DCC

DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN

IIC3745 - TESTING

2023 - 1º SEMESTRE

UNIDAD 2:

- Modelos de ciclo de vida del desarrollo de software
- Niveles de prueba
- Tipos de prueba

Alison Fernandez Blanco

UNIDAD 2:

- **Modelos de ciclo de vida del desarrollo de software**
- Niveles de prueba
- Tipos de prueba

3 Discusión: Ciclo de vida del software

Les encargaron desarrollar un software para un cliente.

¿En qué orden realizan las actividades?

1) D, C, A, B
2) D, A, C, B
3) A, C, B, D

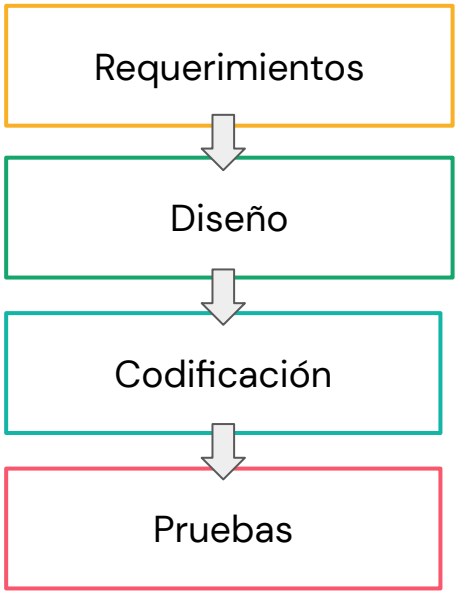
A Planear la arquitectura, el lenguaje y tecnologías que mejor se adaptaran al proyecto

B Verificar que el software se construyó según las especificaciones del cliente

C Codificar

D Recuperar toda la información que sea posible sobre los detalles y especificaciones del cliente

4 Discusión: Ciclo de vida del software



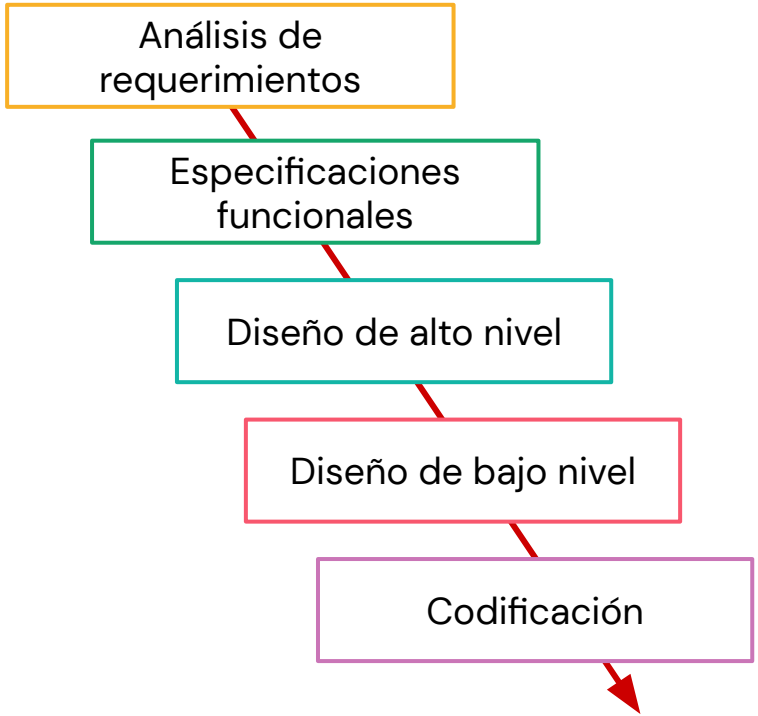
D Recuperar toda la información que sea posible sobre los detalles y especificaciones del cliente

A Planear la arquitectura, el lenguaje y tecnologías que mejor se adaptaran al proyecto

