TIPOS DE PRUEBAS DE SOFTWARE

OBJETIVOS

- Identificar los diferentes niveles y tipos de pruebas que pueden encontrarse.
- Determinar el/los tipo(s) de prueba que se deberían realizar de acuerdo al contexto

1

CLASIFICACiÓN

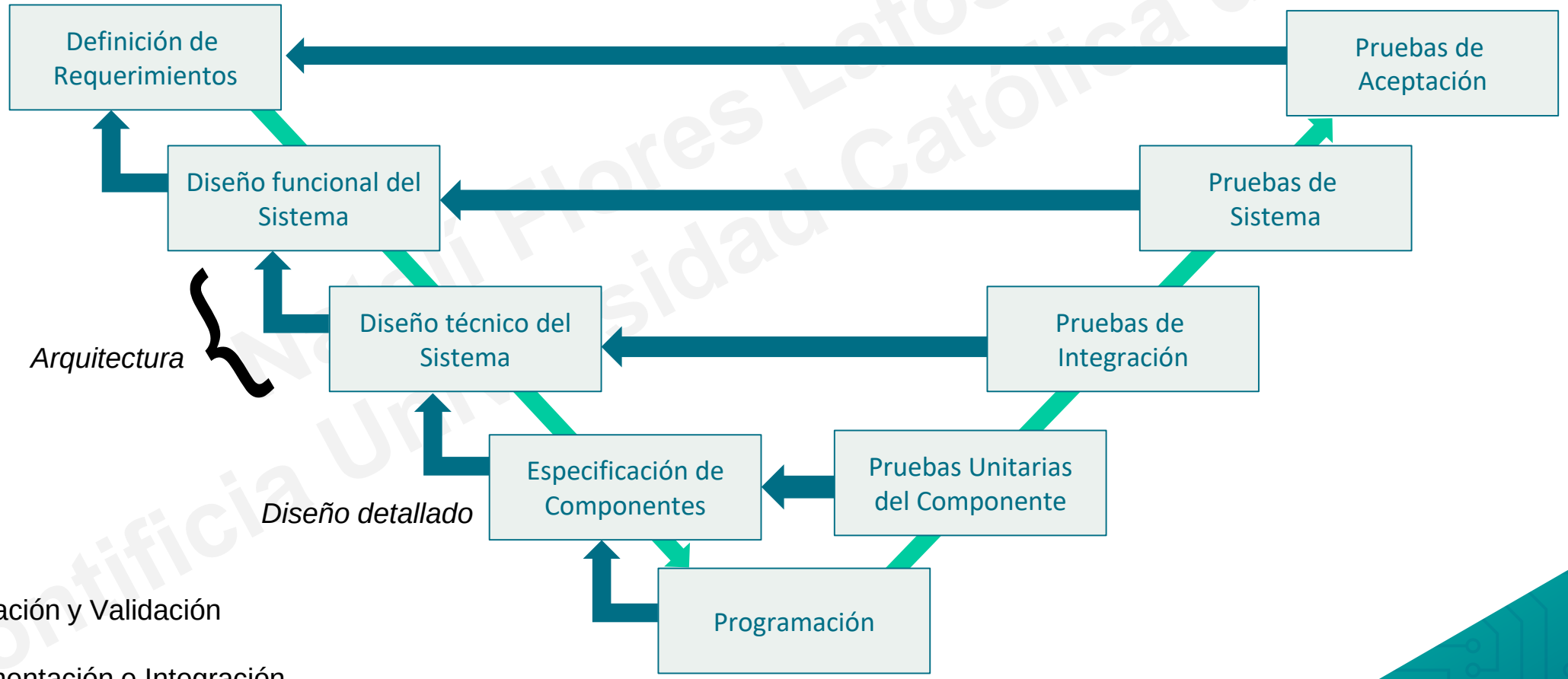
Generalidades

¿Cómo clasificar las pruebas?

- Podemos clasificarlas por nivel, por el método o técnica de aplicación, por los objetivos que buscan, por el tipo de requerimientos que revisan, etc.

