ADSI – TEMA 7

Evaluación / Pruebas del Software

Vitales en cualquier Desarrollo

- https://www.youtube.com/watch?v=TIMY9j5jh-g&list=PLF8D60F9C7DE7624A&index=1

Índice

- Pruebas del Software
- Pruebas de Unidad
  - Caja Blanca
  - Caja Negra
- Pruebas de Integración
- Pruebas de Aceptación
- Pruebas de Regresión
- Otros tipos de pruebas
- Documentación de pruebas
- Conclusiones

Pruebas del Software

- Las pruebas son parte del proceso de desarrollo de un sistema
- Probar un software es ejecutarlo “con mala idea” para que falle
- El objetivo de las pruebas es **encontrar errores**, no ver lo bien que funciona nuestro software

Pruebas del Software

- Tipos de Pruebas
  - **De unidad**
    - **Caja blanca**
    - **Caja negra**
  - **Integración**
  - **Aceptación**
  - **Regresión**

Pruebas de Unidad

- Consisten en ir probando los distintos módulos uno a uno
- Las pruebas de caja blanca se centran en el código del módulo
- Las pruebas de caja negra prescinden del detalle del código

Pruebas de Unidad. Caja blanca

- Se trata de ir probando las distintas posibilidades de ejecución del código
- Hay que asegurarse que todo el código es necesario y alcanzable (cobertura de segmentos)

Pruebas de Unidad. Caja blanca

- Si en el código hay una instrucción condicional, habrá que hacer una prueba que satisfaga la condición y otra que no
- Si la condición es compuesta habrá que probar las distintas combinaciones

Pruebas de Unidad. Caja blanca

- Si lo que tenemos es un bucle habrá que probar qué ocurre si no se ejecuta nunca, si sólo se ejecuta una vez o si se ejecuta n veces
- Hay que tener especial cuidado con los bucles con instrucciones tipo "*break*"

Pruebas de Unidad. Caja blanca

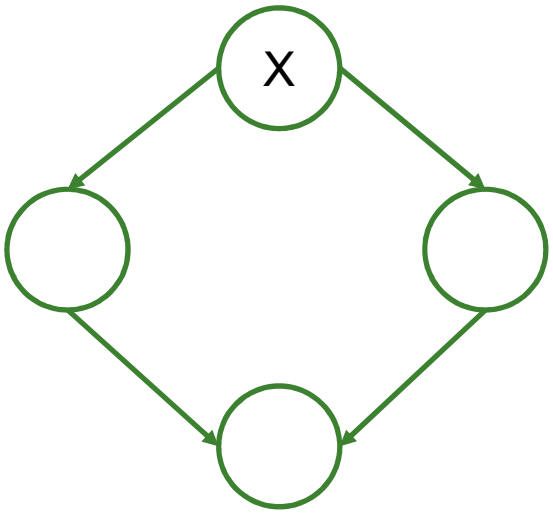
- Complejidad Ciclomática (La complejidad de McCabe V (G))
  - La métrica de McCabe ha sido muy popular en el diseño de pruebas.
  - Es un indicador del número de caminos independientes que existen en un grafo

Pruebas de Unidad. Caja blanca

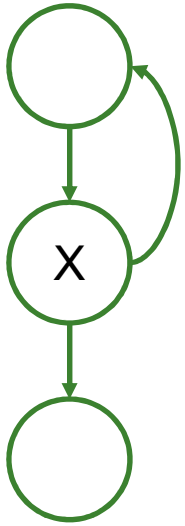
- V (G) marca el límite mínimo de casos de prueba para un programa.
- Cuando V (G) >10 quizás sea interesante dividir el módulo.

Pruebas de Unidad. Caja blanca

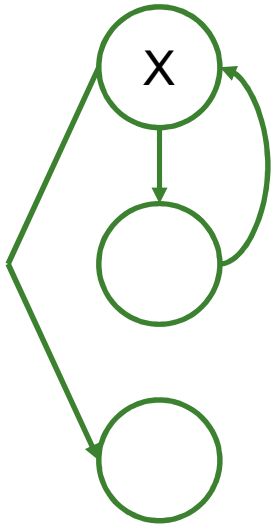
Secuencia



Si x Entonces…
(If x Then … Else…)



