UNIVERSIDAD
Popular del cesar

Ingeniería de Sistemas

UNIVERSIDAD
Popular del cesar

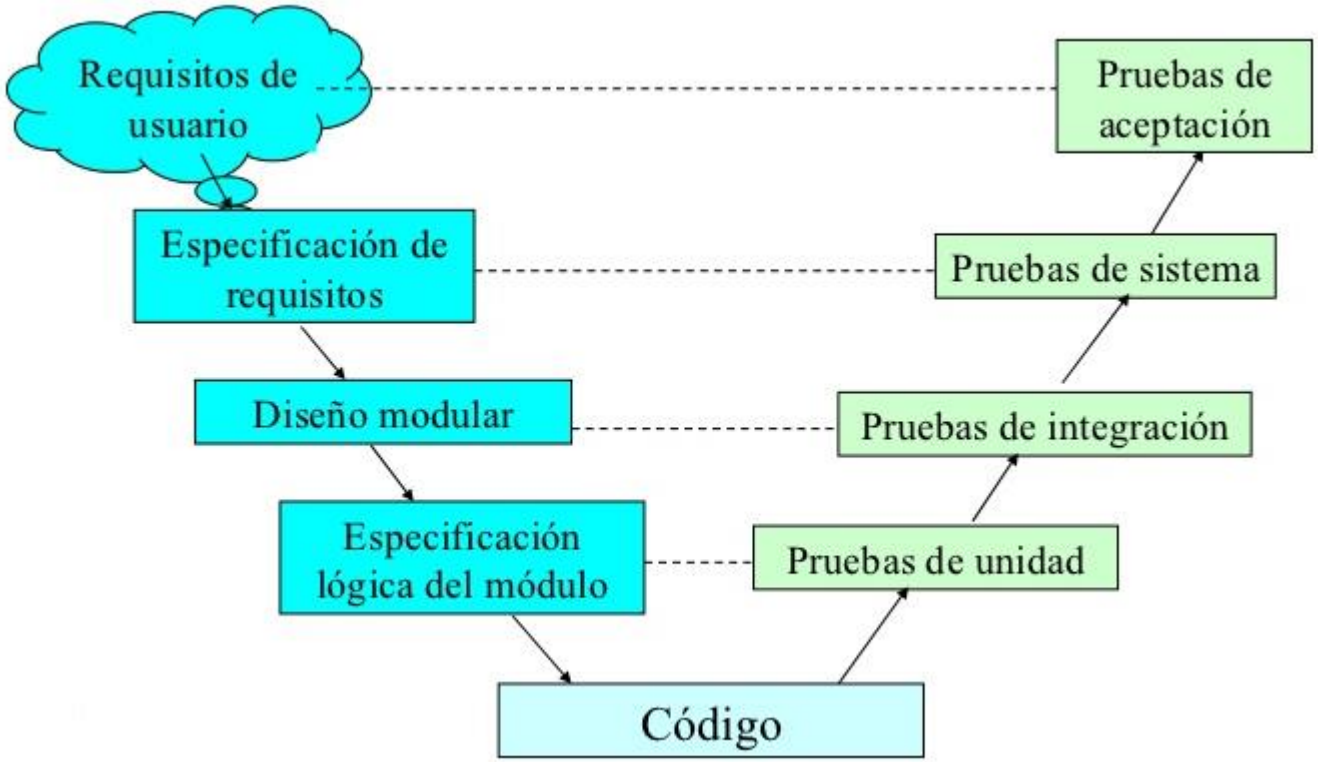
Ingeniería de Software II

Unidad No. 1
Pruebas de Software
Niveles de pruebas

UNIVERSIDAD Popular del cesar

1.4. Niveles de pruebas

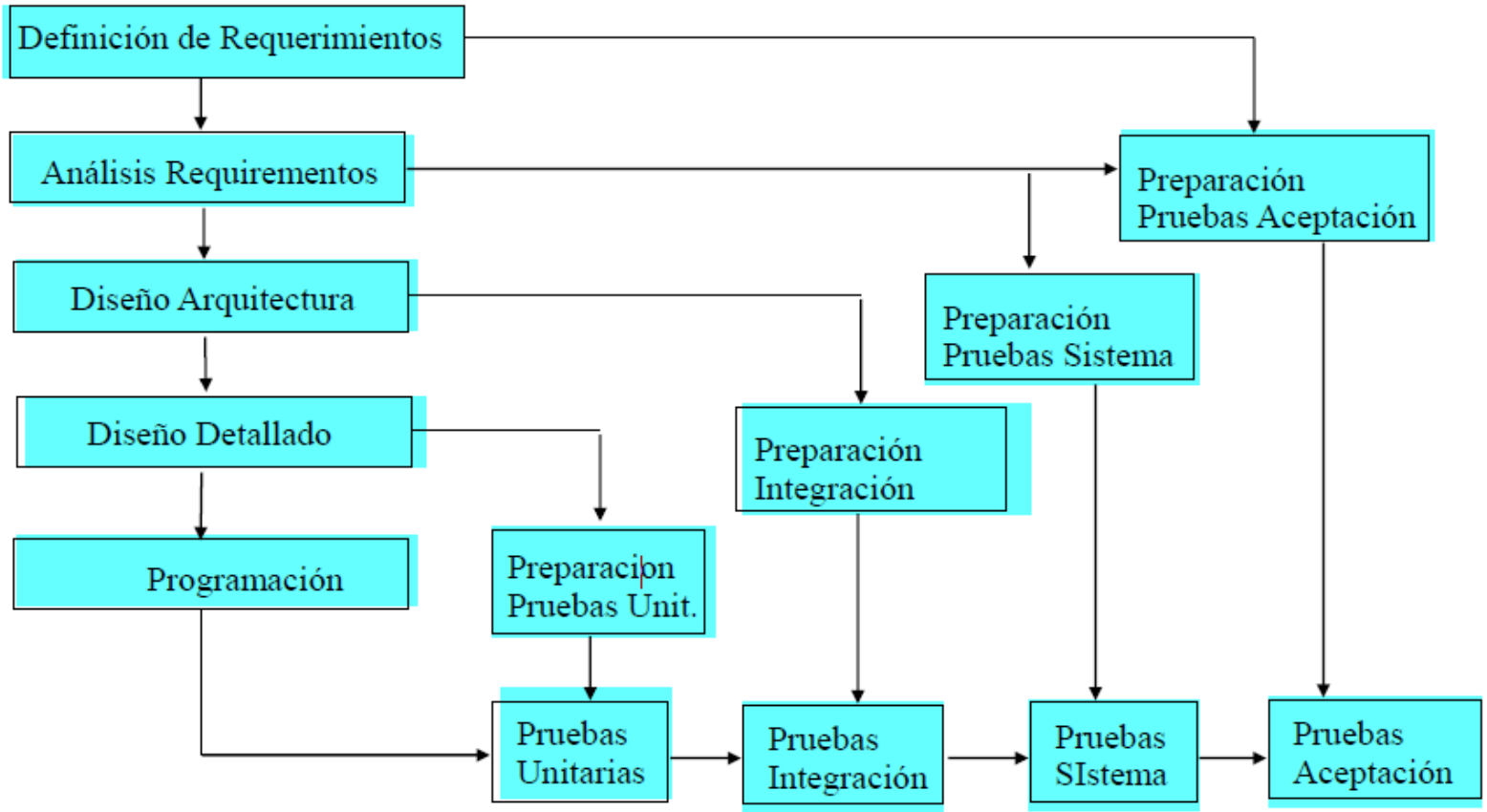
Las pruebas se llevan a cabo en diferentes niveles, para cada fase