- Vamos a ver dos formas

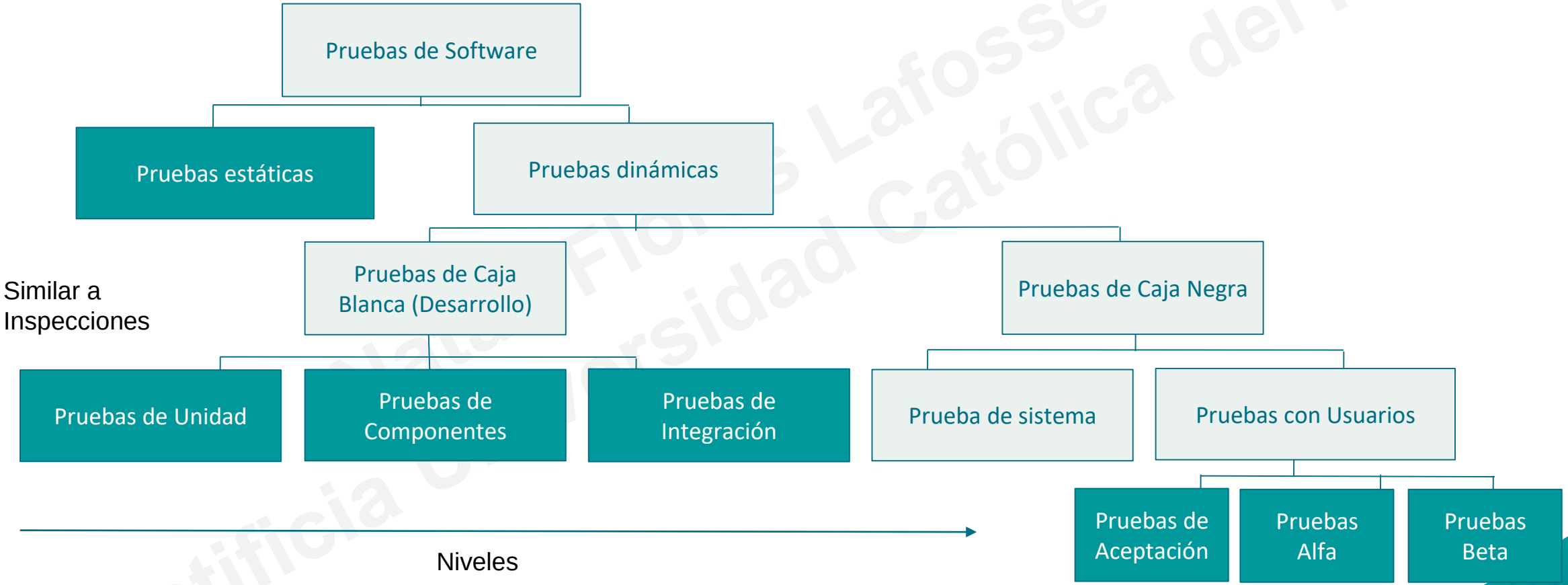
Modelo V



Verificación y Validación

Implementación e Integración

Árbol de tipos de pruebas

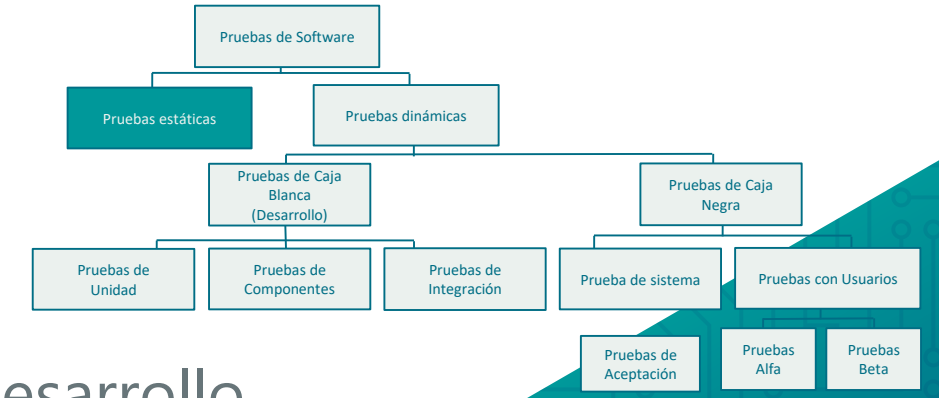


2

ESTÁTICAS VS DINÁMICAS

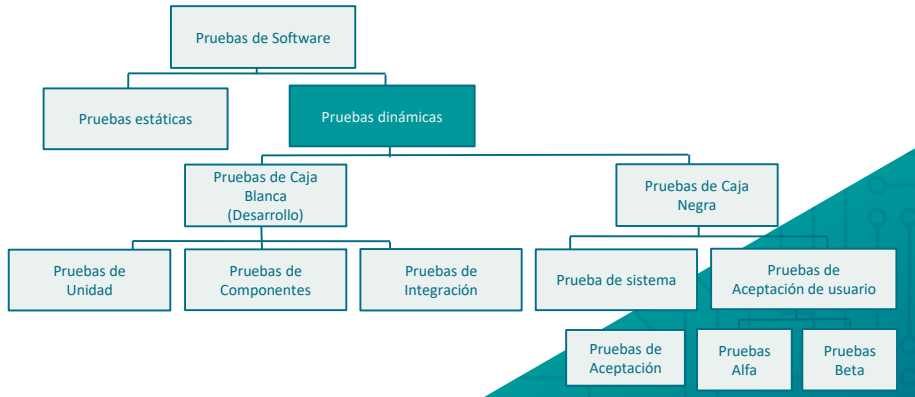
PRUEBAS ESTÁTICAS

- Se refieren al análisis del software (o elementos relacionados) cuando no se encuentra ejecutando.
- Por ejemplo:
  - Código fuente (Complejidad, Estándares)
  - Especificación de Requisitos
  - Documentación de casos de uso
- Se busca encontrar errores antes de mayor desarrollo



PRUEBAS DINÁMICAS

- Son pruebas en que se realizan con el código ya en ejecución.
- Pueden clasificarse como Caja Blanca o Caja Negra.
- Así mismo, pueden tipificarse por NIVELES.

