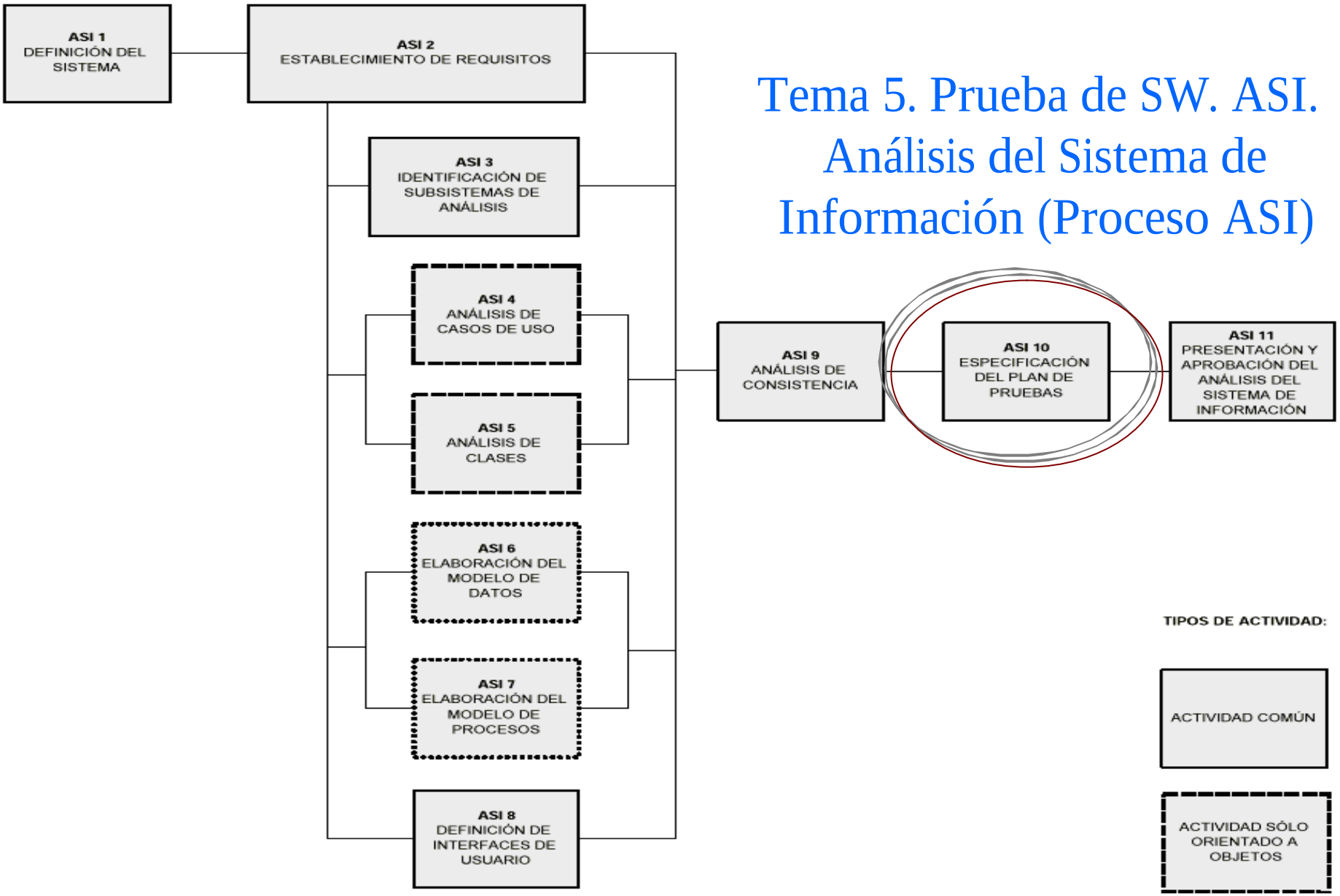
- Objetivos de la prueba
- Importancia de la prueba
- Principios de la prueba
- El proceso de prueba
- Métodos de diseño de casos de prueba
- Enfoque estructural
  - Prueba del camino básico
    - Notación de grafo de flujo
    - Complejidad ciclomática
    - Derivación de los casos de prueba
  - Prueba de bucles
- Enfoque funcional
  - Particiones o clases de equivalencia
  - Análisis de Valores Límite (AVL)
- Prueba de interfaces gráficas de usuario
- Estrategias de prueba del software
  - Relación entre productos de desarrollo y niveles de prueba
  - Organización para la prueba del software
  - Prueba de unidad
  - Prueba de integración
  - Integración incremental descendente
  - Integración incremental ascendente
  - Módulos críticos
  - Prueba de aceptación

Tema 5. Prueba de Software. Bibliografía

- (Piattini et *al.* 96) Capítulo 12
- (Pressman 02) Capítulos 17 y 18
- (MAP 95) Ministerio de Administraciones Públicas. Guía de Técnicas de Métrica y Guía de Referencia. v.2.1. 1995

