FACULTAD DE INGENIERIA

Fundamentos de Ingeniería de Software

Pruebas de software:

Verificación y validación

Clase 2

María de las Nieves Freira

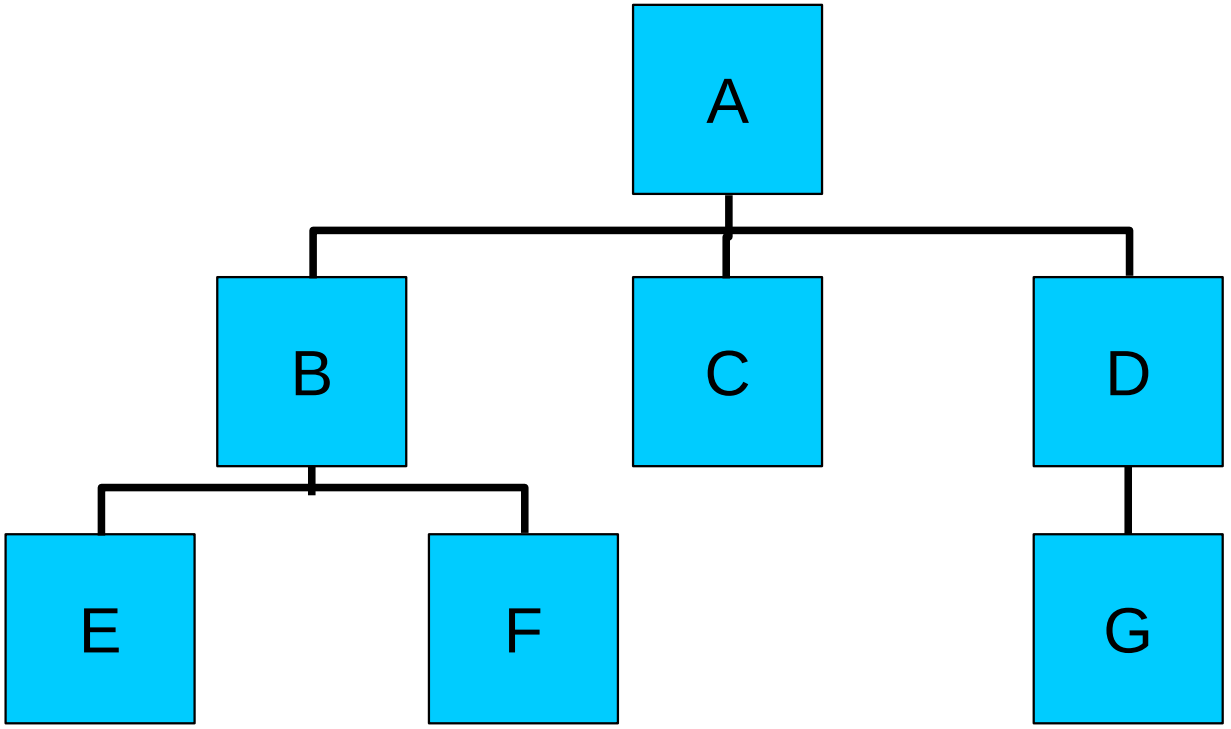
Universidad de la República

Temario

- Pruebas de módulos
- Pruebas del sistema
- Pruebas de desempeño
- Pruebas de aceptación
- Planificación de pruebas

Pruebas de módulos (integración) (I)

- El módulo A «usa» a los módulos B, C, D.
- El módulo B «usa» a los módulos E y F
- El módulo D «usa» al módulo G