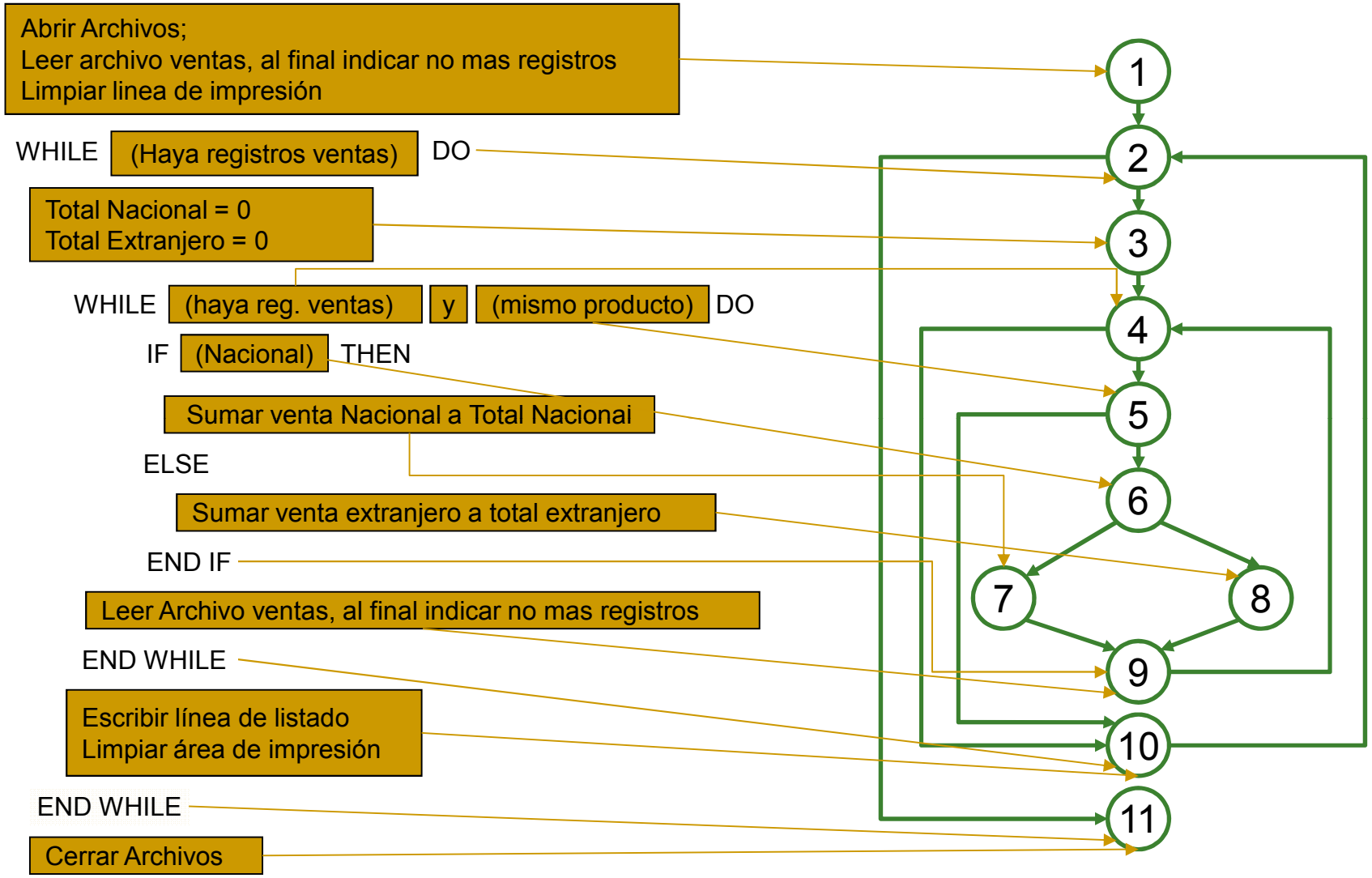
Hacer … hasta x
(Do … Until x)
Repetir



Mientras x Hacer …
(While x Do …)

- Separar todas las condiciones
- Agrupar sentencias ‘simples’ en bloques
- Numerar todos los bloques y las condiciones