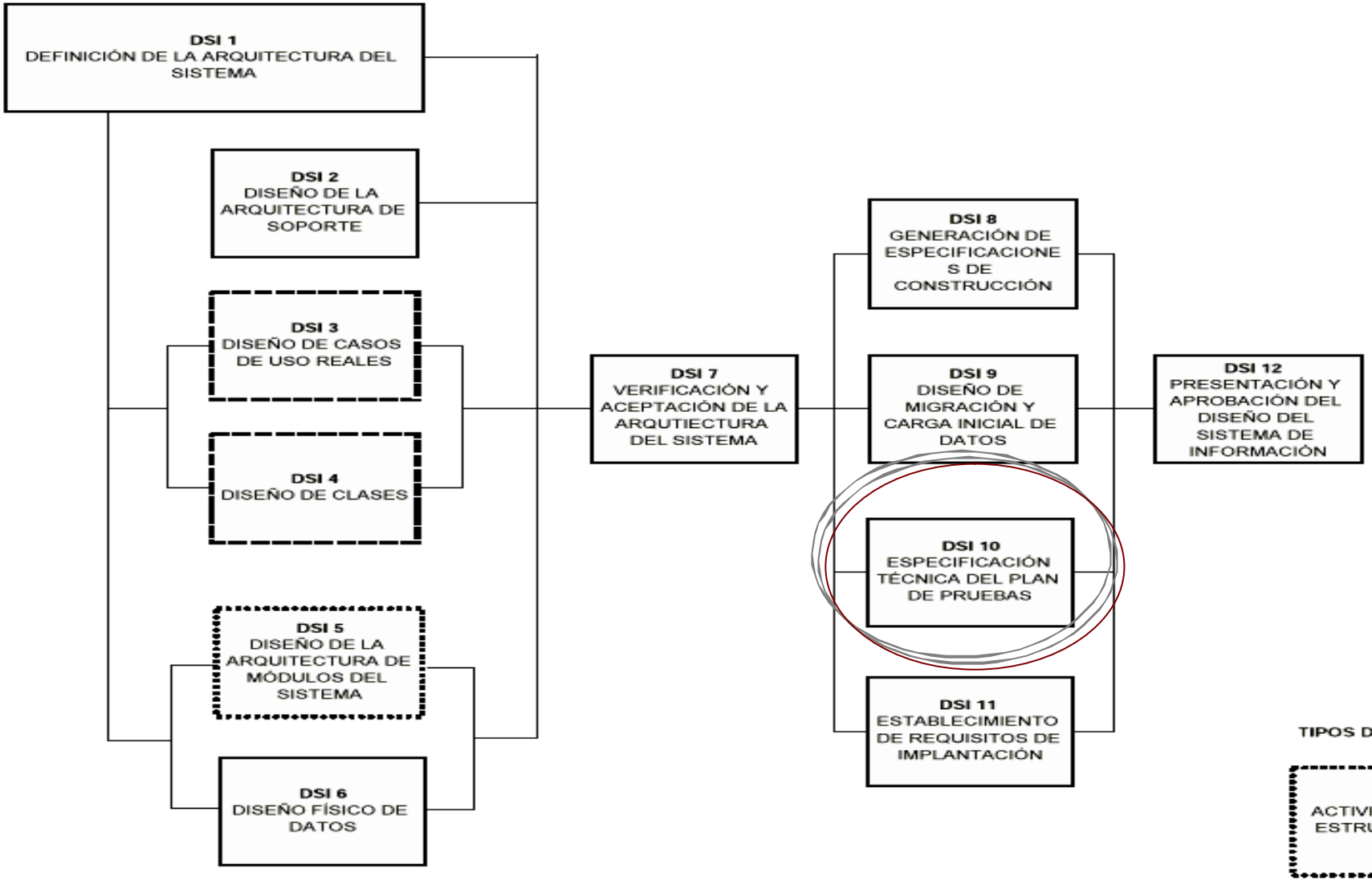
- *Métrica 3: Análisis del Sistema de Información (Proceso ASI).*
- *Métrica 3: Diseño del Sistema de Información (Proceso DSI).*

Tema 5. Prueba de SW. ASI. Análisis del Sistema de Información (Proceso ASI)



Tema 5. Prueba de SW. Métrica 3 (II).
Análisis del Sistema de Información (Proceso ASI)
ASI 10: Especificación del plan de prueba

- Se inicia la definición del plan de pruebas.
  - Se definen también las *pruebas de aceptación*.



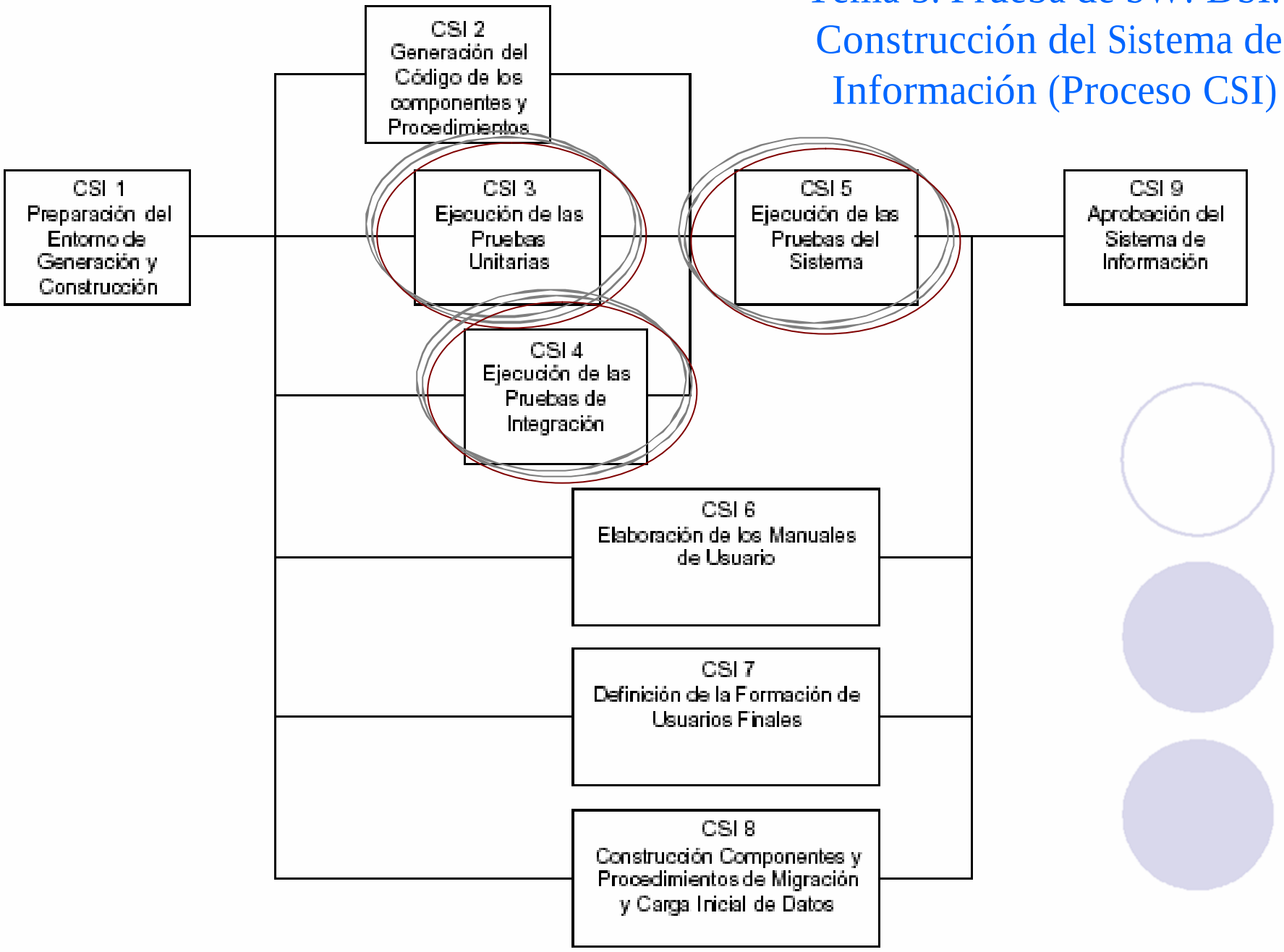
Tema 5. Prueba de SW. DSI. Diseño del Sistema de Información (Proceso DSI)