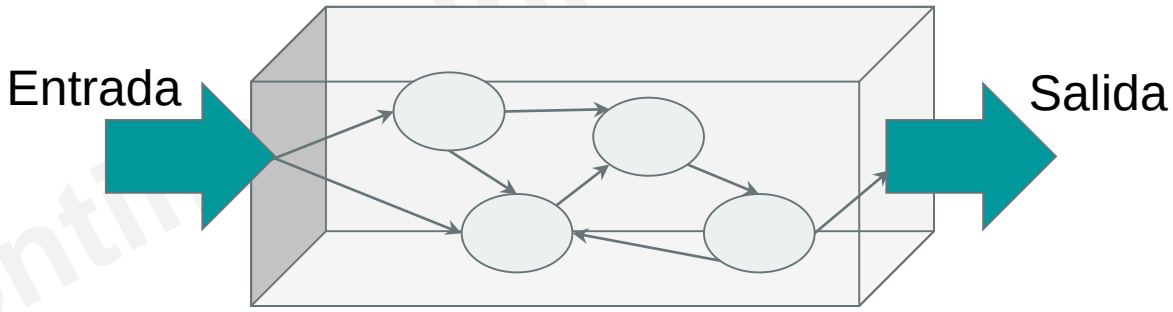


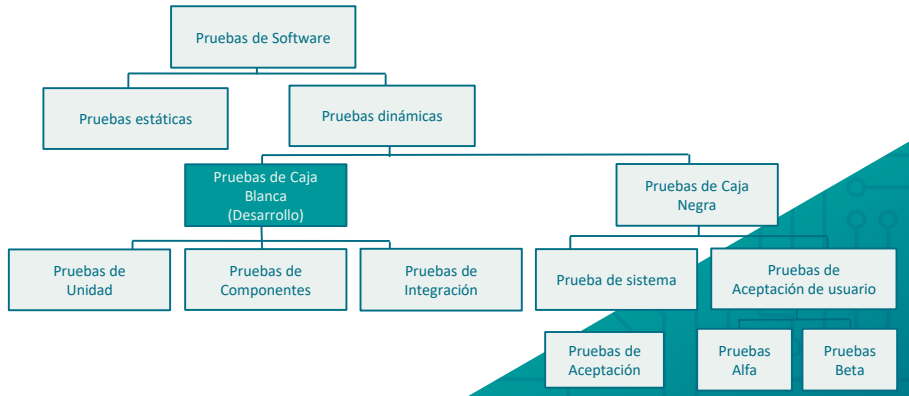
3

CAJA BLANCA VS CAJA NEGRA

PRUEBAS DE CAJA BLANCA

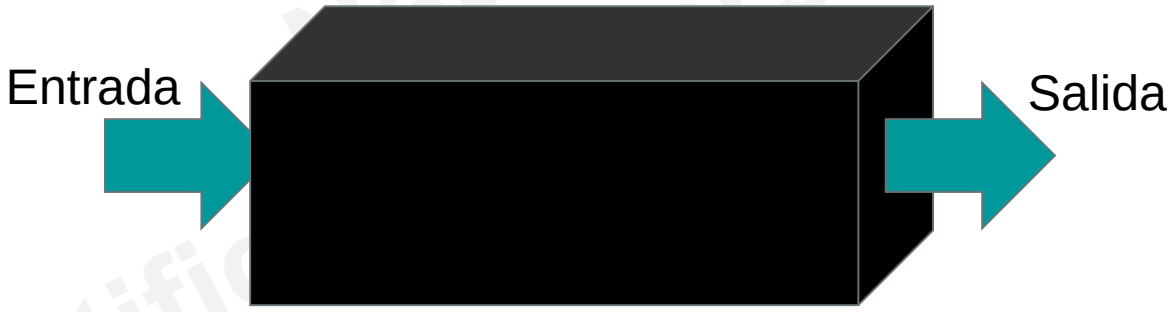
- Se analiza como el procedimiento va pasando parte por parte
- Se puede hacer a distintos niveles (granularidad)
- Pruebas de cobertura (¿Se pasa por todo el código?)

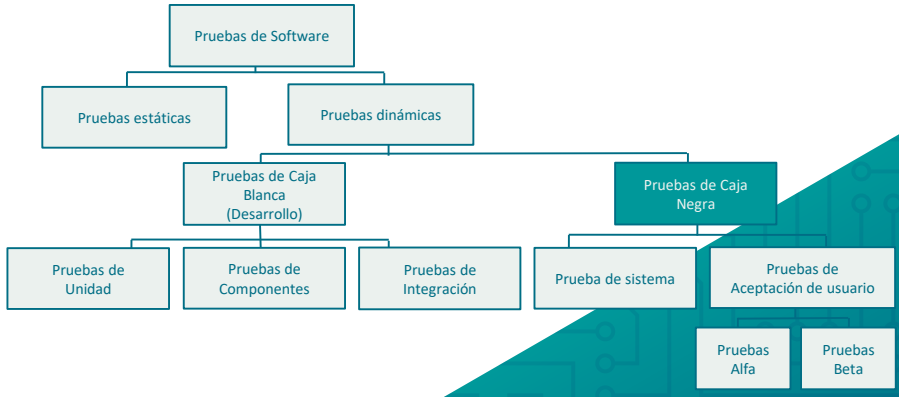




PRUEBAS DE CAJA NEGRA

- Sólo enfocarse en entradas VS salidas esperadas
- Se ve el componente/módulo sin el detalle interno.





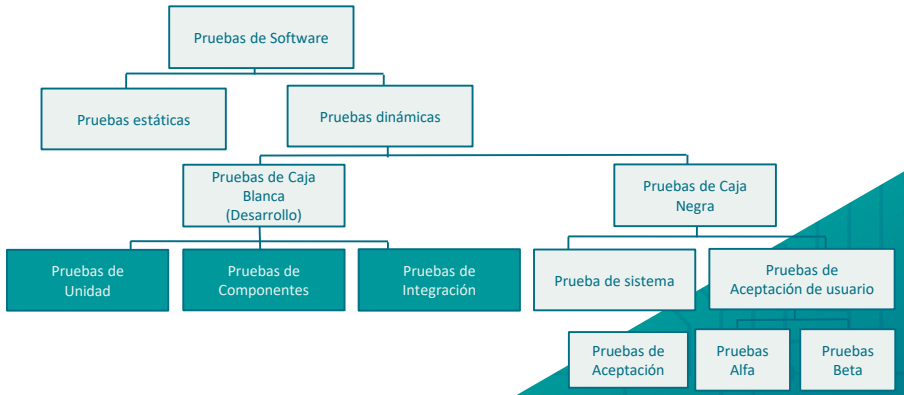
4

PRUEBAS POR NIVELES

Pruebas de Desarrollo por Niveles

Todas las actividades de pruebas en la elaboración del sistema, revisando los flujos de información.