Pruebas de Unidad. Caja blanca

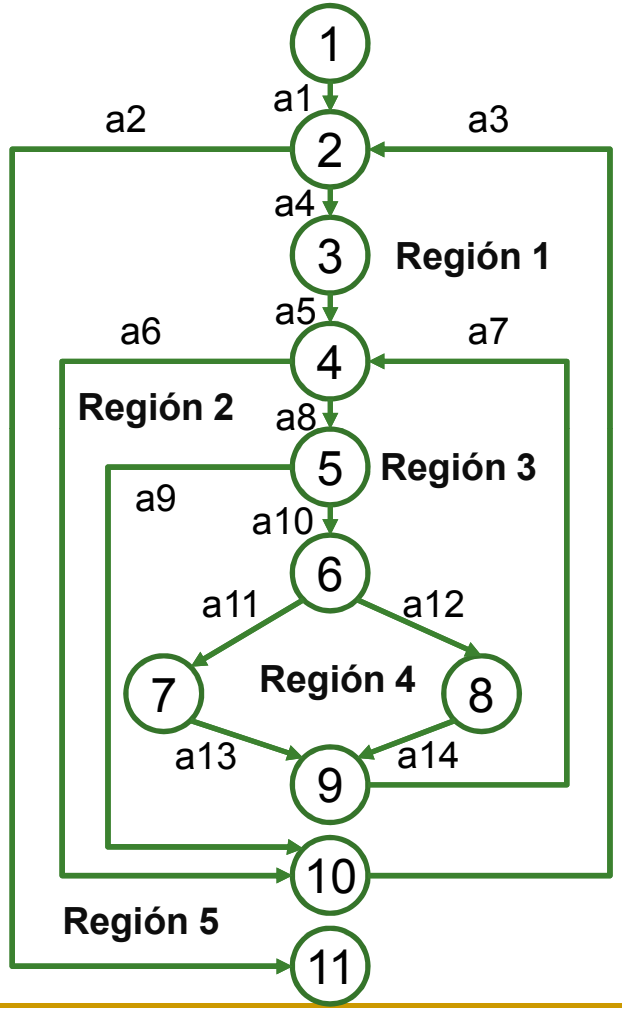


Pruebas de Unidad. Caja blanca

- 3 formas de calcular V (G)
  - V (G) = a – n + 2
  - V (G) = r
  - V (G) = c + e

  Donde
  - a : # de arcos o aristas del grafo.
  - n : # de nodos.
  - r : # de regiones cerradas del grafo.
  - c : # de nodos de condición.
  - e: # de returns o puntos de finalización.

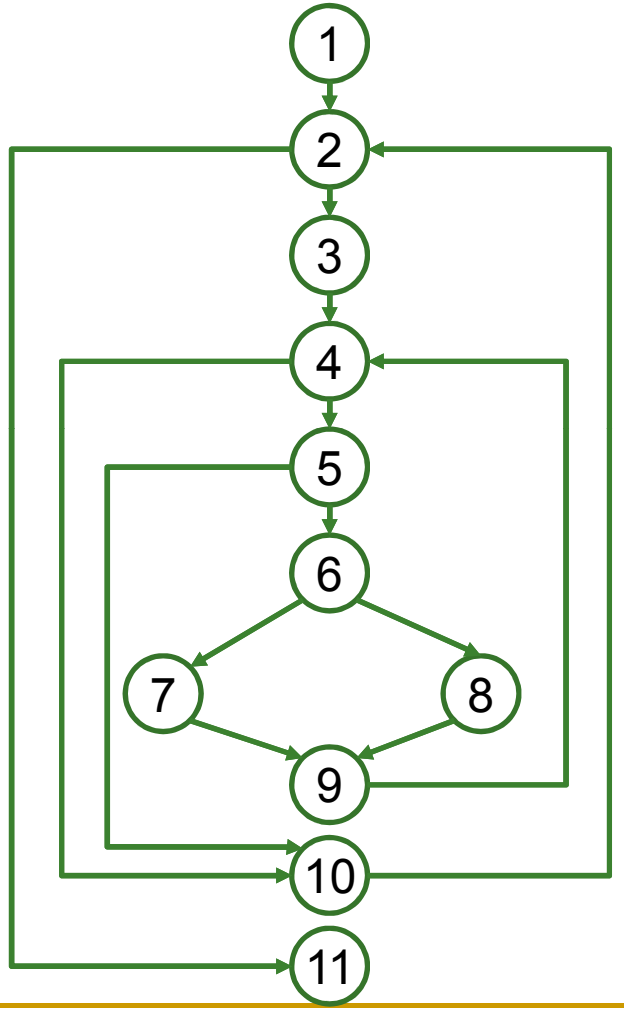
Pruebas de Unidad. Caja blanca



- a) V (G) =14-11+2=5
- b) V (G) = 5 Regiones Cerradas
- c) V (G) = 4+1= 5

Pruebas de Unidad. Caja blanca

- SI V(G) =5 hay 5 caminos independientes en el grafo
  - *Camino 1: 1-2-11*
  - *Camino 2: 1-2-3-4-10-2….*
  - *Camino 3: 1-2-3-4-5-10…*
  - *Camino 4: 1-2-3-4-5-6-7-8-9-4…*
  - *Camino 5: 1-2-3-4-5-6-8-…*