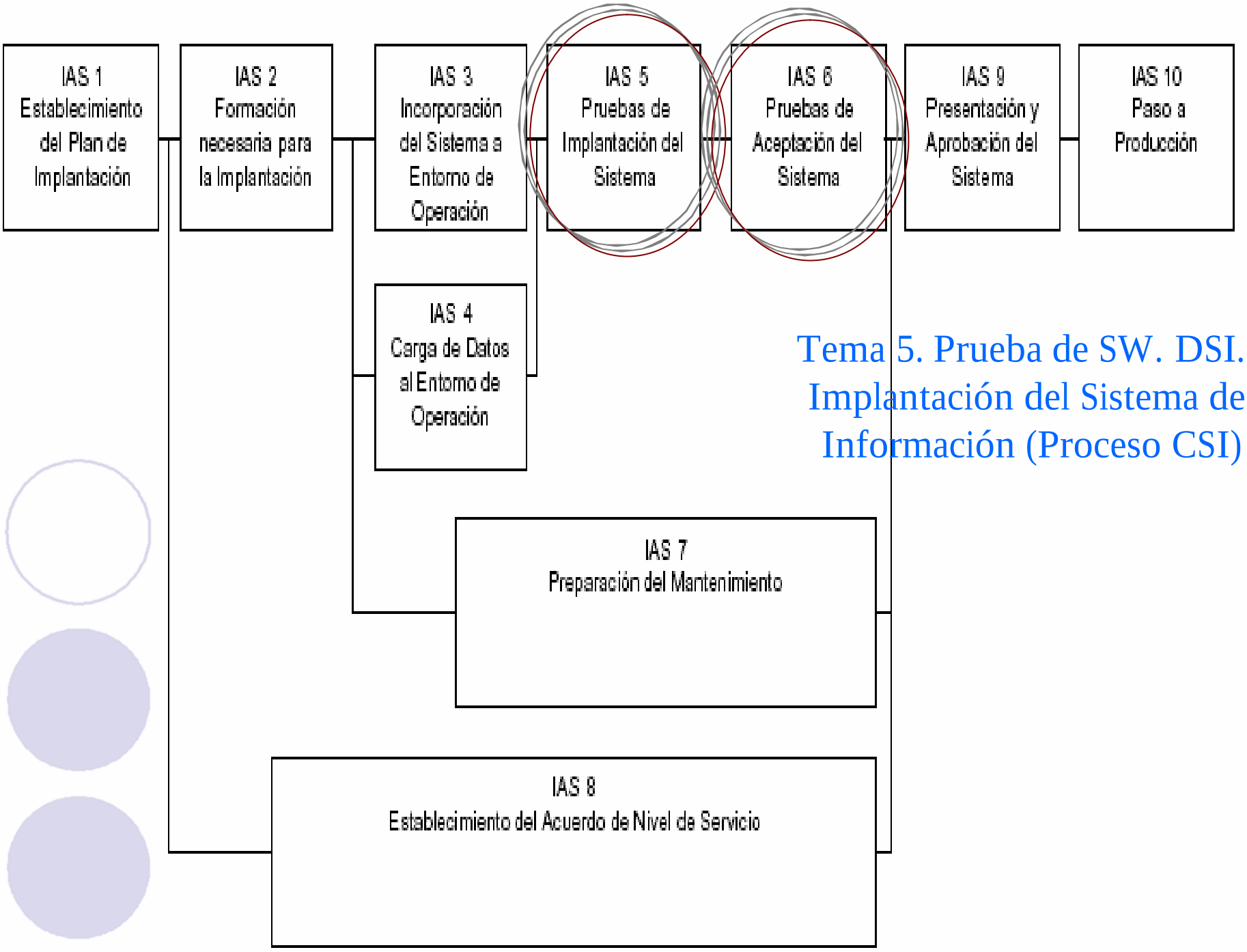
Tema 5. Prueba de SW. Métrica 3 (II).
Diseño del Sistema de Información (Proceso DSI)
DSI 10: Especificación técnica del plan de pruebas

- Se especifica en detalle el plan de pruebas del SI, para los niveles de prueba:
  - *Pruebas unitarias.*
  - *Pruebas de Sistema.*
  - *Pruebas de integración.*
  - *Pruebas de implantación.*
  - *Pruebas de aceptación.*
- Se especifica el entorno de las pruebas.
- Se definen los *CASOS DE PRUEBA*.

- Las pruebas *unitarias, de integración y del sistema* se llevan a cabo en el proceso *Construcción del Sistema de Información* (CSI), mientras que las pruebas de implantación y aceptación se realizan en el proceso *Implantación y Aceptación del Sistema* (IAS).

Tema 5. Prueba de SW. DSI. Construcción del Sistema de Información (Proceso CSI)





Tema 5. Prueba de SW. DSI. Implantación del Sistema de Información (Proceso CSI)

Tema 5. Prueba de Software. Prueba del Software

¿Labor destructiva y rutinaria?

Objetivos de las pruebas:

1. La prueba es el proceso de **ejecución** de un programa con la intención de descubrir un error.
2. Un buen *caso de prueba* es aquel que tiene una alta probabilidad de descubrir un error no encontrado hasta entonces.
3. Una prueba tiene éxito si descubre un error no detectado hasta entonces.

- No sólo se prueba el código: tb. documentación y ayuda.

Tema 5. Prueba de Software. Prueba del Software

- Índice de la fiabilidad del software:

  se comporta de acuerdo a las especificaciones y requisitos de rendimiento.