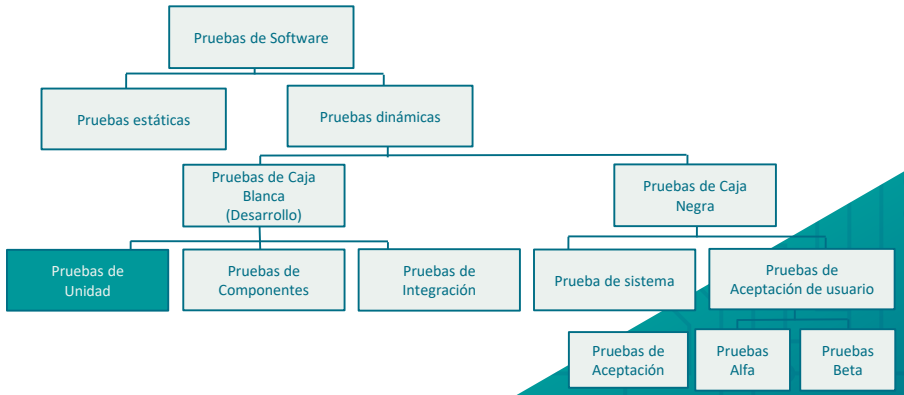
- Pruebas de Unidad
- Pruebas de Componente
- Pruebas de Integración



PRUEBAS DE UNIDAD (Unitarias)

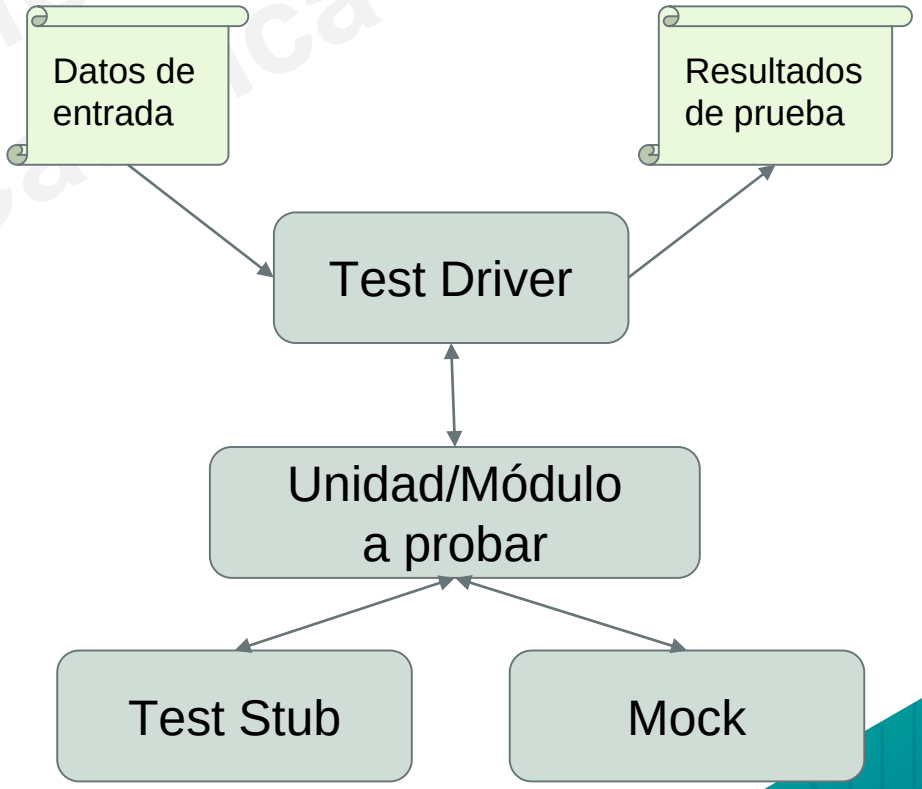
Se debe probar el objeto (o módulo) por completo.

- Todos los métodos
- Todos las salidas
- Todas las entradas
- Todos los estados
- Todas las herencias



PRUEBAS DE UNIDAD (Unitarias)

- **Test Driver (Manejador de prueba):** Objeto que llama al módulo a probar.
- **Test Stub:** Objeto (que el módulo a probar llama) que siempre devuelve un resultado estático.
- **Mock Object:** Objeto (que el módulo a probar llama) que devuelve resultados, sin que esté implementado.

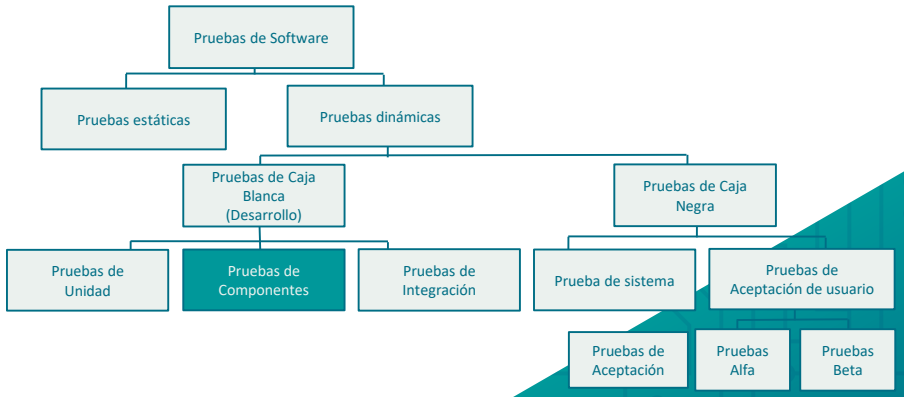


PRUEBAS DE COMPONENTES

Se enfocan en el uso de las interfaces entre los componentes.

- Poner a prueba la arquitectura
- Uso incorrecto de interfaz
- Errores temporización (carrera)

¿Caja Gris?