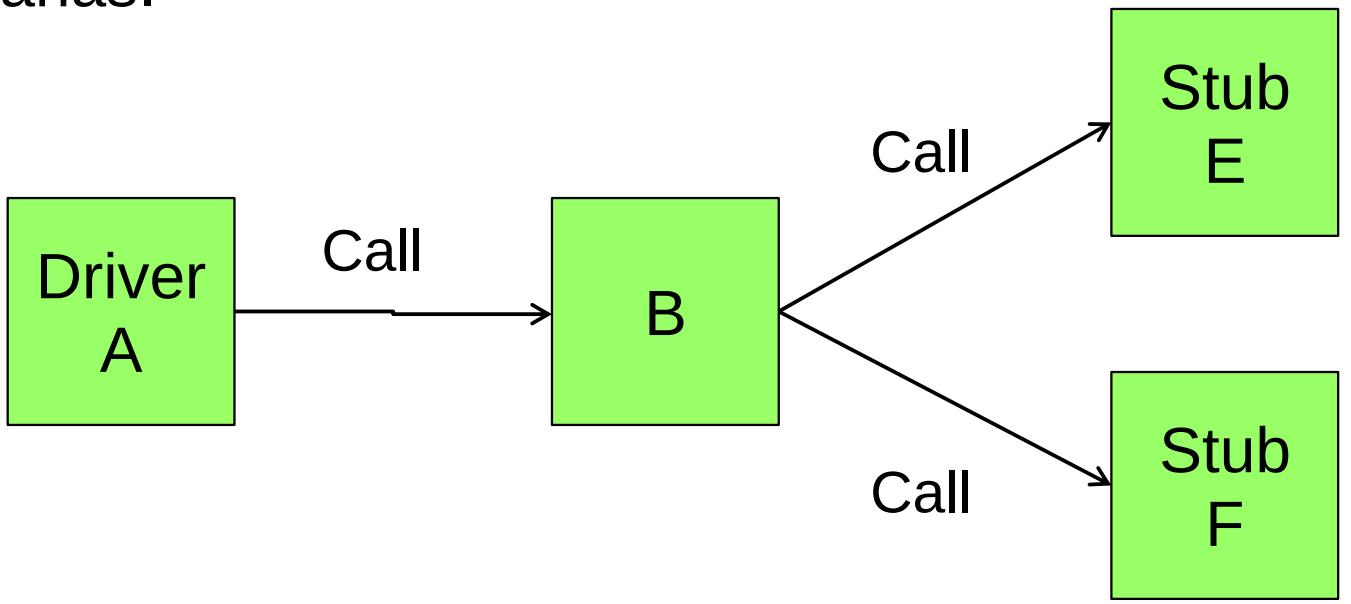


Pruebas de módulos (2)

- Quiero probar al módulo B de forma aislada. No uso los módulos A, E y F
  - **El módulo B es usado por el módulo A**
  - Debo simular la llamada del modulo A al B – *Driver*
    - Pieza de código que simula el uso (por otro módulo) del módulo que está siendo testeado.
  - **El módulo B usa a los módulos E y F**
  - Cuando llamo desde B a E o F debo simular la ejecución de estos módulos – *Stub*
    - Es una pieza de código con los mismos parámetros de entrada y salida que el módulo faltante, pero con una alta simplificación del comportamiento.

Pruebas de módulos (3)

- Quiero probar al módulo B de forma aislada. No uso los módulos A, E y F.
  - Se prueba al módulo B con los métodos vistos en pruebas unitarias.



Estrategias de integración (I)

- El objetivo es lograr combinar módulos o componentes individuales para que trabajen correctamente de forma conjunta.
  - **No incremental**
    - Big-Bang
      - Se prueba cada módulo de forma aislada y luego se prueba la combinación de todos los módulos a la vez.
      - Requiere el uso de *stubs* y *drivers.*

Estrategias de integración (2)

- **Incrementales**
  - *Bottom-up* (ascendente)
    - Comienza por los módulos que no requieren de ningún otro para ejecutar.
    - Se sigue hacia arriba según la jerarquía «usa».
    - Requiere de *drivers,* pero no de *stubs.*
  - *Top-down* (descendente)
    - Comienza por el módulo superior de la jerarquía «usa».
    - Se sigue hacia abajo según la jerarquía «usa».
    - Requiere de *stubs,* pero no de d*rivers.*
  - *Sandwich* (intercalada)
    - Usa *top-down* y *bottom-up*. Busca probar los módulos más críticos primero.
  - Por disponibilidad
    - Se integran los módulos a medida que están disponibles.

Comparación de estrategias

- No incremental vs. incremental
  - Requiere mayor trabajo. Se deben codificar más *stubs* y *drivers* que en las pruebas incrementales.
  - Se detectan más tardíamente los errores de interfaces o suposiciones incorrectas entre los módulos, ya que estos se integran al final.
  - Es más difícil encontrar la falta que provocó la falla. En la prueba incremental es muy factible que el defecto esté asociado con el módulo recientemente integrado, o en interfaces con los otros módulos.
  - Los módulos se prueban menos. En la incremental vuelvo a probar indirectamente a los módulos ya probados.

Pruebas del sistema

- Prueba de función o prueba funcional
  - Verifica que el sistema integrado realiza sus funciones especificadas en los requisitos.
- Prueba de desempeño
  - Verifica los requisitos no funcionales del sistema.
- Prueba de aceptación
  - Se realiza junto con el cliente.
  - Busca asegurar que el sistema es el que el cliente quiere.
- Prueba de instalación
  - Se realizan las pruebas en el ambiente objetivo.