Pruebas de Unidad. Caja blanca

- Tipos de cobertura y criterios
  - Cobertura de Sentencias.
    - Ejecutar casos de prueba que aseguren que cada sentencia se ejecute al menos 1 vez.
    - En el ejemplo bastaría con ejecutar los caminos 4 y 5
  - Cobertura de Decisiones.
    - Ejecutar casos de prueba que aseguren que cada decisión toma al menos una vez el valor verdadero y otra el valor falso.

Pruebas de Unidad. Caja blanca

- Tipos de cobertura y criterios
  - Cobertura de Condiciones.
    - Ejecutar casos de prueba que aseguren que cada condición va a tomar al menos una vez el valor verdadero y otra el falso.

Pruebas de Unidad. Caja blanca

- Tipos de cobertura y criterios
  - Criterios de decisión/Condición.
    - Consiste en exigir que se cumplan la cobertura de condiciones y la de decisiones
  - Criterio de Condición Múltiple.
    - Consiste en subdividir las condiciones múltiples en condiciones sìmples y aplicar la cobertura de ese modo
  - Criterio de Cobertura de Caminos
    - Recorrer todos los caminos. Impracticable

Pruebas de Unidad. Caja blanca

- En la práctica es imposible cubrir el 100% de los segmentos del código
- Lo ideal, en sistemas no críticos, es cubrir un 75% de los segmentos (aprox.)

Pruebas de Unidad. Caja blanca

- Una buena cobertura con pruebas de caja blanca es deseable, no suficiente
- Las pruebas de caja blanca nos dicen que el código hace bien lo que hace; no que eso sea lo que tiene que hacer

Pruebas de Unidad. Caja negra

- Las pruebas de caja negra son pruebas funcionales, se centran en lo que se espera del módulo
- Se limitan a dar unos datos de entrada y observar la salida ignorando el interior

Pruebas de Unidad. Caja negra

- Las pruebas de caja negra se basan en la especificación de los requisitos, qué debe hacer el sistema
- Estas pruebas deben tener una cobertura cercana al 100%
- A la hora de introducir datos para las pruebas, hay que hacerlo “con mala idea”

Pruebas de Unidad. Caja negra

- A la hora de introducir datos, hablaremos de clases equivalentes:
  - Partimos el rango de los posibles datos en distintas clases, y basta con probar con un dato de cada clase; los demás son equivalentes
  - Recomendable añadir siempre los valores frontera
  - Ej: El dato a introducir es un número >=0
    - Clases equivalentes
      - -Números negativos
      - -Números positivos
      - -El cero
      - -Cualquier carácter que no sea un número

Pruebas de Unidad. Caja negra

- Lograr una buena cobertura de pruebas de caja negra es deseable, pero no suficiente
- Pueden existir defectos internos que no surjan en las pruebas de caja negra (Ej. Fechas)
- Una prueba de caja negra asegura que el módulo hace lo que queremos, no que no haga nada más que eso

Pruebas de Integración

- Las pruebas de integración involucran a un número creciente de módulos hasta llegar a probar el sistema completo
- Hay que probar las conexiones entre los módulos (semántica y funcionalidad)
- Las pruebas finales de integración cubren todo el sistema