del desarrollo del software se define un nivel de prueba. Dentro de cada nivel de prueba el probador debe asegurarse de que los resultados cumplen con la verificación y la validación del software.



UNIVERSIDAD Popular del cesar

1.4. Niveles de pruebas

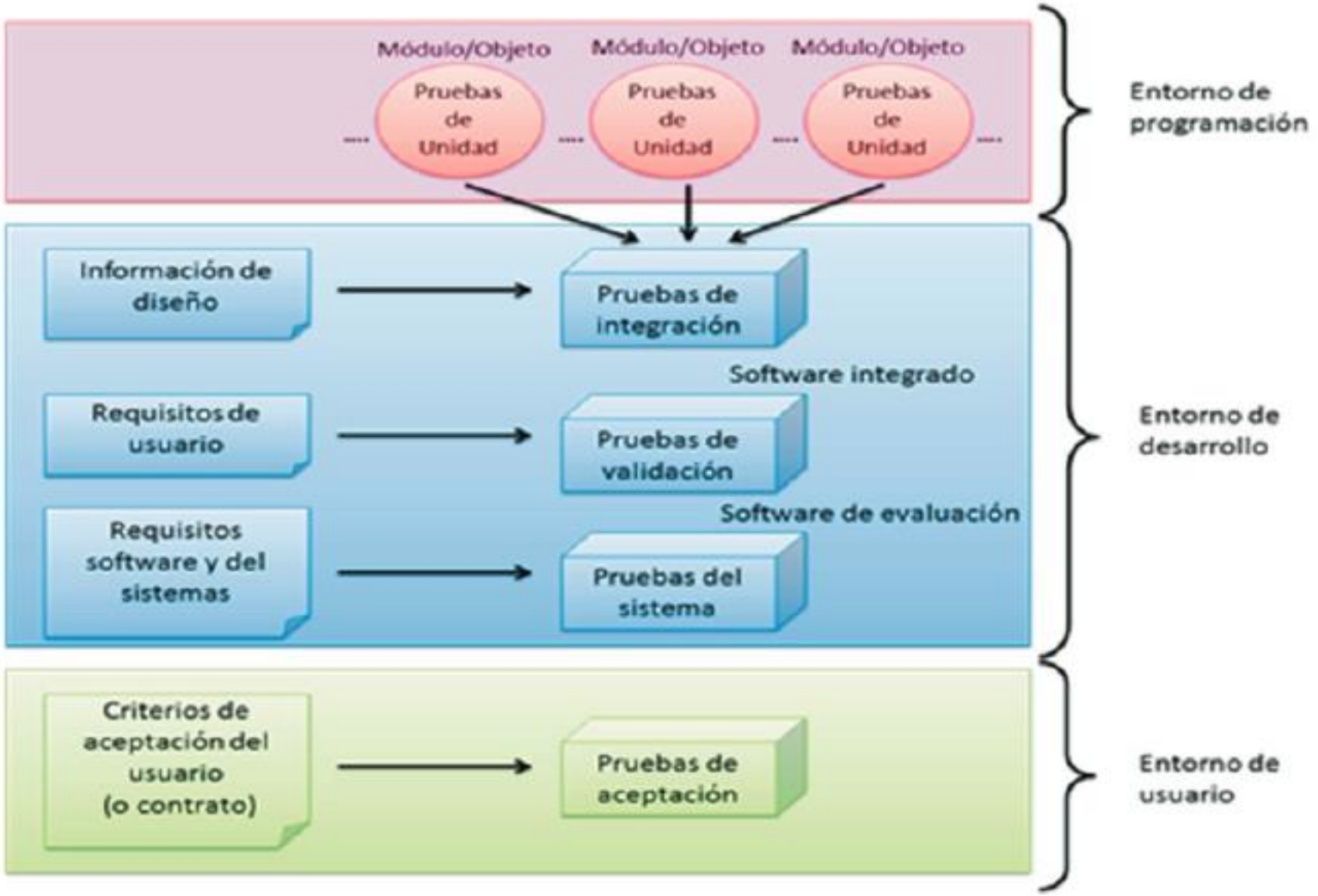
Para cada fase del desarrollo del software se define un nivel de prueba.