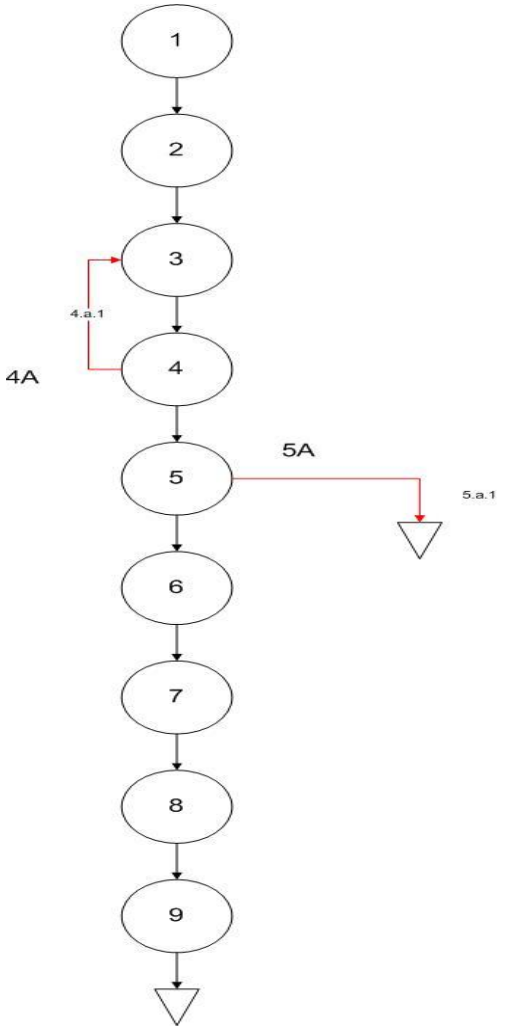
Prueba de función – casos de uso

- Se hace la prueba funcional a partir de los casos de uso.
- Se identifican los distintos escenarios posibles y se usan como base para las condiciones de prueba.
- Se completan las condiciones en función de los tipos de datos de entrada y de salida.
- A partir de las condiciones se crean los casos de prueba.

Ejemplo – Caso de uso: Retiro

Nombre	Retiro
Precondición	El cliente ingresó una tarjeta válida e ingresó nro. de PIN
Flujo principal	1.El CA despliega las distintas alternativas disponibles, el cliente elige Retiro 2.El CA pide cuenta e importe 3.El cliente elige la cuenta de la cual retirar, si es caja de ahorro o cuenta corriente, y el importe 4.El CA envía id. tarjeta, PIN, cuenta e importe al SC (servicio de cajeros) 5.SC (servicio de cajeros) contesta: continuar (OK) o no continuar 6.CA dispensa el dinero 7.CA devuelve la tarjeta 8.CA imprime el recibo 9.Fin CU
Flujos alternativos	**4A. El importe indicado por el cliente no puede obtenerse a partir de los billetes que dispone CA (importe inválido)** 4A1.El CA despliega un mensaje de advertencia y le solicita que ingrese nuevamente el importe. 4A2.Sigue en el punto 3 del FP **5A. No hay suficiente saldo en la cuenta.** 5A1. CA despliega mensaje al cliente: «No hay suficiente saldo en la cuenta» y devuelve la tarjeta (no se imprime recibo) 5A2. Fin CU

CU Retiro – Identificar escenarios (I)



- Ver todos los caminos posibles para recorrer el CU

1. E1: FP
2. E2: FP – 4A – FP (3)
3. E3: FP – 5A
4. E4: FP – 4A – 5A

- ¿Consideramos el caso en que los mismos flujos se repitan más veces? ¿Vale la pena?

- ¿Qué tan crítico es que el sistema indique 1000 veces que el importe es inválido?