Pruebas de Integración

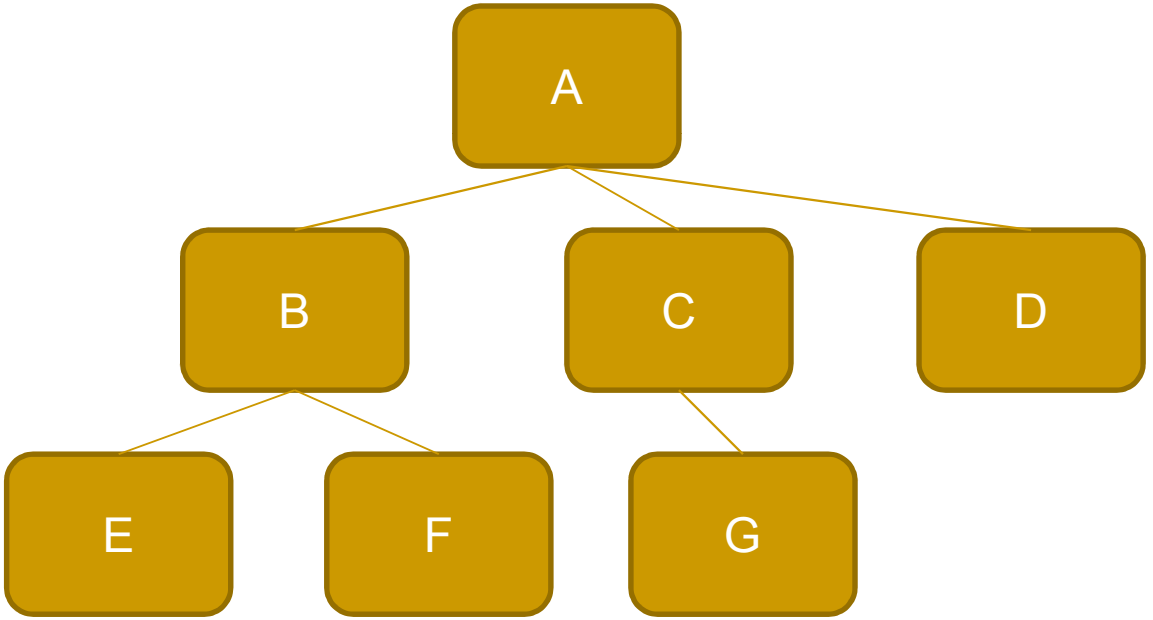
- El orden de integración elegido afecta a diversos factores:
  - La forma de preparar casos
  - Las herramientas necesarias
  - El orden de codificar y probar los módulos
  - El coste de la depuración
  - El coste de preparación de casos

Pruebas de Integración

- Tipos de Integración
  - Integración incremental. Se combina el siguiente módulo que se debe probar con el conjunto de módulos que ya han sido probados
    - Integración ascendente. Se comienza por los módulos hoja.
    - Integración descendente. Se comienza por el módulo raíz.
  - Integración no incremental. Se prueba cada módulo por separado y luego se integran todos de una vez y se prueba el programa completo

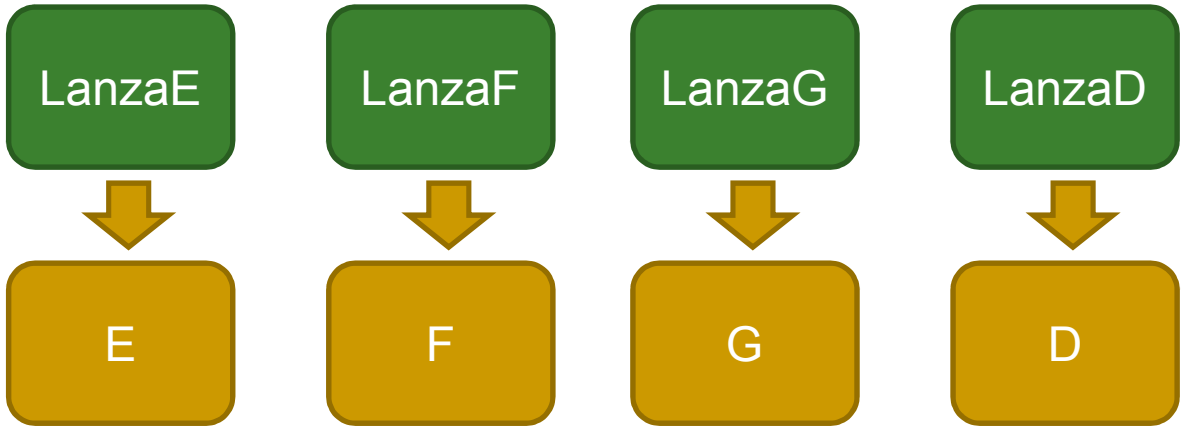
Pruebas de Integración

- Ejemplo prueba de integración
  - Supongamos la siguiente estructura de módulos