Pruebas de Software

UNIVERSIDAD Popular del cesar

1.4. Niveles de pruebas



UNIVERSIDAD Popular del cesar

1.4.1. Pruebas unitarias

- El primer nivel de las pruebas es el de la **prueba unitaria** o de componente, que consiste en probar el correcto funcionamiento de una unidad de código.
- En este nivel se comprueba la **lógica**, la funcionalidad y especificación de cada unidad (componente, clase u objeto) del sistema de manera **aislada** con respecto al resto de unidades.
- Se hace la evaluación de cada uno de los bloque más pequeños con identidad propia en el sistema, y es realizada por el programador en su entorno de desarrollo.
- Las pruebas unitarias usan técnicas de caja blanca y caja negra. Tanto las pruebas de caja blanca como las de caja negra se han de aplicar para probar de la manera más completa posible un módulo.

UNIVERSIDAD Popular del cesar

1.4.1. Pruebas unitarias

El **propósito** de las pruebas unitarias es encontrar errores en la lógica, datos o algoritmos en componentes o subsistemas individuales. Los casos de prueba que se generan deben:

- Tratar de detectar errores en los algoritmos y la lógica
- Tratar de detectar errores en la manipulación de las estructuras de datos
- Tratar de detectar errores en el llamado a otros módulos
- Identificar todos los caminos posibles del módulo y tratar de hacer casos de prueba que los cubran
- Tratar de detectar errores usando datos límites

UNIVERSIDAD Popular del cesar

1.4.1. Pruebas unitarias

¿Qué son las pruebas unitarias?

- Las **pruebas unitarias** (también test unitarios, o unit testing) son un método de pruebas de software que se realizan escribiendo fragmentos de código que testeará **unidades de código fuente**.
- El objetivo es asegurar que cada unidad funciona como debería de forma independiente.

UNIVERSIDAD Popular del cesar

1.4.1. Pruebas unitarias

¿Qué probamos? - Según el tipo de componente

- Funciones individuales o métodos estáticos
- Clases de objetos
  - Pruebas aisladas de los métodos
  - Asignación y consulta de atributos
  - Pruebas de secuencias de operaciones
- Probar también el manejo de errores (Excepciones)

UNIVERSIDAD Popular del cesar

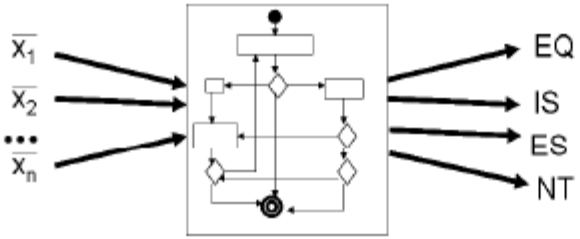
1.4.1. Pruebas unitarias

Existen dos enfoques principales para el **diseño de casos de prueba:**

- Enfoque funcional o de caja negra.
- Enfoque estructural o de caja blanca.

Pruebas de unidad (a nivel de método)

- Pruebas de caja blanca



- **Idea**: realizar un seguimiento del código fuente según se van ejecutando los casos de prueba

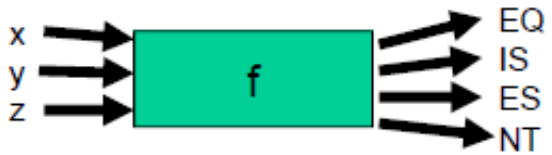
UNIVERSIDAD Popular del cesar

1.4.1. Pruebas unitarias

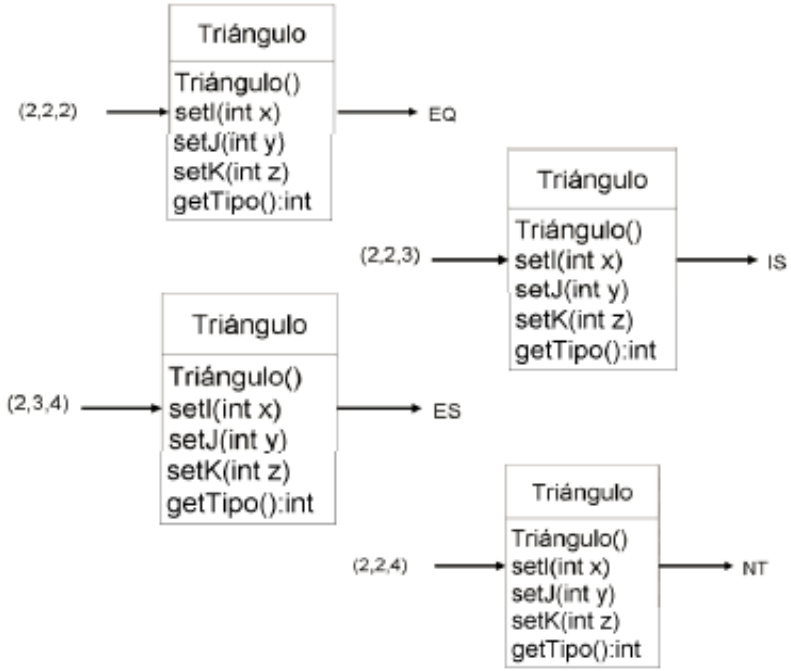
- Enfoque funcional o de caja negra.