  "La prueba no puede asegurar la ausencia de defectos: sólo puede demostrar que existen defectos en el software".

- No es una actividad secundaria:
  - 30-40% del esfuerzo de desarrollo
  - En aplicaciones críticas (p.ej. control de vuelo, reactores nucleares),

    ¡de 3 a 5 veces más que el resto de pasos juntos de la ingeniería del software!
  - El coste aproximado de los errores del software (*bugs*) para la economía americana es el equivalente al 0,6% de su PIB, unos 60.000 millones de dólares

    ⇒ ¡Evitar bichos puede ser un gran negocio!

Tema 5. Prueba de Software. Principios de la Prueba

- A todas las pruebas se les debería poder hacer un seguimiento hasta los requisitos de los clientes *(trazabilidad).*
- Las pruebas deberían planificarse (DSI) antes de que empiecen (CSI/IAS).
- El principio de Pareto es aplicable a la prueba del software *(“donde hay un defecto, hay otros”)*.
- Las pruebas deberían empezar por “lo pequeño” y progresar hacia “lo grande”. *Clasificación de as pruebas sw.*
- No son posibles las pruebas exhaustivas.
- Para ser más efectivas, las pruebas deberían ser conducidas por un equipo independiente.
- Se deben evitar los casos de prueba no documentados ni diseñados con cuidado.
- No deben realizarse planes de prueba suponiendo que prácticamente no hay defectos en los programas y, por tanto, dedicando pocos recursos a las pruebas.

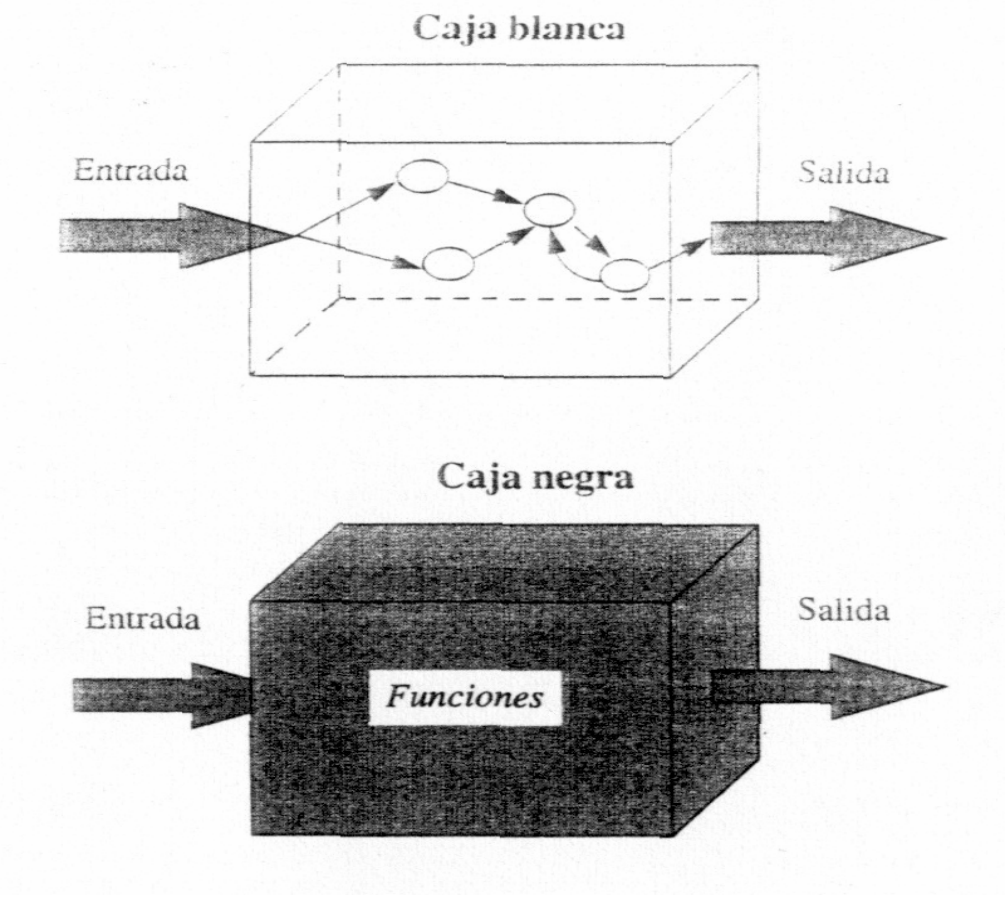
Tema 5. Prueba de Software. Diseño de casos de Pruebas

- Seguridad total: prueba exhaustiva *(no practicable).*
- Objetivo: Conseguir confianza aceptable en que se encontrarán todos los defectos existentes, sin consumir una cantidad excesiva de recursos.
- *"Diseñar las pruebas que tengan la mayor probabilidad de encontrar el mayor número de errores con la mínima cantidad de esfuerzo y tiempo posible."*

Tema 5. Prueba de Software.

Diseño de casos de prueba.

Enfoques principales.



Tema 5. Prueba de Software.

Enfoques principales.

a) *Enfoque estructural o de caja blanca (transparente*):

- Se centra en la estructura interna del programa para elegir los casos de prueba.
- La prueba exhaustiva consistiría en probar todos los posibles caminos de ejecución.
- nº caminos lógicos ↑↑ (buscar los más importantes).

b) *Enfoque funcional o de caja negra*:

- Para derivar los casos, se estudia la especificación de las funciones, la entrada y la salida.

- *No son excluyentes.*

Tema 5. Prueba de Software.

Prueba de Caja Blanca.

¿Porqué no dedicar todo el esfuerzo a probar que se cumplen los requisitos?

¿Porqué usar pruebas de caja blanca?

- Los errores lógicos y las suposiciones incorrectas son inversamente proporcionales a la probabilidad de que se ejecute un camino del programa.
- A veces creemos que un camino lógico tiene pocas posibilidades de ejecutarse cuando puede hacerlo de forma normal.

"Los errores se esconden en los rincones y se aglomeran en los límites" (Beizer 90)

Tema 5. Prueba de Software.

Prueba del camino básico.