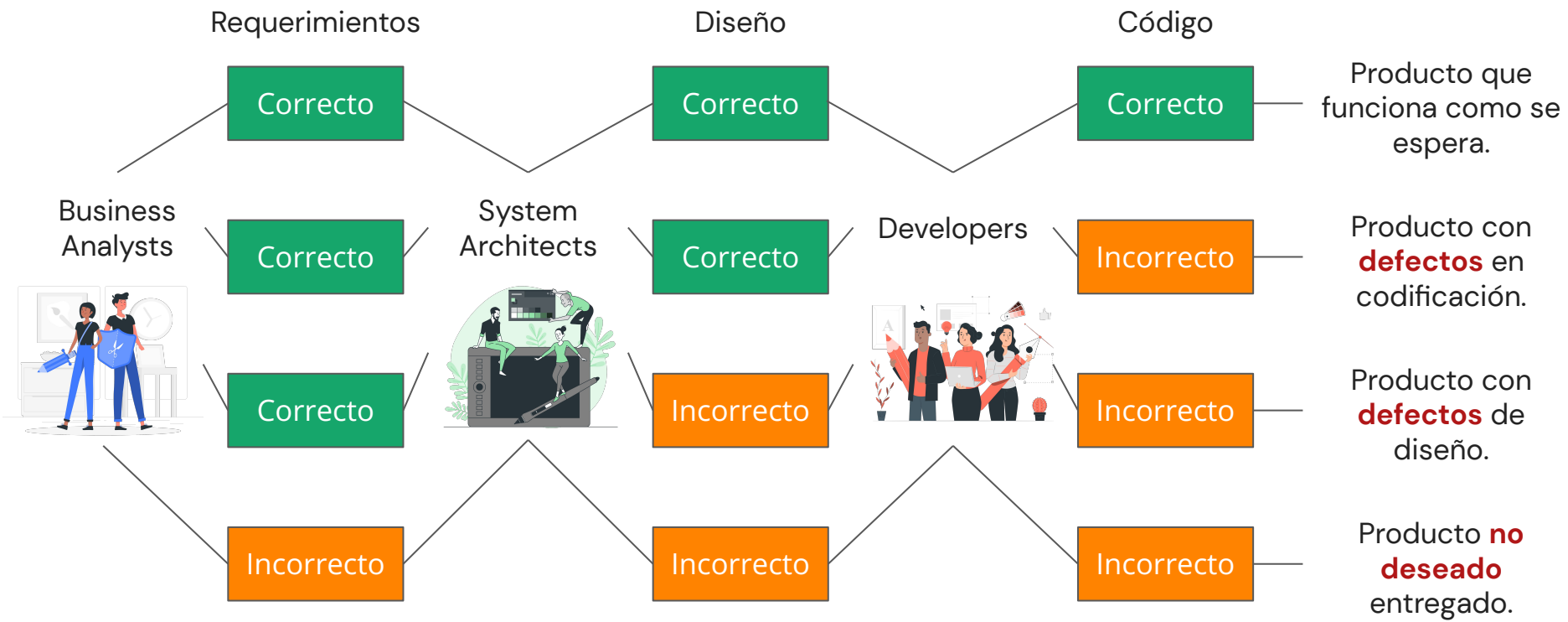
C Codificar

B Verificar que el software se construyó según las especificaciones del cliente

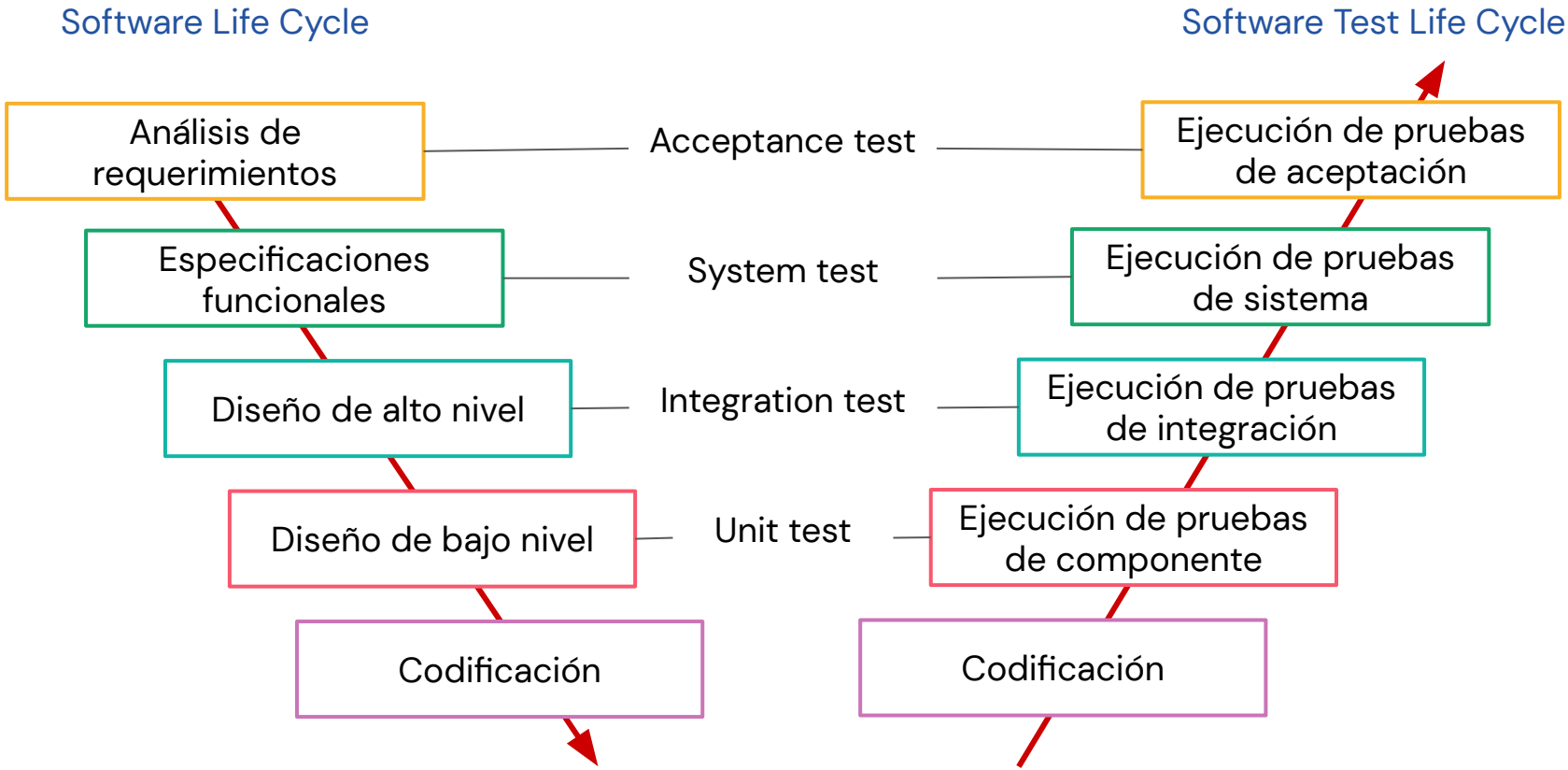
5 Ciclo de vida del Software: Waterfall Method



6 ¿Cuando se introducen los defectos?



7 V-model


Software Life Cycle
Software Test Life Cycle
Análisis de requerimientos
Acceptance test
Ejecución de pruebas de aceptación
Especificaciones funcionales
System test
Ejecución de pruebas de sistema
Diseño de alto nivel
Integration test
Ejecución de pruebas de integración
Diseño de bajo nivel
Unit test
Ejecución de pruebas de componente
Codificación
Codificación