CU Retiro – Identificar escenarios (2)

- Instancias del caso de uso:

1. Flujo principal
2. Flujo principal - Importe inválido (¿n?)
3. Flujo principal - No hay suficiente saldo en cuenta
4. Flujo principal - Importe inválido - No hay suficiente saldo en la cuenta

CU Retiro – Condiciones de prueba

Condiciones	Tipo de resultado	Caso
1. Normal	Retiro exitoso	
1.1 Normal caja de ahorros		A1
1.2 Normal cuenta corriente		A2
2. Norma – Importe inválido (n)	Retiro exitoso	A3
3. Normal – Sin saldo	No exitoso	A4
4. Flujo principal – Importe inválido – Sin saldo	No exitoso	A5

CU Retiro – Escenarios y condiciones

ESCENARIO-CONDICIÓN	CA	CC	IMPORTE VÁLIDO	CUENTA CON SALDO	RESULTADO
Normal – CA	V	N/A	V	V	Retiro exitoso
Normal – CC	N/A	V	V	V	Retiro exitoso
Normal - importe inválido	N/A	V	I	N/A	Importe inválido
	N/A	V	V	V	Retiro exitoso
Normal – sin saldo	V	N/A	V	I	No exitoso
Normal - importe inválido - sin saldo	N/A	V	I	N/A	Importe inválido
	N/A	V	V	I	No exitoso

Casos de prueba y datos de prueba (I)

Caso	A1 Flujo normal con CA	A2 Flujo normal con CC	A3 Importe inválido		A4 Sin saldo
Entradas					
Tarjeta	13131	13131	13131		23232
PIN	9001	9001	9001		9002
Cuenta	CA128	CC321	CC321	CC321	CA228
Monto	100	500	12	1200	100
Salidas			Importe inválido		Sin saldo
Dispensa	$100	$500		$1200	No
Dev. tarjeta	Sí	Sí		Sí	Sí
Recibo	13131...	13131...		13131...	No
Condiciones	1.1	1.2		2	3

Casos de prueba y datos de prueba (2)

Caso	A5 Importe inválido – sin saldo				
Entradas					
Tarjeta	13131				
PIN	9001				
Cuenta	CC321	CC321			
Monto	12	100			
Salidas	Importe inválido	Sin saldo			
Dispensa		No			
Dev. tarjeta		Sí			
Recibo		No			
Condiciones	4				

Prueba de desempeño (I)

- Lo esencial es definir
  - Procedimientos de prueba
    - Ejemplo: Tiempo de respuesta
    - Simular la carga, tomar los tiempos de tal manera, cantidad de casos a probar, etc.
  - Criterios de aceptación
    - Ejemplo:
    - El cliente lo valida si en el 90 % de los casos de prueba en el ambiente de producción se cumple con los requisitos de tiempo de respuesta.
  - Características del ambiente
    - Definir las características del ambiente de producción, ya que estas hacen grandes diferencias en los requisitos no funcionales.

Prueba de desempeño (2)

- Prueba de estrés (esfuerzo)
  - Implica someter al sistema a grandes cargas o esfuerzos considerables.
  - No confundir con las pruebas de volumen. Un esfuerzo grande es un pico de volúmenes de datos (normalmente por encima de sus límites) en un *corto período de tiempo.*
  - No solo volúmenes de datos sino también de usuarios, dispositivos, etc.
  - Analogía con dactilógrafo:
    - Volumen: Ver si puede escribir un documento enorme.
    - Esfuerzo: Ver si puede escribir a razón de 50 palabras por minuto.
- Prueba de volumen
  - Esta prueba implica someter al sistema a grandes volúmenes de datos.
  - El propósito es mostrar que el sistema no puede manejar el volumen de datos especificado.

Prueba de desempeño (3)

- Prueba de facilidad de uso
  - Encontrar problemas en la facilidad de uso:
    - ¿Ha sido ajustada cada interfaz de usuario al nivel educacional del usuario final y a las presiones del ambiente?
    - ¿Son las salidas del programa significativas y desprovistas de la «jerga de las computadoras»?
    - ¿Son directos los diagnósticos de error (mensajes de error), o requieren de un doctor en ciencias de la computación?
  - Es importante darse cuenta de lo necesario de este tipo de pruebas. Si no se realizan, el sistema puede ser inutilizable.
  - Algunas se pueden realizar durante el prototipado.

Prueba de desempeño (4)

- Prueba de seguridad
  - La idea es tratar de generar casos de prueba que burlen los controles de seguridad del sistema.
  - Una forma de encontrar casos de prueba es estudiar problemas conocidos de seguridad en sistemas similares.
  - Existen listas de descripciones de fallas de seguridad en diversos tipos sistemas.
- Prueba de compatibilidad
  - Son necesarias cuando el sistema a prueba tiene interfaces con otros sistemas.
  - Se trata de determinar que las funciones con la interfaz no se realizan según especifican los requisitos.