PRUEBAS DE COMPONENTES

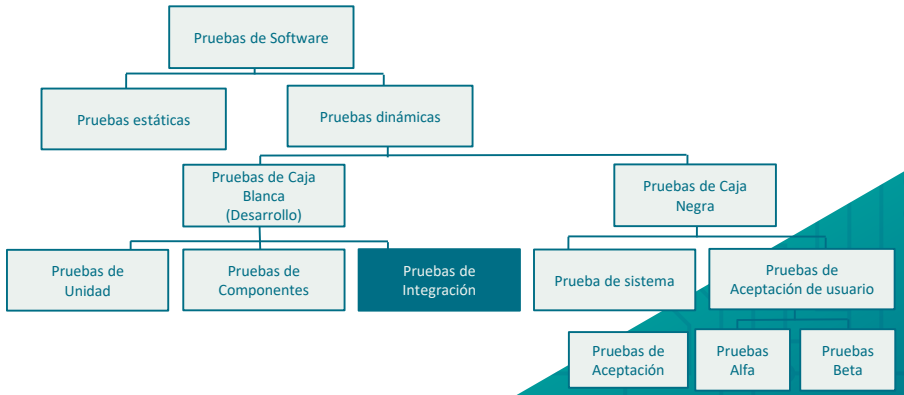
Lineamientos sugeridos

- Listar llamados de interfaces
- Forzar llamados a nulos (si no está el componente)
- Probar que pasa cuando falla el otro componente
- Pruebas de esfuerzo al enviar mensajes
- Pruebas de memoria compartida oner a prueba la arquitectura
- Uso incorrecto de interfaz
- Errores temporización (carrera)

PRUEBAS DE INTEGRACIÓN

Las pruebas de integración son parte de las pruebas de sistema.
Para integrar, se pueden usar diferentes estrategias:

- Top Down
- Bottom Up
- Sandwich
- BigBang

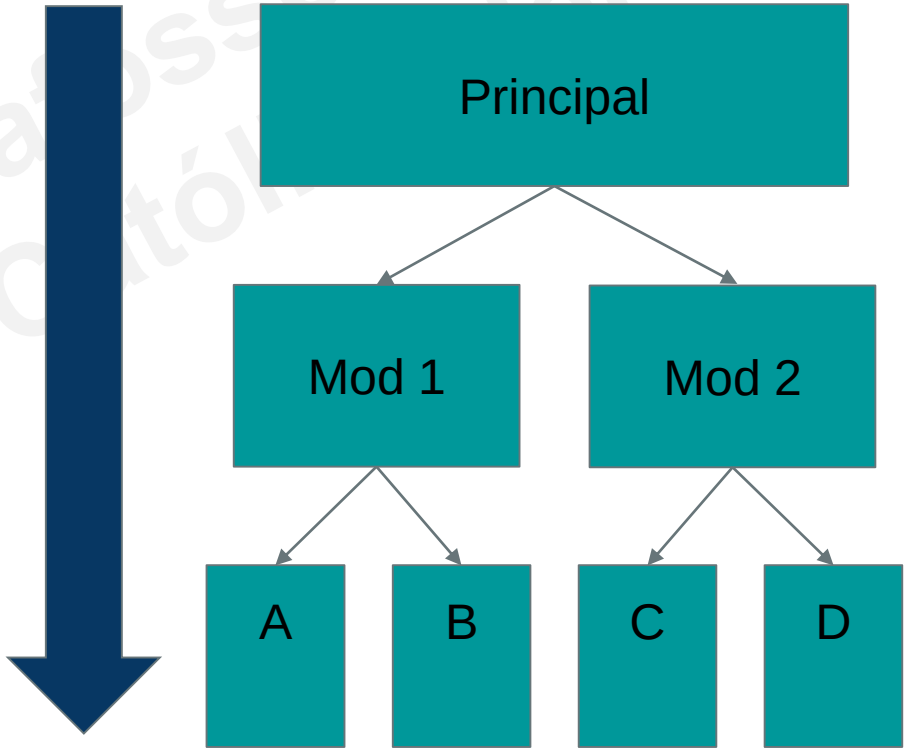


PRUEBAS DE INTEGRACIÓN

TOP DOWN

Se va probando la funcionalidad desde el módulo principal hasta los módulos inferiores.

- Requiere muchos stubs de prueba.
- Mayor abstracción



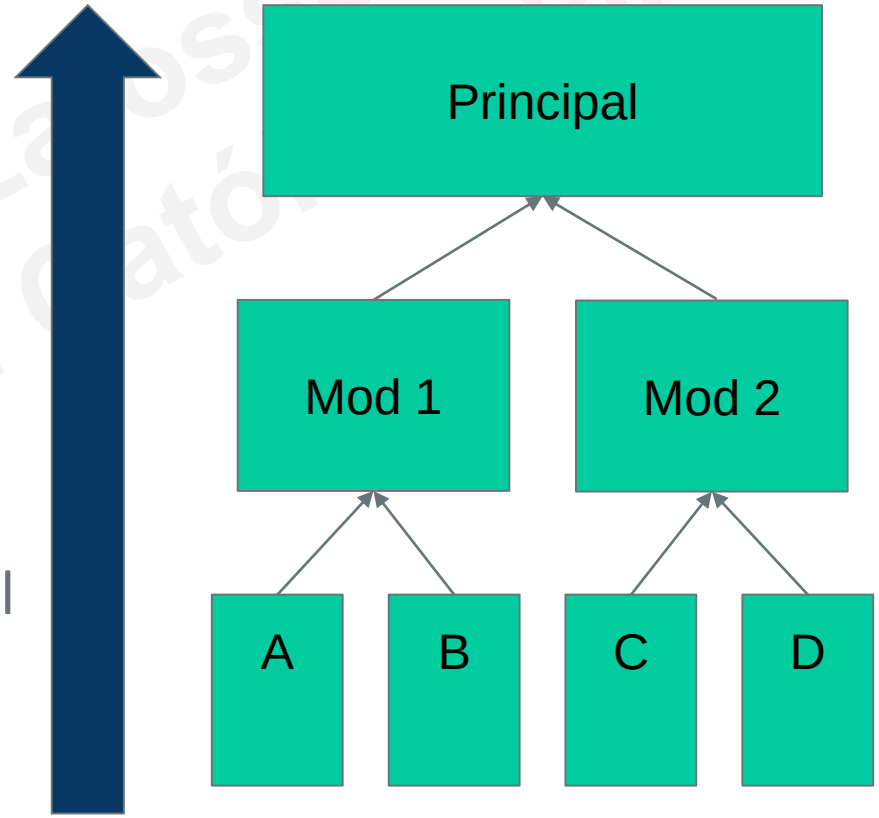
PRUEBAS DE INTEGRACIÓN

BOTTOM UP

Se empieza con los módulos más atómicos. Luego se van probando los de mayor jerarquía.