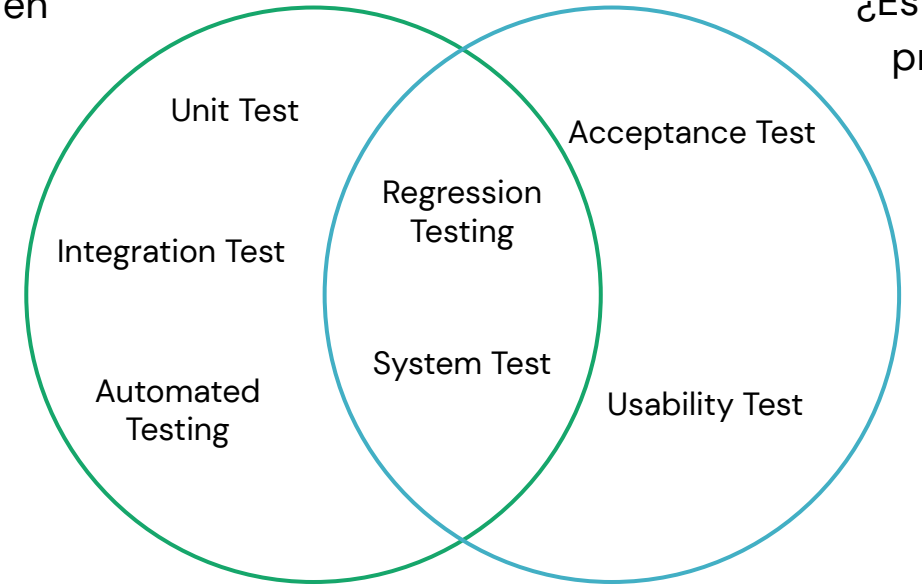
8 Características de un buen Testing

- Por cada actividad de desarrollo existe una actividad de Testing.
- Cada nivel de Testing tiene objetivos específicos de ese nivel.
- El análisis y diseño de pruebas por cada nivel debe empezar durante su respectiva actividad de desarrollo.
- Los testers (i) participan en discusiones para definir y redefinir requerimientos y (ii) revisan los productos tan pronto como un borrador está disponible.

9 Características de un buen Testing

Verificación

¿Estoy construyendo bien el producto?

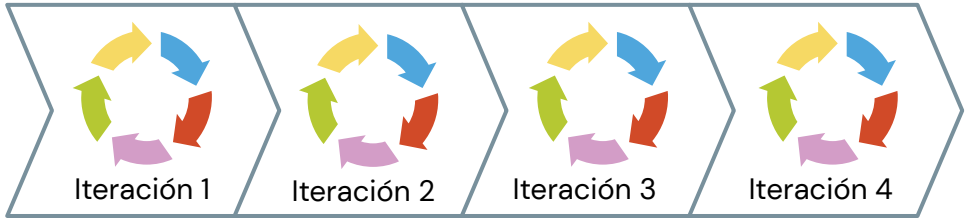


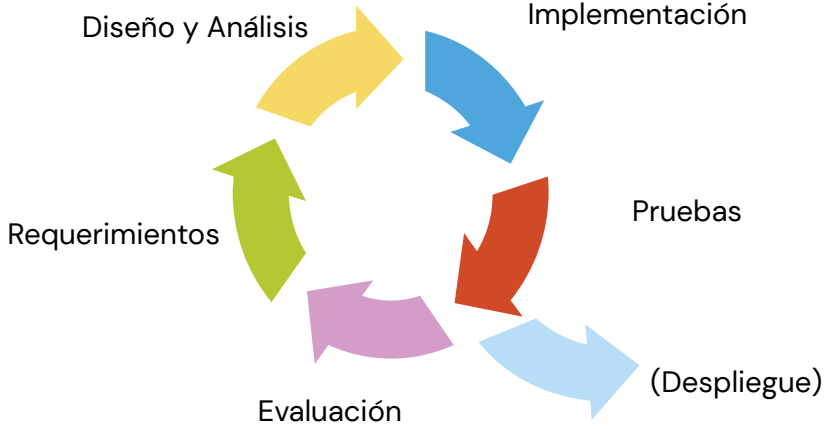
Validación

¿Estoy construyendo el producto correcto?

10 Ciclo de vida de software: Iterativo e incremental

"Build a little, test a little"