Pruebas de unidad (a nivel de método)

- Pruebas de caja negra



- Entrada: longitud lados
  - (x, y, z)
- Salida: tipo triangulo
  - EQ (Equilatero)
  - IS (Isosceles)
  - ES (Escaleno)
  - NT (No triangulo)



UNIVERSIDAD Popular del cesar

1.4.1. Pruebas unitarias

Ejemplo

Entrada Tipo	Objeto/Variable de Estado(Valores)	Clases válidas (valores válidos)	Clases inválidas (valores inválidos)	# de Clases x Factor
CódigoDeUsuario String	Usuario: CódigoDeUsuario, Estado (Normal/ Sancionado/ Retirado)	1. Formato válido (8 dígitos (0-9) , Existe, Estado normal	2. Existe, Sancionado o Retirado, 3. No existe 4. Formato inválido (menos/más caracteres, no dígitos) incluye "en blanco" (cero caracteres).	4

Las Particiones de la variable CódigoDeUsuario

Variable Entrada:		CódigoDeUsuario	Tipo:	String
#	Válido/No Válido	Formato	Objeto/Variable Estado	
			Usuario	Estado
1	Válido	8 dígitos	Existe	Estado Normal
2	No válido	8 dígitos	Existe	Sancionado
				Retirado
3	"	8 dígitos	No Existe	
4	"	En blanco		
		No dígitos		
		No 8 dígitos		

UNIVERSIDAD Popular del cesar

1.4.1. Pruebas unitarias

Estructura de las pruebas unitarias

- **Organizar:** En esta parte de la prueba se establecen las condiciones iniciales para poder realizarla, así como el resultado que se esperan obtener.
- **Accionar**: Es la parte de ejecución, tanto del fragmento de código de la prueba, como del fragmento de código a testear.
- **Comprobar:** Por último, se realiza la comprobación para verificar que el resultado obtenido, coincide con el esperado.

UNIVERSIDAD Popular del cesar

1.4.1. Pruebas unitarias

Frameworks para pruebas unitarias

- **Java**

  Para Java se cuenta con Junit

- **PHP**

  Para PHP dispone de PHPUnit

- **Javascript**
  - Para Javascript, existe una mayor variedad de frameworks, entre los que podría destacar MochaJS o JEST.
  - En el caso de MochaJS, trabaja tan bien con el front end como con el back end.
  - Mientras que JEST, destaca por ser sencillo.

- **C#**

  NUnit es el framework Open source de pruebas unitarias para .NET más conocido. Las pruebas pueden realizarse tanto desde la consola, como con Visual Studio, o a través de terceros.

UNIVERSIDAD Popular del cesar

1.4.2. Pruebas de integración

Aún cuando los módulos de un programa funcionen bien por separado es necesario probarlos conjuntamente. Un módulo o método puede tener un efecto adverso o inadvertido sobre otro módulo. Por lo tanto, es necesario probar el software **ensamblando todos los módulos probados** previamente. Realizado por los desarrolladores de los módulos que serán integrados.

En este nivel se utilizan técnicas que verifican el correcto manejo de las entradas y salidas del software (pruebas funcionales). Generalmente implementan técnicas de Caja negra, basadas en las especificaciones de las interfaces.

UNIVERSIDAD Popular del cesar

1.4.2. Pruebas de integración

El propósito de las pruebas de integración es encontrar errores en las interfaces entre los módulos. Los casos de prueba que se generan deben:

- Tratar de detectar errores en los formatos de intercambio de datos
- Tratar de detectar errores en el orden en que interactúan los módulos, la sincronización y los tiempos de respuesta

UNIVERSIDAD Popular del cesar

1.4.2. Pruebas de integración

Existen diferentes estrategias de integración:

- **Incremental:** Se combina el siguiente módulo que se debe probar con el conjunto de módulos que ya han sido probados.
- **Integración no incremental**. Se prueba cada módulo por separado y luego se integran todos de una vez y se prueba el programa completo.
- **Guiadas por la arquitectura**: Los componentes se integran según hilos de funcionalidad

UNIVERSIDAD Popular del cesar

1.4.2. Pruebas de integración

Integración incremental

- Los componentes se integran y prueban poco a poco. Se combina el siguiente módulo que se debe probar con el conjunto de módulos que ya han sido probados.

Existen dos tipos:

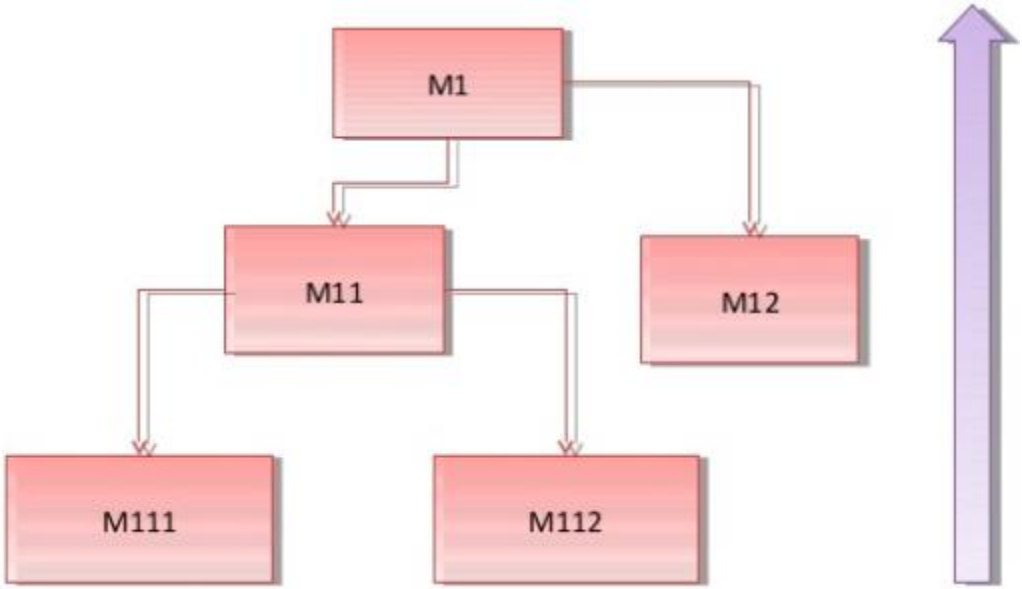
- **Integración incremental descendente** (componentes de funcionales)
- **Integración incremental ascendente** (componentes de infraestructura, acceso a BD)