- Errores pueden posponerse hasta el final (puede ser costoso)

- Recomendable cuando hay reutilización.

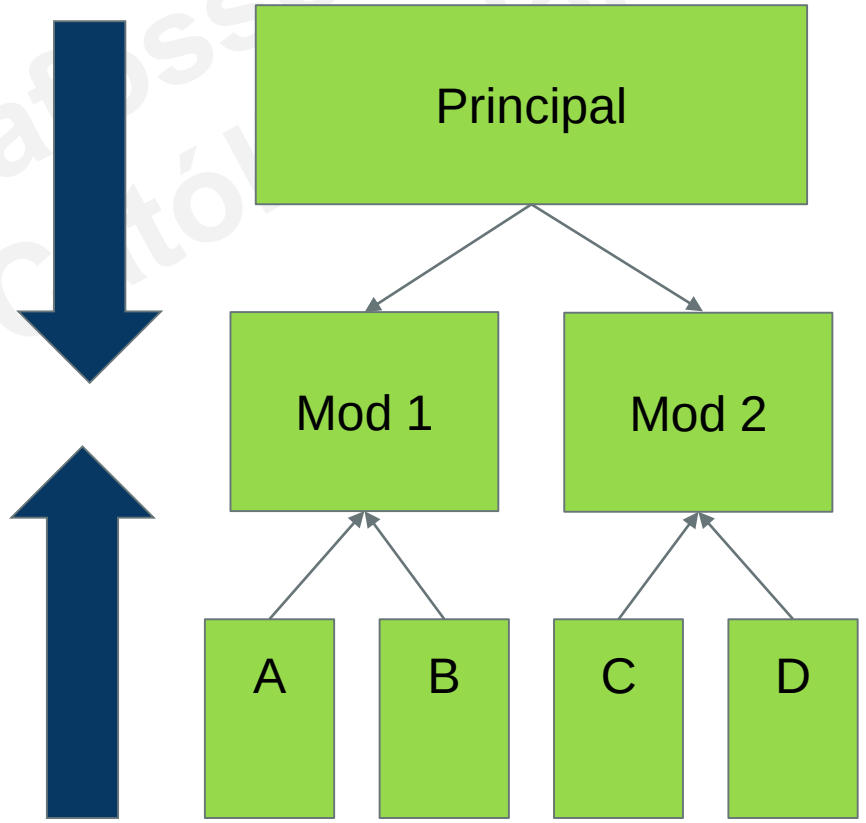


PRUEBAS DE INTEGRACIÓN

SANDWICH

Combinación de Top Down & Bottom Up.

- El sistema se ve en 3 capas: Inferior, Media, Superior.
- Las pruebas convergen en las capas medias, donde está la lógica del negocio.

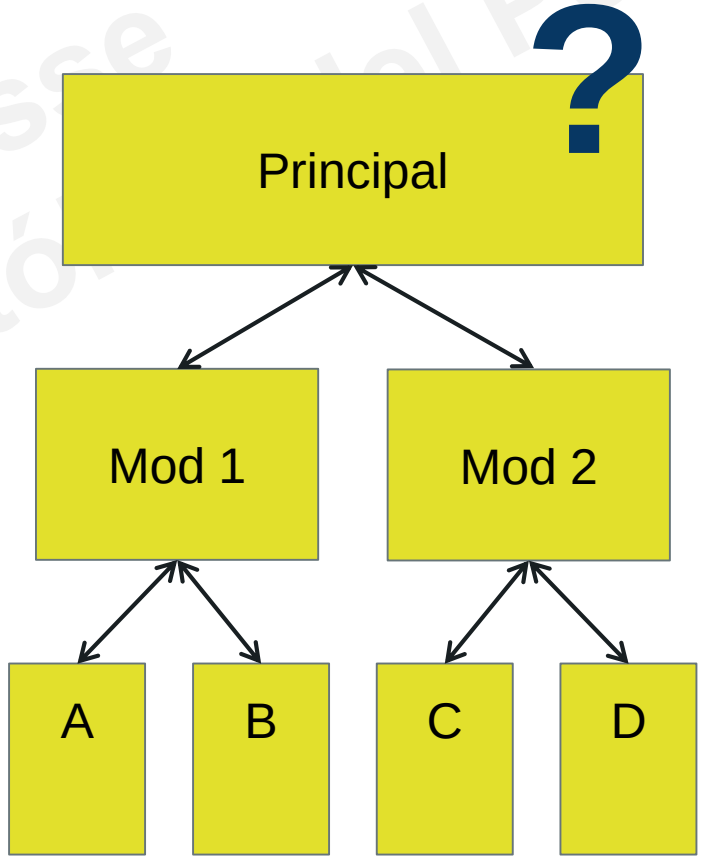


PRUEBAS DE INTEGRACIÓN

BIG BANG

Combinación de Top Down & Bottom Up.

- Se prueban los módulos individuales y luego se junta todo.
- Sólo para sistemas pequeños (económico)
- Muy riesgoso en sistemas grandes.