Prueba de desempeño (5)

- Prueba de rendimiento
  - Generar casos de prueba para mostrar que el sistema no cumple con sus especificaciones de rendimiento.
  - Casos:
    - Tiempos de respuesta en sistemas interactivos
    - Tiempo máximo en ejecución de alguna tarea
  - Normalmente esto está especificado bajo ciertas condiciones de carga y de configuración del sistema.
  - Ejemplo:
    - El proceso de facturación no debe durar más de una hora en las siguientes condiciones:
      - Se ejecuta en el ambiente descrito en el anexo 1
      - Se ejecuta para 10.000 empleados

Prueba de desempeño (6)

- Prueba de configuración
  - Se prueba el sistema en distintas configuraciones de hardware y software.
- Prueba de facilidad de instalación
  - Se prueban las distintas formas de instalar el sistema.
- Prueba de recuperación
  - Se intenta probar que no se cumplen los requerimientos de recuperación.
  - Para esto se simulan o crean fallas y luego se testea la recuperación del sistema.

Prueba de desempeño (7)

- Prueba de confiabilidad
  - Se realiza cuando existen requisitos específicos de confiabilidad del sistema.
    - Se deben diseñar casos de prueba especiales.
    - Esto no siempre es fácil.
- Prueba de documentación
  - La documentación del usuario debe ser objeto de inspección controlando su exactitud y claridad (entre otros).
  - Además, todos los ejemplos que aparezcan en ella deben ser codificados como casos de prueba.
    - ¿Qué pasa si un sistema tiene en su manual de usuario un ejemplo que no funciona?

Prueba de aceptación e instalación

- Prueba de aceptación
  - El cliente realiza estas pruebas para ver si el sistema funciona de acuerdo con sus requerimientos y necesidades
    - ¿Qué técnicas/herramientas puede utilizar el cliente para hacer estas pruebas?
    - ¿En qué artefactos generados durante el proceso de desarrollo de software se basará para esto?
- Prueba de instalación
  - Su mayor propósito es encontrar errores de instalación.
  - Es de mayor importancia si la prueba de aceptación no se realizó en el ambiente de instalación.

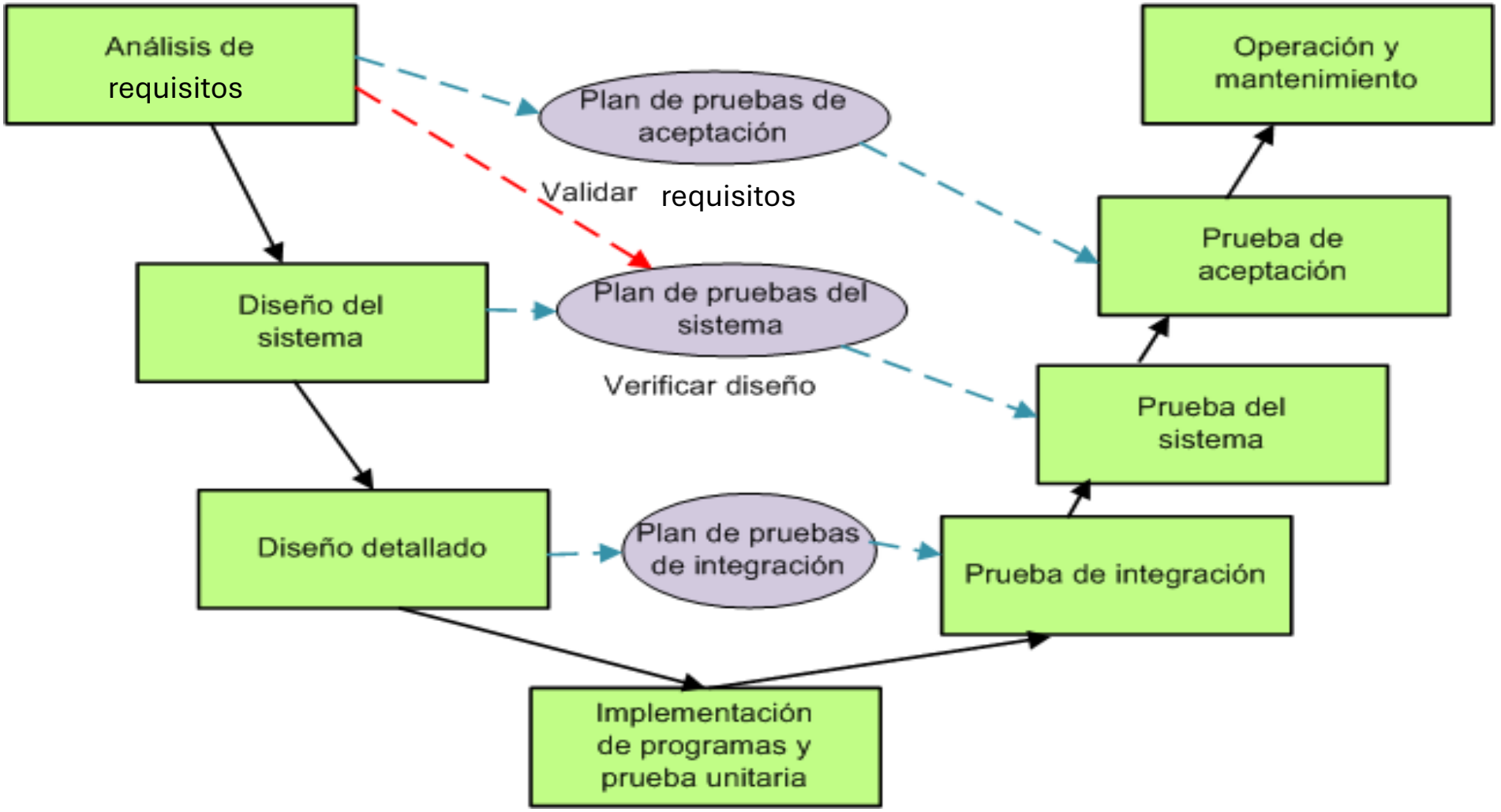
Otras pruebas (I)