UNIVERSIDAD Popular del cesar

1.4.2. Pruebas de integración

Integración incremental Ascendente (Bottom-Up)

1. Se comienza por los módulos hoja (pruebas unitarias)
2. Se combinan los módulos según la jerarquía
3. Se repite en niveles superiores



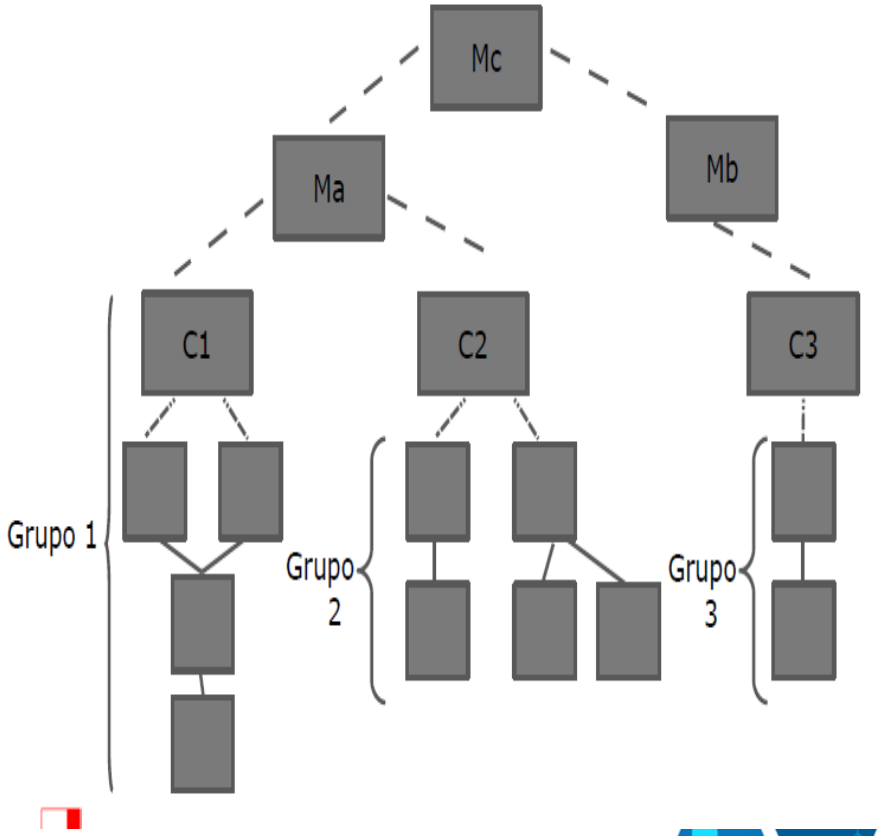
UNIVERSIDAD Popular del cesar

1.4.2. Pruebas de integración

Integración incremental Ascendente (Bottom-Up)

En estas pruebas se implementan los siguientes pasos:

1. Se combinan los módulos de bajo nivel en grupos que realicen una subfunción específica
2. Se escribe un *controlador* (un programa de control de la prueba) para coordinar la entrada y salida de los casos de prueba.
3. Se prueba el grupo
4. Se eliminan los controladores y se combinan los grupos moviéndose hacia arriba por la estructura del programa.

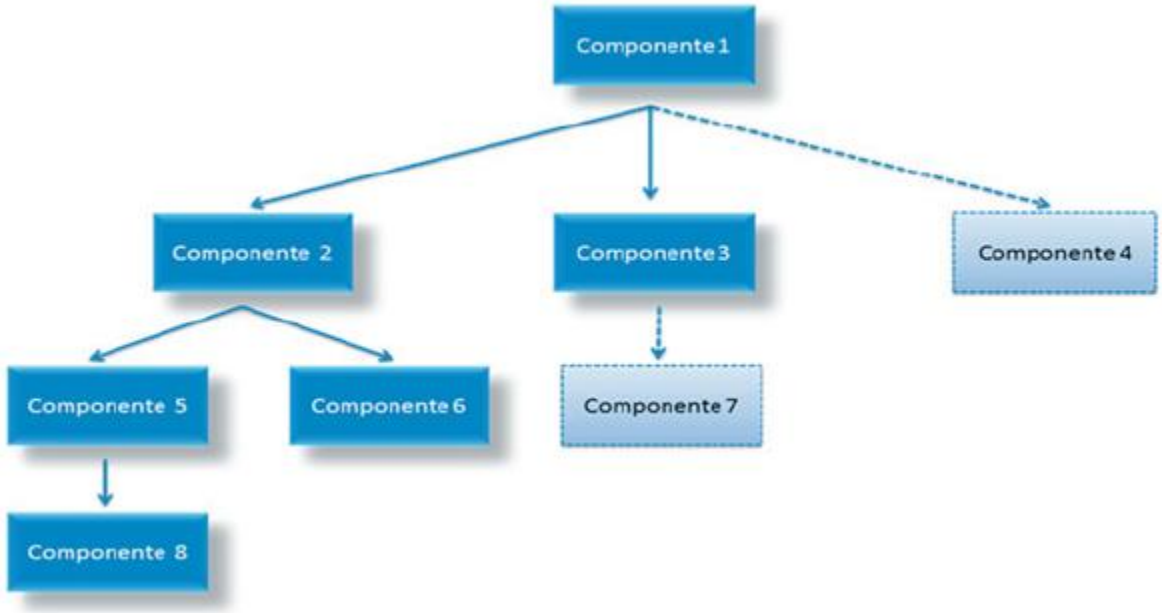


UNIVERSIDAD Popular del cesar

1.4.2. Pruebas de integración

Integración incremental Descendente (Top-Down)