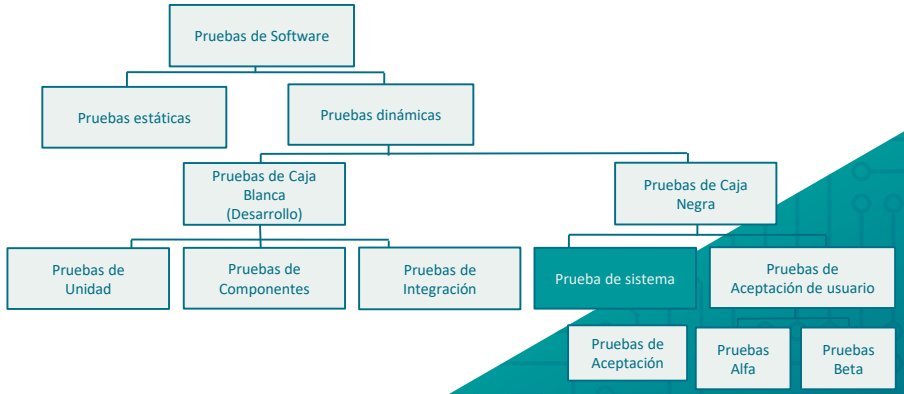


5

PRUEBAS DE SISTEMA Y CON USUARIO

PRUEBAS DE SISTEMA

- Se prueba el Sistema por Completo
- Se dan las pruebas de integración completa
- Pruebas usando los casos de uso/actividades



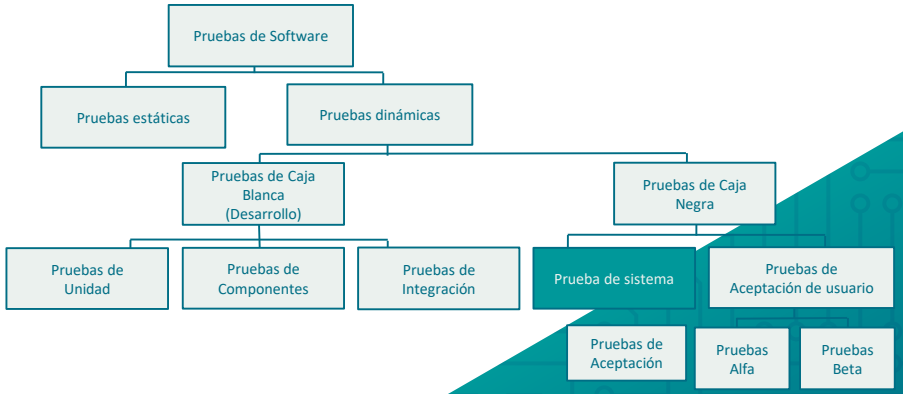
Pruebas de Sistema específicas

Pruebas que se pueden realizar para tipos de sistemas especializados o en escenarios específicos.

Dependen mucho del tipo de sistema y proyecto.

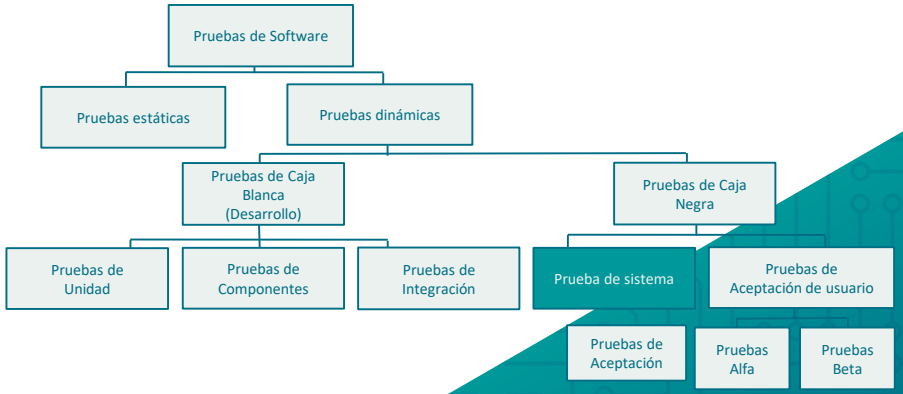
Pruebas de Sistema específicas

- **PRUEBAS BASADAS EN REQUERIMIENTOS**
  - Analizamos cada uno
  - Medibles y específicos.
- **PRUEBAS DE ESCENARIOS**
  - Escenarios según roles
  - Especificar interacciones
- **PRUEBAS DE ASEGURAMIENTO DE CALIDAD DE DATOS**
  - Datos correspondan a lo esperado.
  - Nuevos sistemas, migraciones o productos de datos



Pruebas de Sistema específicas

- **PRUEBAS DE RENDIMIENTO**
  - Prueba de esfuerzo (estrés)
    - Automatización. (Jmeter, AB, Siege)
    - Se debe aguantar más de lo máximo necesario.
  - Prueba de seguridad
    - Protocolos para probar capas (red) del sistema.
- **PRUEBA de HUMO**
  - Sólo confirmar que lo principal funciona.
  - Puede usarse en niveles menores de ser útil.

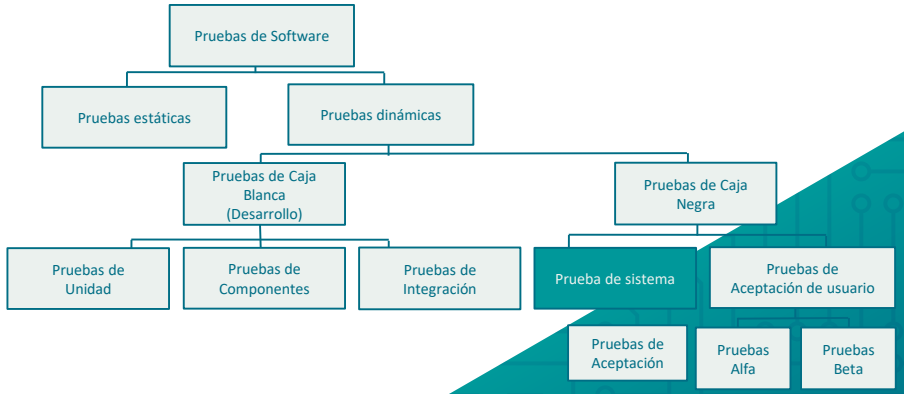


Pruebas de Sistema específicas

- **PRUEBAS DE REGRESIÓN**
  - No romper la funcionalidad anterior.
  - Puede incluir otras pruebas.