(Mc Cabe 76)

Objetivos:

1. Obtener una medida de la complejidad lógica de un diseño procedimental.

   Þ *Complejidad ciclomática de Mc Cabe*

2. Usar esa medida como guía para la definición de un *conjunto básico de caminos* de ejecución.

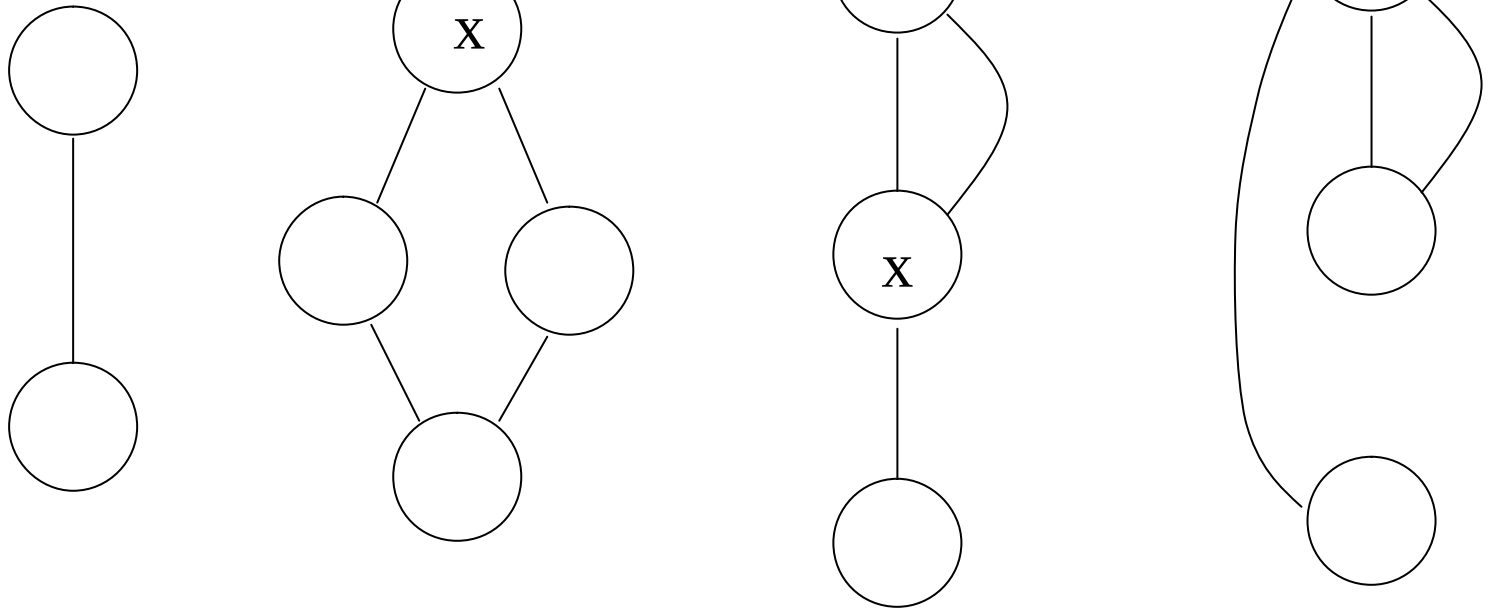
- Los casos de prueba obtenidos garantizan que durante la prueba se ejecuta al menos una vez cada sentencia del programa (cobertura de sentencias).
- Se puede automatizar. *Ejemplo:*

Rational. software

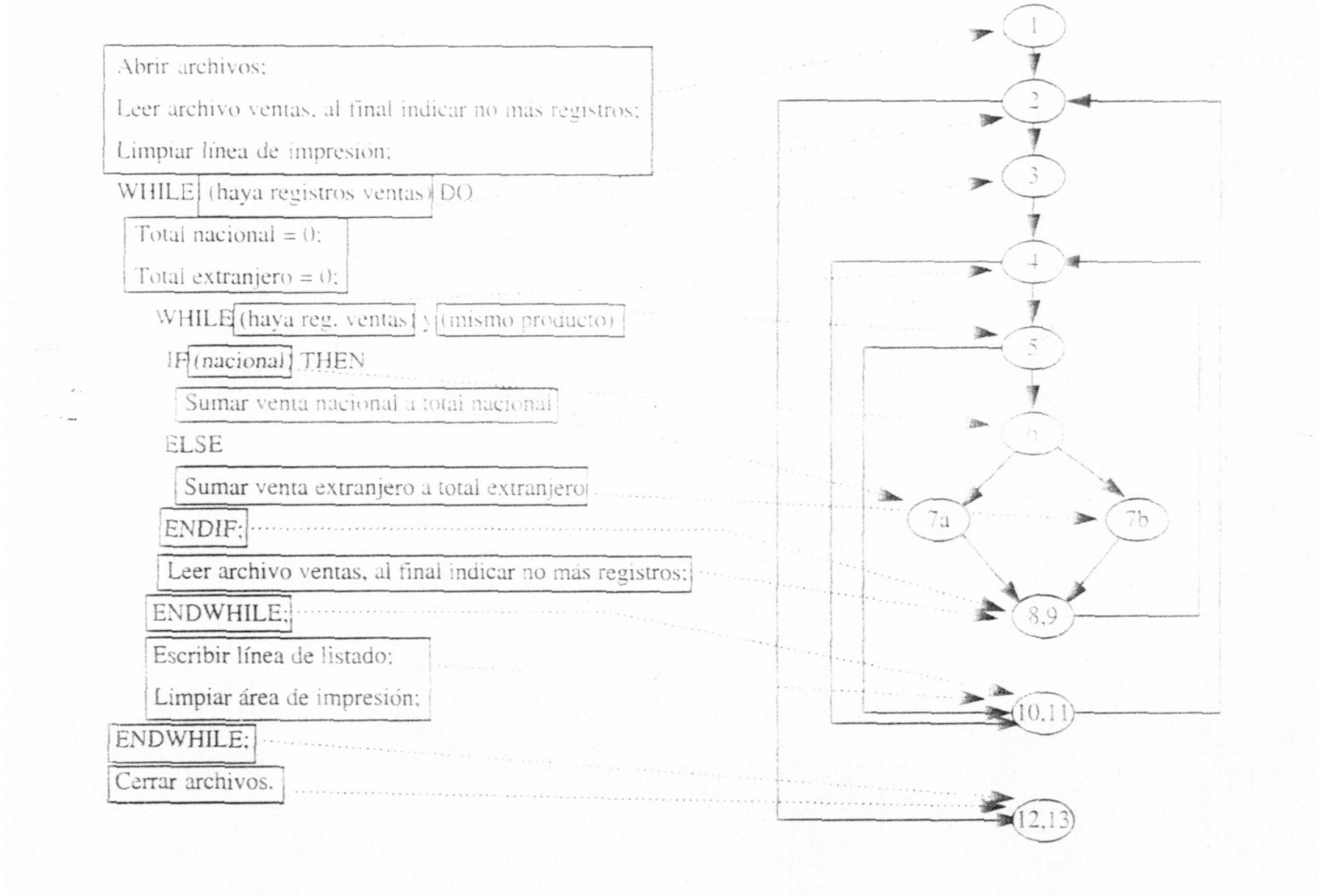
http://www-306.ibm.com/software/info/ecatalog/es_ES/rational/SW700.html?&S_TACT=none&S_CMP=none