En estas pruebas se comienza con el módulo superior y se avanza hacía los módulos inferiores



UNIVERSIDAD Popular del cesar

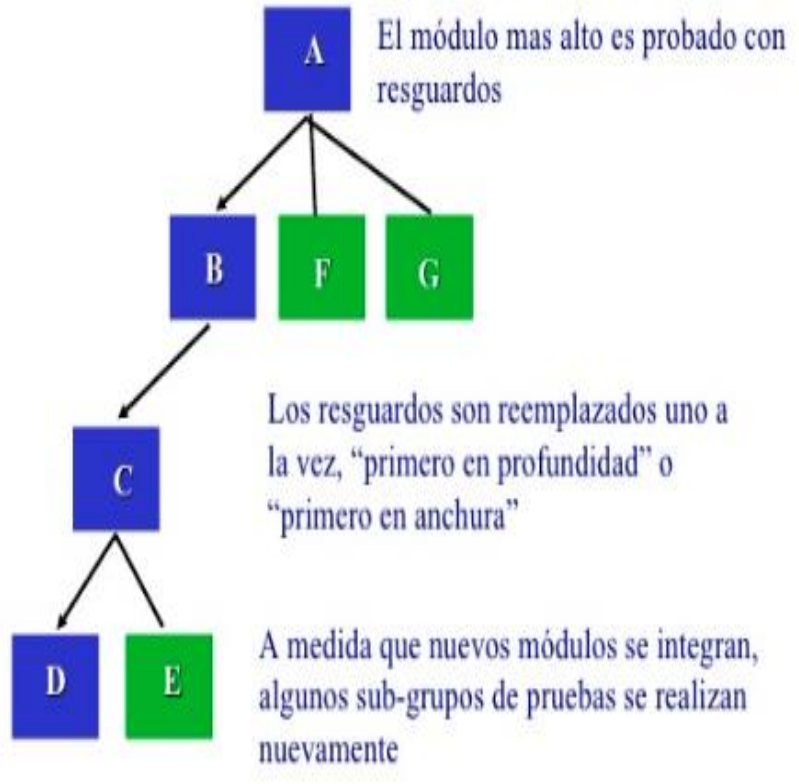
1.4.2. Pruebas de integración

Integración incremental

Descendente (Top-Down)

Existen dos formas básicas de hacer esta integración:

- Primero en profundidad, completando ramas del árbol
- Primero en anchura, completando niveles de jerarquía



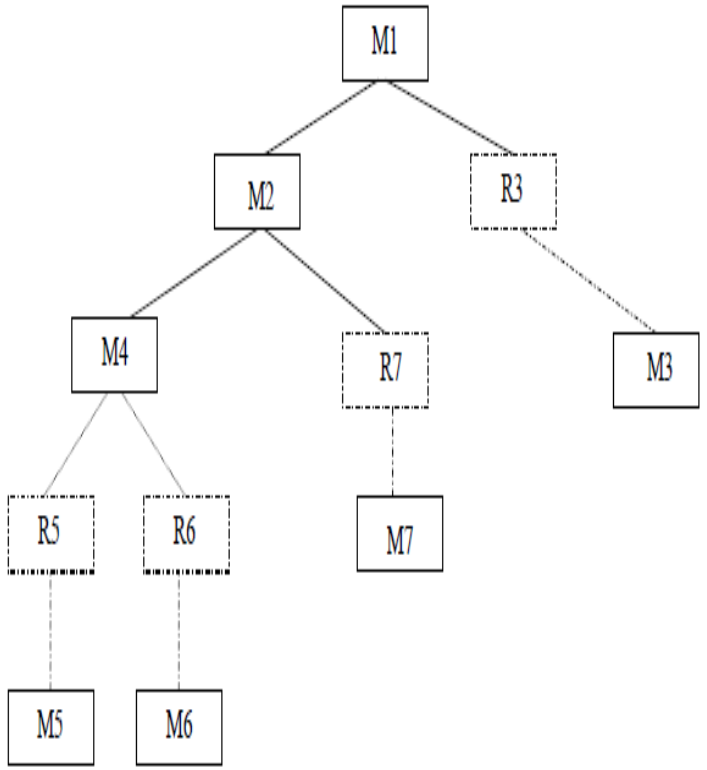
UNIVERSIDAD Popular del cesar

1.4.2. Pruebas de integración

Integración incremental Descendente (Top-Down)

En estas pruebas se implementan los siguientes pasos:

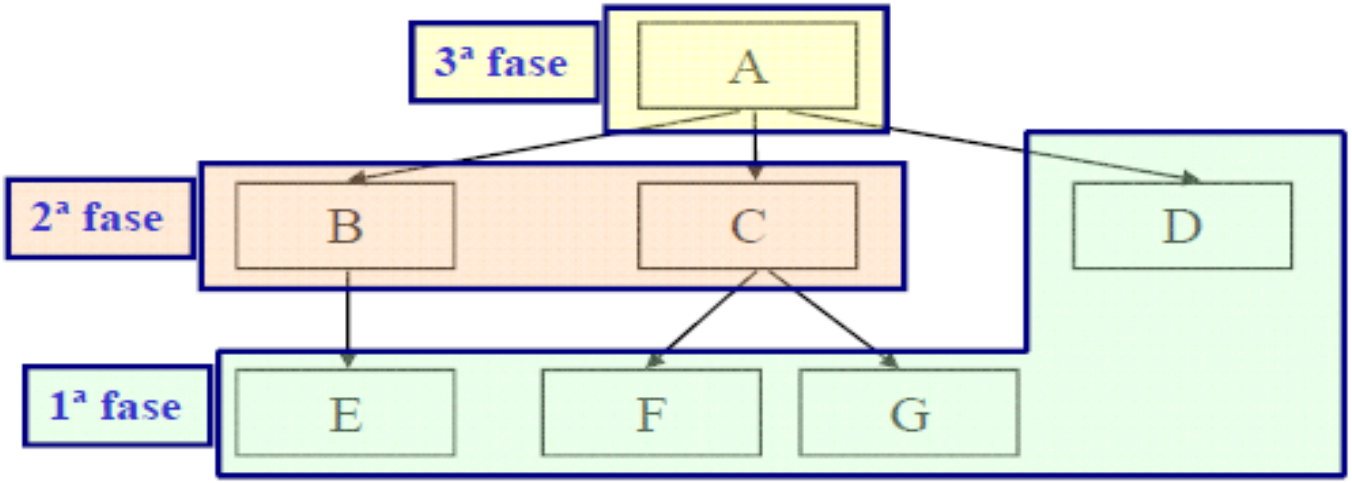
1. Se usa el módulo de control principal como controlador de la prueba, creando ***resguardos*** (módulos que simulan el funcionamiento de los módulos que utiliza el que está probando) para todos los módulos directamente subordinados al módulo de control principal.
2. Dependiendo del enfoque e integración elegido (es decir, primero-en-profundidad, o primero-en-anchura) se van sustituyendo uno a uno los resguardos subordinados por los módulos reales.
3. Se llevan a cabo pruebas cada vez que se integra un nuevo módulo.
4. Tras terminar cada conjunto de pruebas, se reemplaza otro resguardo con el módulo real.



UNIVERSIDAD Popular del cesar

1.4.2. Pruebas de integración

- Incremental Ascendente (Bottom-Up)
  1. Unitarias de E, F, G y D
  2. Integración de (B con E), (C con F) y (C con G)
  3. Integración de (A con B), (A con C) y (A con D)



- Incremental Descendente (Top-Down)
  - Primero en **profundidad**, completando ramas del árbol
    - (A, B, E, C, F, G, D)
  - Primero en **anchura**, completando niveles de jerarquía
    - (A, B, C, D, E, F, G)

UNIVERSIDAD Popular del cesar

1.4.2. Pruebas de integración

Pruebas basadas en hilos

- Integra el conjunto de clases necesarias para responder a una entrada o evento del sistema.
- Cada hilo se integra y prueba individualmente
- Los **casos de usos** como la unidad de pruebas de integración
- El **diagrama de secuencia** que representan varios escenarios.
- Un diagrama de secuencia presenta varios elementos importantes para la prueba de integración:
  - El conjunto de clases (objetos) interactúan.
  - Las interacciones entre los objetos , mostradas como secuencia de mensajes que activan métodos.

UNIVERSIDAD Popular del cesar

1.4.2. Pruebas de integración

Pruebas basadas en hilos

- **Cada caso de uso determinara los casos de pruebas.**
- **Uno por cada diagrama de secuencia, originados por dos elementos:**
  - **El estado de los diferentes componentes del diagrama y**
  - **Los posibles valores que pueden tomar los diversos parámetros de los métodos invocados**

Posibles estados según el tipo de elemento

Tipo de elemento	Posibles estados
objeto	Existe, no existe, existe pero es de versión inadecuada
Recurso compartido (archivo, BD)	Existe, No existe, No es accesible (falta autorización, exceso de usuarios, bloqueado)
Dispositivo (red, impresora)	Listo, en problemas, desconectado

UNIVERSIDAD Popular del cesar

1.4.2. Pruebas de integración

Pruebas basadas en hilos

Description

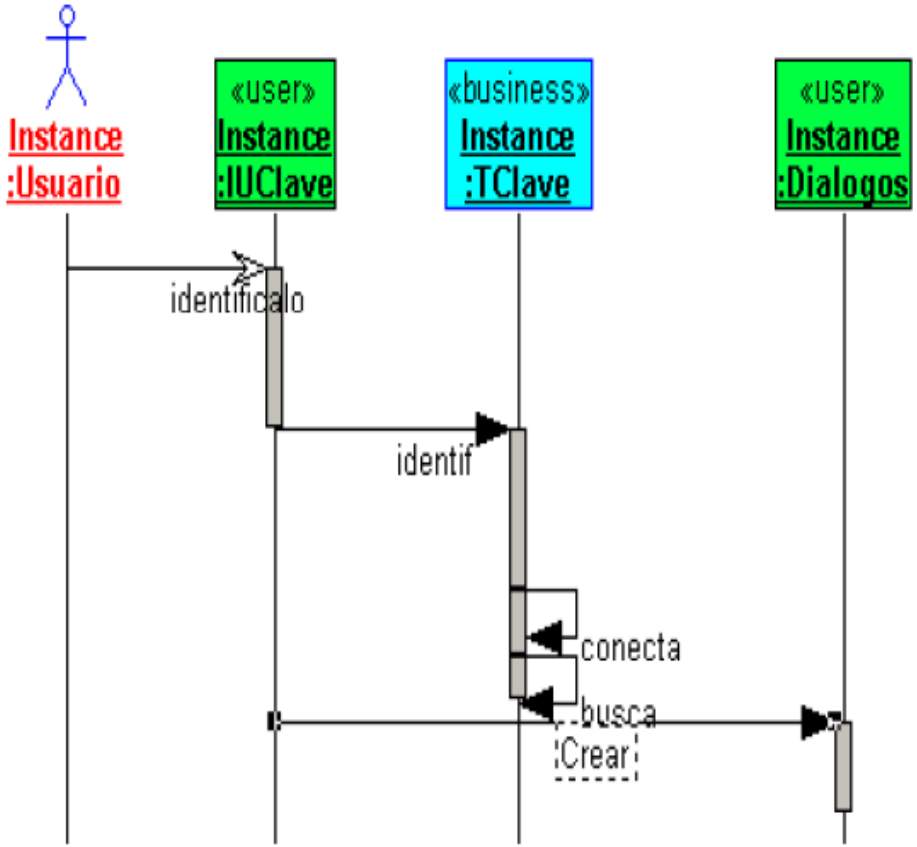
Usuario escribe nombre y contraseña y oprime Listo