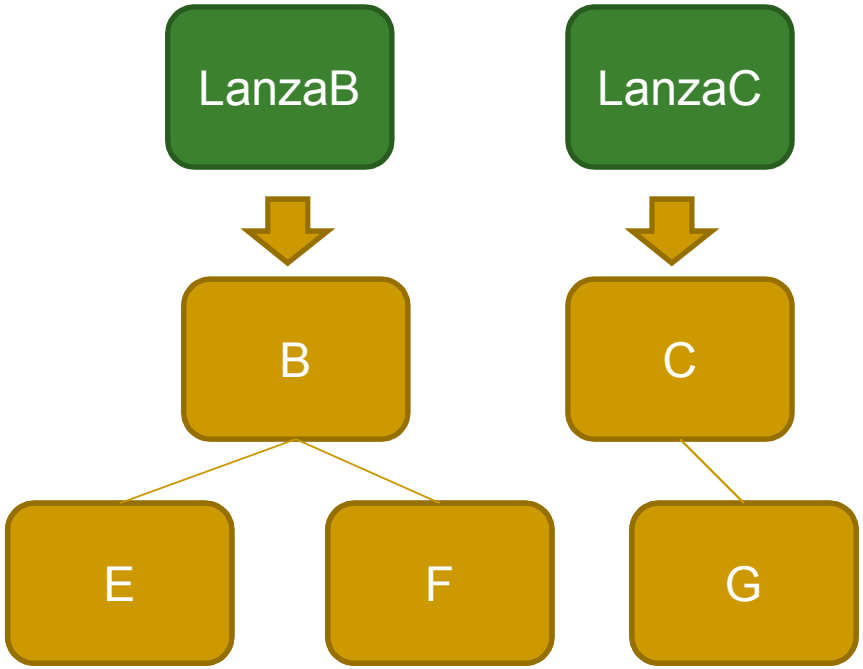
Pruebas de Integración

- Integración incremental ascendente
  - Se necesitan módulos “lanzadores”
    - Serán los encargados de ejecutar los módulos que se desean probar
    - Permite automatizar las pruebas



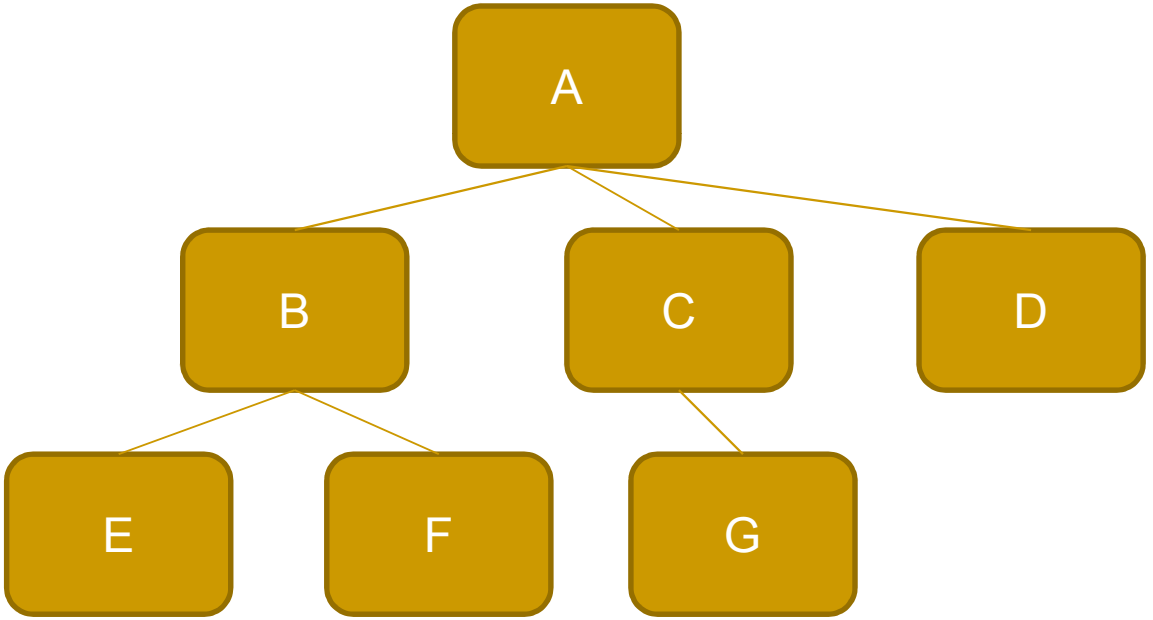
Pruebas de Integración

- Integración incremental ascendente (cont.)