- Desarrollo para múltiples clientes
  - Prueba alfa
    - Realizada por el equipo de desarrollo sobre el producto final
  - Prueba beta
    - Realizada por clientes elegidos sobre el producto final
    - Se podría decir que Windows está siempre en *beta-testing*
- Desarrollo de un sistema que sustituye a otro
  - Pruebas en paralelo
    - Se deja funcionando el sistema anterior mientras se empieza a usar al nuevo
    - Se hacen comparaciones de resultados y se intenta ver que el sistema nuevo funciona adecuadamente.

Otras pruebas (2)

- Pruebas de regresión
  - Después de que se realizan modificaciones, se verifica que lo que ya estaba probado sigue funcionando.
  - Conviene tener automatizadas las pruebas; así las pruebas de regresión se pueden hacer con mucha mayor rapidez.
- Pruebas piloto
  - Se pone a funcionar el sistema en producción de forma localizada.
  - Menor impacto de las fallas.

¿Recuerdan...? Modelo V



Planificación de las pruebas (I)

- El mayor error cometido en la planificación de un proceso de prueba:

  Suponer, al momento de realizar el cronograma, que no se encontrarán fallas

- Los resultados de esta equivocación son obvios
  - Se subestima la cantidad de personal a asignar
  - Se subestima el tiempo de calendario a asignar
  - Entregamos el sistema fuera de fecha o ni siquiera entregamos

Planificación de las pruebas (2)

- La V&V es un proceso caro → se requiere llevar una planificación cuidadosa para obtener el máximo provecho de las revisiones y las pruebas para controlar los costos del proceso de V&V.
- El proceso de planificación de V&V:
  - Pone en la balanza los enfoques estático y dinámico para la verificación y la validación.
  - Utiliza estándares y procedimientos para las revisiones y las pruebas de software.
  - Establece listas de verificación para las inspecciones.
  - Define el plan de pruebas de software.
- A sistemas más críticos, mayor verificación estática.

Planes de pruebas (Sommerville) (I)

- Componentes principales del plan
- Proceso de prueba
- Descripción principal de las fases de prueba
- Requisitos de rastreo o seguimiento
- Se planifican pruebas de tal forma que se puedan testear individualmente todos los requerimientos
- Elementos probados
- Se especifican los productos del proceso de software a probar
- Calendarización de las pruebas
- Calendarización de todas las pruebas y asignación de recursos. Esto se vincula con el calendario general del proyecto

Planes de pruebas (Sommerville) (2)

- Componentes principales del plan (2)
  - Procedimiento de registro de las pruebas
    - Los resultados de la ejecución de las pruebas se deben grabar sistemáticamente. Debe ser posible auditar los procesos de prueba para verificar que se han llevado a cabo correctamente.
  - Requisitos de hardware y software
    - Esta sección señala el hardware y software requerido y la utilización estimada del hardware.
  - Restricciones
    - Se anticipan las restricciones que afectan al proceso de prueba, como, por ejemplo, la escasez de personal.
- Como otros planes, el plan de pruebas no es estático y debe revisarse regularmente.

Próxima clase:

Evolución de software