Tema 5. Prueba de Software.

Notación de Grafo.



Secuencia ***Si x entonces...*** ***Hacer ... hasta (x)*** ***Mientras (x) hacer...***


Abrir archivos;
Leer archivo ventas, al final indicar no más registros;
Limpiar línea de impresión;
WHILE (haya registros ventas) DO
Total nacional = 0;
Total extranjero = 0;
WHILE (haya reg. ventas) y (mismo producto)
IF (nacional) THEN
Sumar venta nacional a total nacional
ELSE
Sumar venta extranjero a total extranjero
ENDIF;
Leer archivo ventas, al final indicar no más registros;
ENDWHILE;
Escribir línea de listado;
Limpiar área de impresión;
ENDWHILE;
Cerrar archivos.
1
2
3
4
5
6
7a
7b
8,9
10,11
12,13

Tema 5. Prueba de Software. Prueba del camino básico. Definiciones

- *Camino*: secuencia de sentencias encadenadas desde la sentencia inicial del programa hasta su sentencia final.
- *Camino de prueba*: un camino que atraviesa, como máximo, una vez el interior de cada bucle que encuentra.

  Algunos autores hablan de pasar 3 veces:

  una sin entrar en su interior

  otra entrando una vez

  otra entrando dos veces
- *Camino linealmente independiente* de otros: introduce por lo menos un nuevo conjunto de sentencias de proceso o una nueva condición.

  Þ *en términos del grafo de flujo, introduce una nueva arista que no haya sido recorrida anteriormente a la definición del camino*

Tema 5. Prueba de Software. Prueba del camino básico. Criterio de Prueba

- Buen criterio de prueba: *Ejecución de un conjunto de caminos independientes*.
- *¿Número **máximo** de caminos independientes?*

  $\Rightarrow$ *Complejidad ciclomática* de Mc Cabe, V(G):

  V(G) = a - n + 2 (“a”, nº de arcos del grafo; “n”, nº de nodos)

  V(G) = r (“r”, nº de regiones del grafo)

  V(G) = c +1 (“c”, nº de nodos de condición)

  (*Case of 1 ... N* cuenta como n-1 en V(G)= c+1)

Tema 5. Prueba de Software. Prueba del camino básico. Criterio de Prueba

1. Usando el diseño o el código como base, dibujamos el correspondiente grafo de flujo.
2. Determinamos la complejidad ciclomática del grafo de flujo resultante, V(G).
3. Determinamos un conjunto básico de **hasta** V(G) caminos linealmente independientes.
4. Preparamos los casos de prueba que forzarán la ejecución de cada camino del conjunto básico.

Tema 5. Prueba de Software. Prueba del camino básico. Derivación de Casos

1. Usando el diseño o el código como base, dibujamos el correspondiente grafo de flujo.
2. Determinamos la complejidad ciclomática del grafo de flujo resultante, V(G).
3. Determinamos un conjunto básico de **hasta** V(G) caminos linealmente independientes.
4. Preparamos los casos de prueba que forzarán la ejecución de cada camino del conjunto básico.

Tema 5. Prueba de Software. Prueba del camino básico. Derivación de Casos

Algunos consejos:

- Numerar los nodos del grafo secuencialmente.
- Describir el conjunto de caminos independientes (subrayar aristas que los hacen independientes de los anteriores).

  1-2-11

  1-2-3-4-...

  ...
- Algunos caminos no se pueden ejecutar solos y requieren la concatenación con otro.
- Algunos caminos pueden ser imposibles $\Rightarrow$ seleccionar otros
- A partir de los caminos, analizar el código para ver qué datos de entrada son necesarios, y qué salida se espera.

Tema 5. Prueba de Software
Complejidad ciclomática. Conclusiones

Según Beizer 90:

- V(G) marca un límite **mínimo** del nº de casos de prueba, contando cada condición de decisión como un nodo individual.
- Parece que cuando V(G) es mayor que 10 la probabilidad de defectos en el módulo crece bastante si dicho valor alto no se debe a sentencias CASE.

  (en ese caso, es recomendable replantearse el diseño modular).

Tema 5. Prueba de Software.

Prueba de bucles *(Beizer 90)*

- Bucles simples (“n” es el nº máx. de iteraciones):
  - 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.
- Bucles no estructurados:
  - Rediseñar primero.
- Bucles anidados (para evitar progresión geométrica):
  - Llevar a cabo las pruebas de bucles simples con el bucle más interior. Configurar el resto de bucles con sus valores mínimos.
  - Progresar hacia afuera, llevando a cabo pruebas para el siguiente bucle, manteniendo los bucles externos en sus valores mínimos y los internos en sus valores “típicos”.
- Bucles concatenados:
  - Si no son independientes, como con los bucles anidados.

Tema 5. Prueba de Software.

Prueba de Caja Negra.

- Estudia la especificación del software, las funciones que debe realizar, las entradas y las salidas – *La interface*.
- Busca tipos de errores diferentes a las pruebas de caja blanca:
  - *¿es el sistema particularmente sensible a ciertos datos de entrada?*
  - *¿qué volumen de datos tolerará el sistema?*
  - *¿qué efectos tendrán determinadas combinaciones de datos sobre el funcionamiento del sistema?*
- Un caso de prueba está bien elegido si:
  - Reduce el nº de casos de prueba adicionales para alcanzar una prueba razonable.
  - Nos dice algo sobre la presencia o ausencia de *clases de errores*.