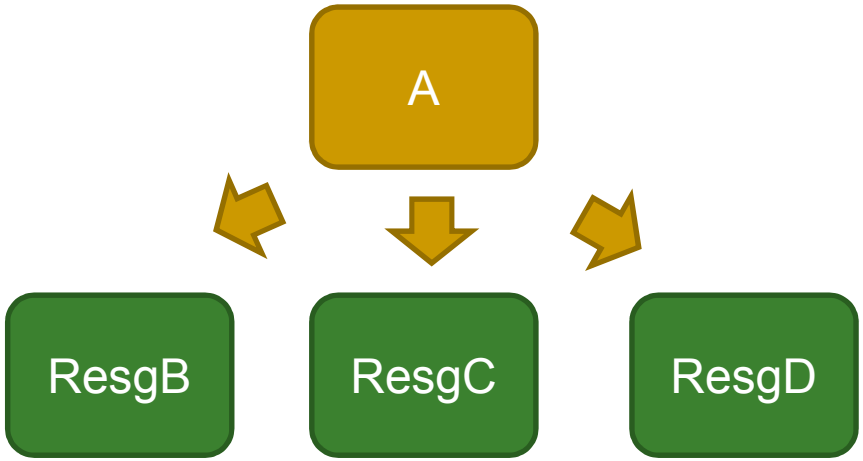
Pruebas de Integración

- Integración incremental ascendente (cont.)



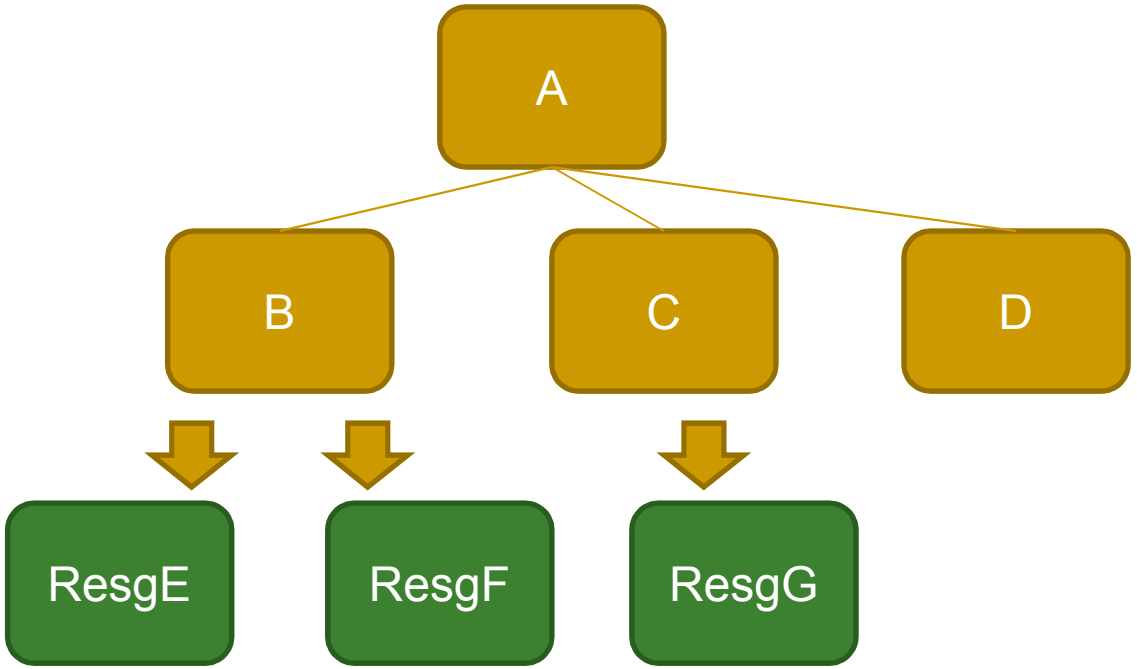
Pruebas de Integración

- Integración incremental descendente
  - Se necesitan módulos “resguardo”
    - Serán los encargados de simular el comportamiento de los módulos que todavía no se tienen disponibles
    - Permiten lanzarlos de manera repetida