La interfaz solicita a la entidad que identifique al usuario

La entidad se conecta a base de datos

La entidad busca el nombre del usuario

La Interfaz envía un mensaje con el resultado



UNIVERSIDAD Popular del cesar

1.4.2. Pruebas de integración

Pruebas basadas en hilos

Estados del ejemplo

Elemento	Estados
IUClave	Presente, ausente
Tclave	Presente, ausente
Dialogos	Presente, ausente
Base de datos	Activa, con problemas

Valores de entrada a considerar

juan, orion7	(nombre y contraseñas registradas)
juan, qwerty	(nombre registrado, contraseña mal)
juan	(falta contraseña)
, orion7	(falta nombre)
ulises, df7th	(nombre no registrado)

UNIVERSIDAD Popular del cesar

1.4.2. Pruebas de integración

Pruebas basadas en hilos

Generación de casos de pruebas

1. Para cada diagrama de secuencia, generar los estados posibles de los elementos involucrados
2. Para cada método en las interacciones incluidas en el diagrama de secuencia determinar los conjuntos de valores adecuados, según las particiones internas y según si permiten continuar la cadena de llamadas a otras clases involucradas y seleccionar valores representativos.

3. Con cada estado y cada representante de conjunto de datos se forma un caso de prueba, combinando los diferentes valores de cada categoría.

UNIVERSIDAD Popular del cesar

1.4.3. Pruebas de sistema

Una vez que se han probado los componentes y la integración de los mismos, entramos en el tercer nivel de prueba, donde se va a comprobar si el producto cumple con los requisitos especificados.

Las pruebas de sistema deben de verificar los requisitos funcionales y no funcionales del sistema y las características de calidad. Para ello se aplican técnicas de prueba de caja negra.

Su propósito es encontrar errores en el comportamiento del sistema de acuerdo con la especificación de requerimientos. Realizadas por un grupo diferente al de desarrollo.

UNIVERSIDAD Popular del cesar

1.4.3. Pruebas de sistema

Este tipo de pruebas tiene como propósito revisar el sistema para verificar que se han integrado adecuadamente todos los elementos del sistema entre el hardware y el software, además, que realizan las funciones adecuadas. Concretamente se debe comprobar que:

- Se cumplen los requisitos funcionales establecidos.
- El funcionamiento y rendimiento de las interfaces hardware, software y de usuario.
- La adecuación de la documentación de usuario.
- Rendimiento y respuesta en condiciones límite y de sobrecarga.

UNIVERSIDAD Popular del cesar

1.4.3. Pruebas de sistema

Tipo de Pruebas del Sistema

- **Prueba de Recuperación**: Es una prueba del sistema que fuerza el fallo del software de muchas formas y verifica que la recuperación se lleva a cabo apropiadamente.
- **Prueba de Seguridad**: Intenta verificar que los mecanismos de protección incorporados en el sistema lo protegerán, de hecho, de acceso impropios.
- **Prueba de Resistencia**: Están diseñadas para enfrentar a los programas con situaciones anormales.
- **Prueba de Rendimiento**: Está diseñada para probar el rendimiento del software en tiempo de ejecución dentro del contexto de un sistema integrado.

UNIVERSIDAD Popular del cesar

1.4.3. Pruebas de sistema

Tipo de Pruebas del Sistema

- Pruebas de carga.
- Pruebas de estrés.
- Pruebas de escalabilidad.
- Pruebas de portabilidad.
- Pruebas de Rendimiento
- Pruebas de instalación
- Pruebas de seguridad
- Pruebas de usabilidad
- Pruebas de compatibilidad
- Pruebas de mantenibilidad

UNIVERSIDAD Popular del cesar

1.4.4. Pruebas de aceptación

Estas pruebas son realizadas en el entorno del usuario, para validar la aceptación por parte del cliente, comprobando que el sistema está listo para ser implantado. Estas pruebas se caracterizan por:

- Participación activa del usuario, que debe ejecutar los casos de prueba ayudado por miembros del equipo de pruebas.
- Están enfocadas a probar los requisitos de usuario, a demostrar que se cumplen los requisitos, los criterios de aceptación o el contrato establecido.
- Está considerada como la fase final del proceso para crear una confianza en que el producto es el apropiado para su uso en explotación.

UNIVERSIDAD Popular del cesar

Referencias

- Roger, P. (2010). Ingeniería del Software: Un Enfoque Práctico. 7ª. Edicion. McGraw Hill.
- Ian S. (2011). Ingeniería del Software. 9ª. Edición. Pearson Educación, México.
- Salvador, S. Miguel, S. y Daniel, Rodríguez. (2012). Ingeniería del software. Un enfoque desde la guía SWEBOK. Alfaomega.
- José, S. (2015). Pruebas de Software. Fundamentos y Técnicas. Universidad Politécnica de Madrid, España.
- Beatriz F. (2013). Técnicas de Pruebas de Software. Universidad del Valle, Colombia.

UNIVERSIDAD
Popular del cesar