Tema 5. Prueba de Software.

Particiones o clases de equivalencia

(Piattini et al. 96)

- Cada caso de prueba debe cubrir el máximo nº de entradas.
- Debe tratarse el dominio de valores de entrada dividido en un nº finito de clases de equivalencia.

  ⇒ la prueba de un valor representativo de la clase permite suponer "razonablemente" que el resultado obtenido será el mismo que probando cualquier otro valor de la clase
- Método de diseño:

  1. Identificación de clases de equivalencia.
  2. Creación de los casos de prueba correspondientes.

Tema 5. Prueba de Software.

Paso 1. Particiones o clases de equivalencia

(Piattini et al. 96)

1.1. Identificar las condiciones de las entradas del programa.

1.2. A partir de ellas, se identifican las clases de equivalencia:

De datos válidos

De datos no válidos

1.3. Algunas reglas para identificar clases:

- Por cada rango de valores, se especifica una clase válida y dos no válidas.
- Si se especifica un nº de valores, se creará una clase válida y dos no válidas.
- Si se especifica una situación del tipo “debe ser” o booleana, se identifica una clase válida y una no válida.
- Si se especifica un conjunto de valores admitidos, **y el programa trata de forma distinta cada uno de ellos**, se crea una clase válida por cada valor, y una no válida.
- **Si se sospecha que ciertos elementos de una clase no se tratan igual que el resto de la misma, debe dividirse en clases menores.**

Tema 5. Prueba de Software.

Paso 1. Particiones o clases de equivalencia

(Piattini et al. 96)

1.3. Algunas reglas para identificar clases:

- Por cada rango de valores, se especifica una clase válida y dos no válidas
  - (válida) 1 ≤ número ≤ 49; (no válidas) número < 1, número > 49
- Si se especifica un nº de valores, se creará una clase válida y dos no válidas
  - (válida) num propietarios = 3;
    (no válidas) num propietarios < 3, num propietarios > 3
- Si se especifica una situación del tipo "debe ser" o booleana, se identifica una clase válida y una no válida
  - (válida) "El primer carácter es una letra"; (no válida) "(...) no es una letra"
  - (válida) "X es un número"; (no válida) "X no es un número"
- Si se especifica un conjunto de valores admitidos, y el programa trata de forma distinta cada uno de ellos, se crea una clase válida por cada valor, y una no válida
  - Tres tipos de inmuebles: (válidas) "pisos", "chalets", "locales comerciales";
  - (no válida) "jkllñ"

Tema 5. Prueba de Software.

Paso 2. Particiones o clases de equivalencia

(Piattini et al. 96)

2.1. Asignación de un número único a cada clase de equivalencia.

2.2. Hasta que todas las clases de equivalencia hayan sido cubiertas por casos de prueba, se tratará de escribir un caso que cubra tantas clases válidas no incorporadas como sea posible.

2.3. Hasta que todas las clases de equivalencia no válidas hayan sido cubiertas por casos de prueba, escribir un caso para una ÚNICA clase no válida sin cubrir.

Tema 5. Prueba de Software.

Análisis de los Valores Límite.

- Los casos de prueba que exploran las condiciones límite producen mejores resultados.
  - *"Los defectos del software se acumulan en las situaciones límite"*
- Diferencias de AVL con particiones de equivalencia:
  - No se elige "cualquier elemento" de la clase de equivalencia, sino uno o más de manera que los márgenes se sometan a prueba.
  - *Los casos de prueba se generan considerando también el espacio de salida – Postcondiciones.*

Análisis de los Valores Límite.
Reglas para Identificar Casos

- Si una condición de entrada especifica un rango delimitado por los valores a y b, se deben generar casos para a y b y casos no válidos justo por debajo y justo por encima de a y b, respectivamente.
- Si una condición de entrada especifica un número de valores, se deben desarrollar casos de prueba que ejerciten los valores máximo y mínimo, uno más el máximo y uno menos el mínimo.
- Aplicar las directrices 1 y 2 a las condiciones de salida.
- Si las estructuras de datos internas tienen límites preestablecidos, hay que asegurarse de diseñar un caso de prueba que ejercite la estructura de datos en sus límites.

Tema 5. Prueba de Software.

Análisis de los Valores Límite. Reglas para Identificar Casos

- Si una condición de entrada especifica un rango delimitado por los valores a y b, se deben generar casos para a y b y casos no válidos justo por debajo y justo por encima de a y b, respectivamente
  - Rango de entrada: [-1.0, 1.0]
    - Casos de prueba para –1.0, +1.0, -1.001, +1.001 (si se admiten 3 decimales)
- Si una condición de entrada especifica un número de valores, se deben desarrollar casos de prueba que ejerciten los valores máximo y mínimo, uno más el máximo y uno menos el mínimo
  - "El fichero de entrada tendrá de 1 a 255 registros"
    - Casos para 0, 1, 254, 255 registros
- Aplicar las directrices 1 y 2 a las condiciones de salida
  - "El programa podrá mostrar de 1 a 4 listados"
    - Casos para intentar generar 0, 1, 4 y 5 listados

Tema 5. Prueba de Software
Prueba de Interfaces Gráficas de Usuario
(GUI, Graphical User Interface)

Uso de una lista de chequeo preestablecida (ver (Pressman 98), p.319):

- Para ventanas:
  - ¿Se abrirán las ventanas mediante órdenes basadas en el teclado o en un menú?
  - ¿Se puede ajustar el tamaño, mover y desplegar la ventana?
  - ¿Se regenera adecuadamente cuando se escribe y se vuelve a abrir?
  - ...
- Para menús emergentes y operaciones con el ratón:
  - ¿Se muestra la barra de menú apropiada en el contexto apropiado?
  - ¿Es correcto el tipo, tamaño y formato del texto?
  - ¿Si el ratón tiene varios botones, están apropiadamente reconocidos en el contexto?
  - ...
- Entrada de datos:
  - ¿Se repiten y son introducidos adecuadamente los datos alfanuméricos?
  - ¿Funcionan adecuadamente los modos gráficos de entrada de datos? (p.e. barra deslizante)
  - ¿Se reconocen adecuadamente los datos no válidos?
  - ¿Son inteligibles los mensajes de entrada de datos?