- **PRUEBA de CORDURA**

Sólo confirmar que lo principal no se ha roto por más desarrollos. (Regresionista)

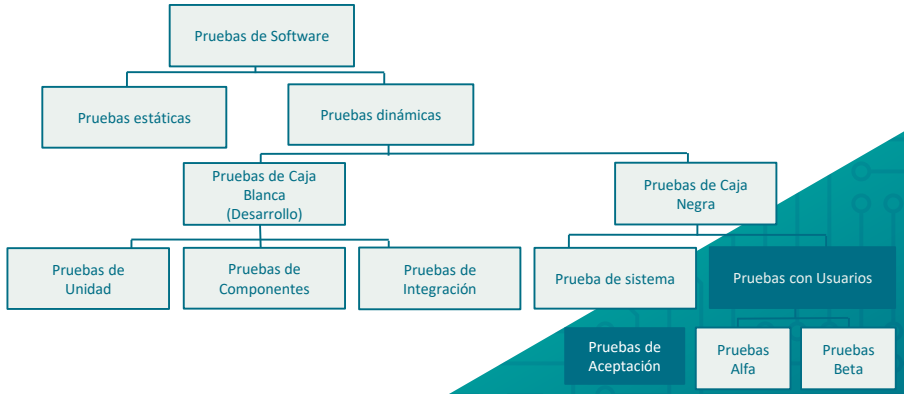


PRUEBAS CON USUARIO

PRUEBAS DE ACEPTACIÓN

- Se decide si el sistema pasa a producción
- Debe cumplir los requerimientos del cliente

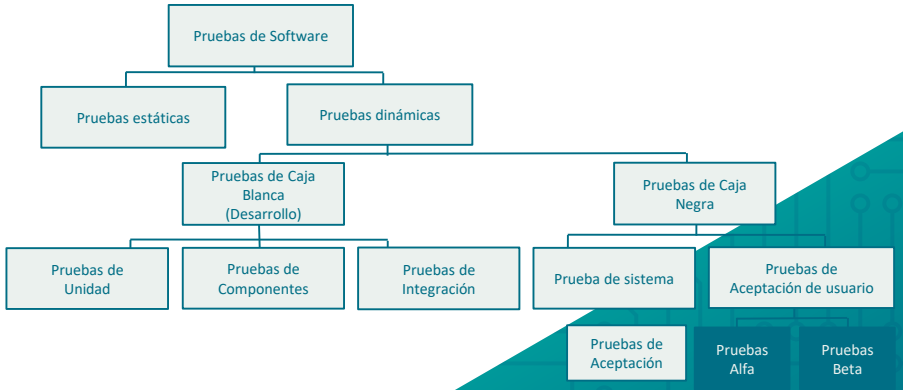
Nota: En XP (y otras metodologías ágiles),
las pruebas de aceptación son parte del desarrollo del sistema, p
or lo que no existe como paso separado.



PRUEBAS CON USUARIO

Algunos tipos especiales de pruebas con usuario

- **PRUEBAS ALFA:** Se trabaja con un sistema que aún está en desarrollo, junto con el equipo.
- **PRUEBAS BETA:** Se permite el uso libre de usuarios al sistema para que experimenten.
  - “Always in Beta”
  - Release Candidate



6

MINI-CASOS

Aplicando

Mini-caso 01

El equipo de desarrollo del Banco ha terminado de desarrollar un nuevo módulo relacionado a Fondos Mutuos. El módulo utiliza una serie de clases ya existentes (cliente, cuenta, depósitos, etc.), que fueron implementados utilizando Stubs y Mocks. Además, el gestor del módulo (capa superior) utiliza una plantilla similar a los otros módulos ya existentes. ¿Cuál consideras que es la mejor estrategia de integración?

- A: Top Down
- B: Bottom Up
- C: Sandwich
- D: Big Bang

Mini-caso 02 y 03

Se va a liberar un nuevo juego en línea la próxima semana. Todo el juego ya ha sido probado funcionalmente (con 10 usuarios en simultáneo) y todos los módulos están integrados. ¿Qué otras pruebas sugeriría hacer antes del lanzamiento?

Se actualizó el módulo de compras del ERP de una institución y el módulo de contabilidad dejó de funcionar. ¿Qué tipo de prueba no se realizó, dado que no se esperaba dicha situación?

Mini-caso 04

Francisco es uno de los representantes de la empresa GoGo, la cual ha solicitado a un equipo de desarrollo de Software la implementación de un aplicativo móvil para su tienda en línea. El día de hoy, ha recibido un correo electrónico del equipo, donde le hicieron llegar una lista de ítems (entradas y salidas esperadas) a probar, indicando que debe seguirla para realizar las pruebas de caja blanca. Además, se adjunta un aplicativo. Francisco considera que el correo podría tratarse de un error.

¿Cuál podría haber sido la situación correcta? ¿Qué podría hacer Francisco?
(Seleccionar todas las respuestas posibles) ?

- A: El correo iba dirigido a miembros del equipo para que desarrollen las pruebas de caja blanca. Francisco puede contestarles indicando que se equivocaron de remitente.
- B: El correo no tiene errores, solo que Francisco no quiere hacerse responsable.
- C: El correo se refería a pruebas de Caja Negra, que sí podrían ser realizadas por Francisco. Francisco puede contestarles para confirmar si esas pruebas debe realizarlas él.
- D: La información enviada es insuficiente para que Francisco realice las pruebas de caja blanca. Francisco debe pedir el detalle del funcionamiento interno del aplicativo, a fin de poder realizar él mismo las pruebas de Caja Blanca.

7

REFERENCIAS

BIBLIOGRAFÍA

- Sommerville, I. (2011). Software engineering (ed.). America: Pearson Education Inc.
- ISO/IEC/IEEE 29119-1:2022 Software and systems engineering https://www.iso.org/standard/81291.html
- Spiller, A. Linz, T. & Shaefer, H. (2014) Software Testing Foundations (4th Ed.) O'Reilly Media

Créditos:

- Plantilla de la presentación por SlidesCarnival
- Diseño del fondo Hero Patterns