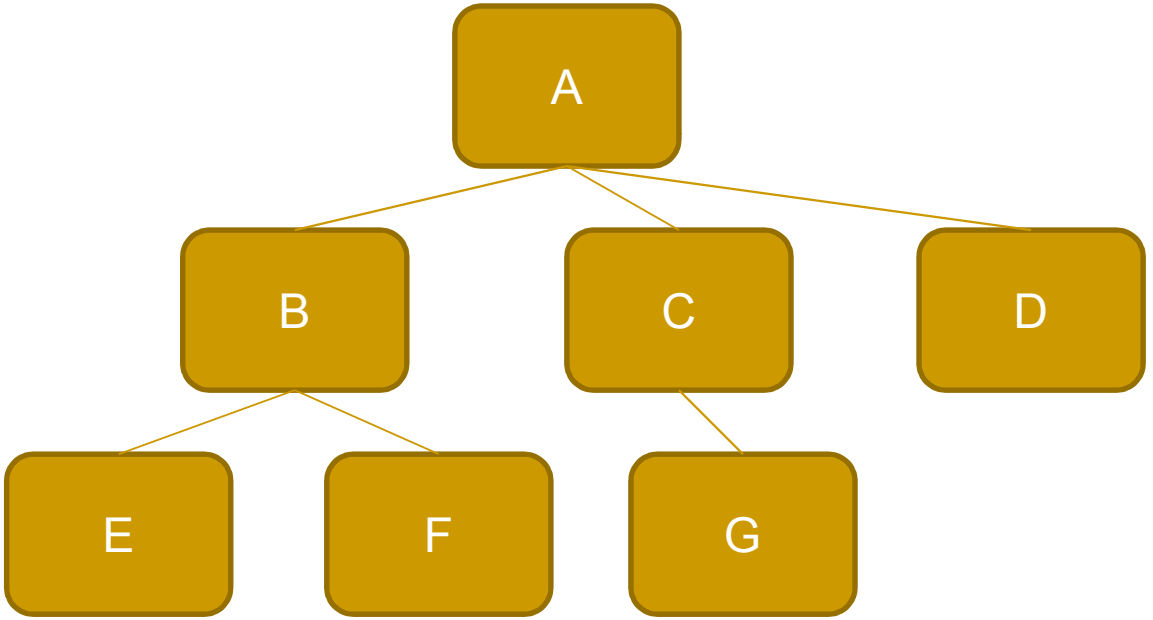
Pruebas de Integración

- Integración incremental descendente (cont.)



Pruebas de Integración

- Integración incremental descendente (cont.)



Pruebas de Aceptación

- Son pruebas funcionales realizadas por el usuario final del sistema
- El cliente siempre tiene la razón
- Dos tipos de pruebas de aceptación
  - Pruebas alfa: con el desarrollador
  - Pruebas beta: el cliente solo

Pruebas de Regresión

- Verifican que el software sigue funcionando como lo hacía tras introducir cambios/mejoras.
  - Es decir, que “no hemos roto nada”.
- Diferentes técnicas
  - Ejecutar todos los casos de prueba
  - Seleccionar los más relevantes
  - Priorizar los más relevantes para que se ejecuten antes
- Se intentan automatizar

Otros tipos de pruebas

- Pruebas de resistencia
- Pruebas de seguridad
- Pruebas de rendimiento
- Pruebas de recuperación

Plan de Pruebas

- Antes de realizar una prueba hay que tener claro:
  - Qué se quiere probar
  - Cómo se va a probar
  - Qué resultado se espera

- Una vez realizada la prueba hay que indicar
  - Qué resultado se ha obtenido y si ha habido que modificar algo

Documentación de Pruebas

- La fase de pruebas supone mucho tiempo en el desarrollo del software
- La fase de pruebas tiene que estar documentada en un Proyecto

Documentación de Pruebas

Código de Prueba	Descripción	Resultado Esperado	Resultado Obtenido	Observaciones
1	Usuario incorrecto y contraseña correcta	Mensaje de error	Mensaje de error	Correcto
2	Usuario correcto y contraseña incorrecta	Mensaje de error	Mensaje de error	Correcto
3	Usuario correcto y contraseña correcta	Acceso al sistema	Mensaje de error	Se busca en la tabla de la BD que no es
3a	Usuario correcto y contraseña correcta	Acceso al sistema	Mensaje de error	Se contempla la diferencia entre mayúsculas y minúsculas en el usuario
3b	Usuario correcto y contraseña correcta	Acceso al sistema	Acceso al Sistema	Correcto
.....				

Definidos PREVIAMENTE
(Plan de Pruebas)

Conclusiones

- Probar es ENCONTRAR FALLOS
- Las pruebas debería diseñarlas alguien distinto al que implementó el módulo
- Las pruebas se van haciendo según se va implementando, no se espera al final
- Las pruebas pueden encontrar fallos; nunca asegurar que éstos no existen