Tema 5. Prueba de Software
Estrategias de prueba del software.
Niveles de prueba – *DSI 10 pag 53*

Sistema de Información:

- Pruebas unitarias.
- Pruebas de integración.
- Pruebas del sistema.
- Pruebas de implantación.
- Pruebas de aceptación.

Tema 5. Prueba de Software
Estrategias de prueba del software.
Niveles de prueba – *DSI 10 pag 53*

- Las ***pruebas unitarias*** comprenden las verificaciones asociadas a cada componente del sistema de información. Su realización tiene como objetivo verificar la funcionalidad y estructura de cada componente individual.

- Las ***pruebas de integración*** comprenden verificaciones asociadas a grupos de componentes, generalmente reflejados en la definición de subsistemas de construcción o en el plan de integración del sistema de información. Tienen por objetivo verificar el correcto ensamblaje entre los distintos componentes.

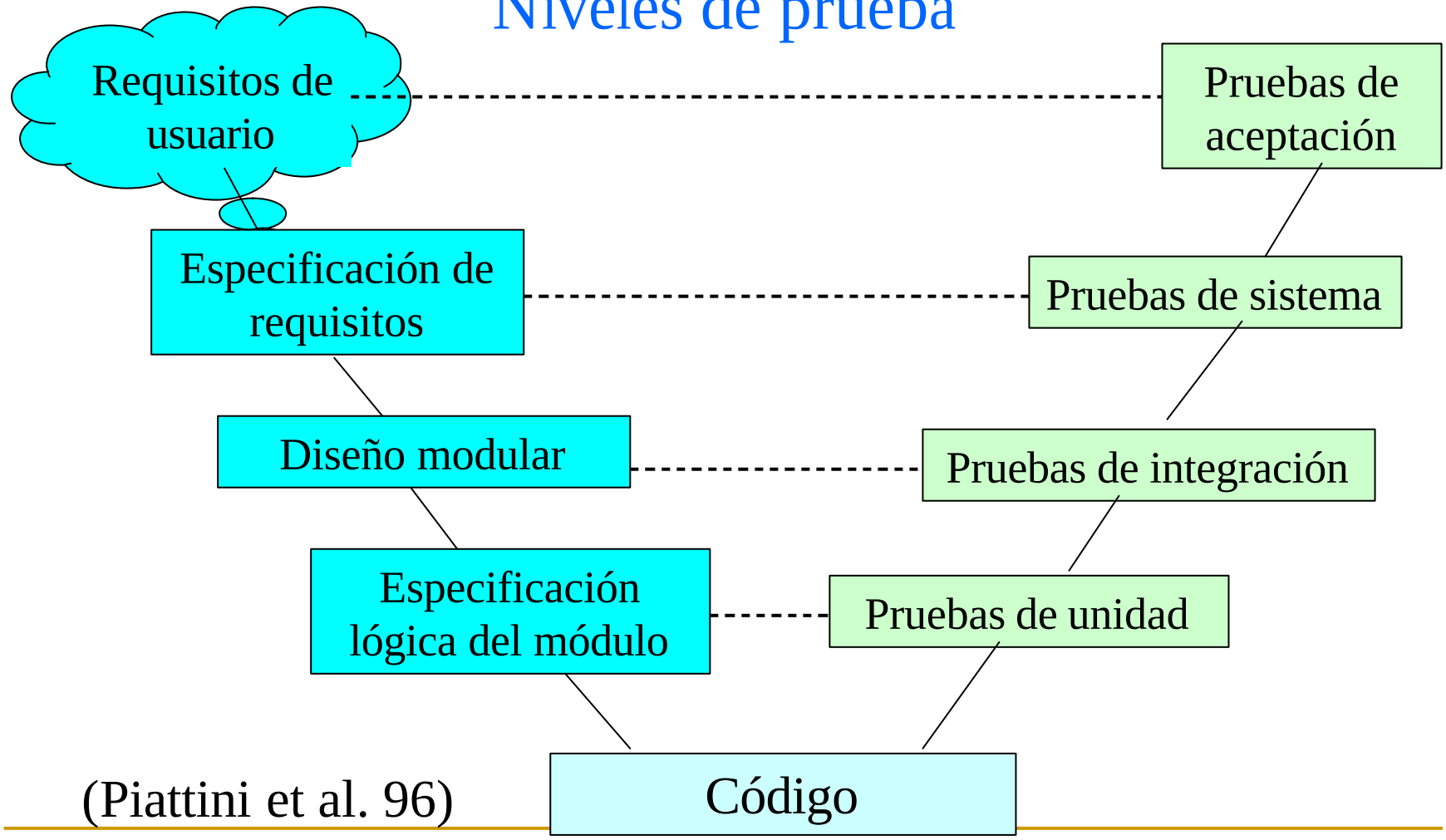
Tema 5. Prueba de Software
Estrategias de prueba del software.
Niveles de prueba – *DSI 10 pag 53*

- Las pruebas del ***sistema, de implantación y de aceptación*** corresponden a verificaciones asociadas al sistema de información, y reflejan distintos propósitos en cada tipo de prueba:
  - Las pruebas del sistema son pruebas de integración del sistema de información completo. Permiten probar el sistema en su conjunto y con otros sistemas con los que se relaciona para verificar que las especificaciones funcionales y técnicas se cumplen.
  - Las pruebas de implantación incluyen las verificaciones necesarias para asegurar que el sistema funcionará correctamente en el entorno de operación al responder satisfactoriamente a los requisitos de rendimiento, seguridad y operación, y coexistenciacon el resto de los sistemas de la instalación, y conseguir la aceptación del sistema porparte del usuario de operación.
  - Las pruebas de aceptación van dirigidas a validar que el sistema cumple los requisitos de funcionamiento esperado, recogidos en el catálogo de requisitos y en los criterios de aceptación del sistema de información, y conseguir la aceptación final del sistema porparte del usuario.

Tema 5. Prueba de Software
Estrategias de prueba del software.
Niveles de prueba – *DSI 10 pag 54*

- Las pruebas unitarias, de integración y del sistema se llevan a cabo en el proceso Construcción del Sistema de Información (CSI), mientras que las pruebas de implantación y aceptación se realizan en el proceso Implantación y Aceptación del Sistema (IAS).

Tema 5. Prueba de Software
Estrategias de prueba del software.
Niveles de prueba



(Piattini et al. 96)