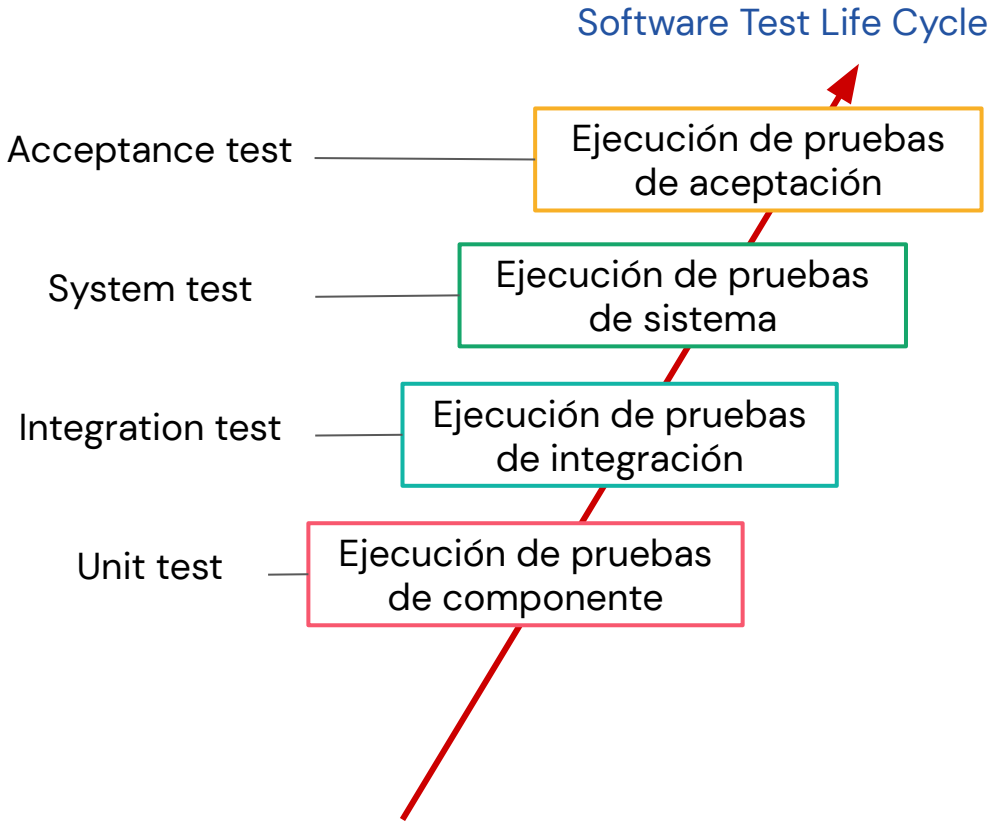


Iteración en detalle

UNIDAD 2:

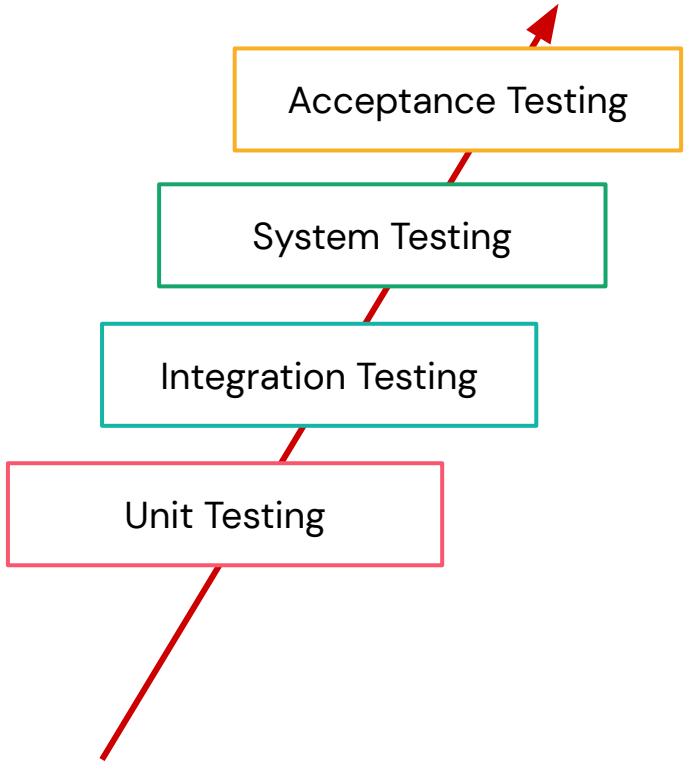
- Modelos de ciclo de vida del desarrollo de software
- **Niveles de prueba**
- Tipos de prueba

12 Niveles de prueba



13 Atributos de niveles

- Objetivos específicos
- Objetos a probar (test object: lo que se probara)
- Bases de las pruebas (test basis)
- Defectos y fallas típicas a encontrar
- Enfoques y responsabilidad



14 Unit/Component Testing

Objetivos:

- Verificar si el comportamiento (funcional o no funcional) del **componente** es el esperado.
- Encontrar defectos en el **componente.**
- Reducir el riesgo.
- Evitar que los defectos escalen a niveles superiores.

Objetos a probar:

- Componentes, unidades o módulos.
- Código y estructuras de datos.
- Clases.
- Módulos de la base de datos.

15 Unit/Component Testing

Bases de las pruebas:	❖ Diseño detallado y código. ❖ Modelos de datos. ❖ Especificaciones del componente.
Defectos y fallas típicas:	❖ Funcionalidad incorrecta. ❖ Problemas con el flujo de datos. ❖ Lógica incorrecta.
Enfoques y responsabilidades:	❖ Usualmente realizado por **programadores**. ❖ TDD o pruebas después de la implementación, etc.

16 Integration Testing

Se enfoca en la interacción entre componentes o sistemas. Hay dos niveles diferentes de pruebas de integración:

- **Component Integration Testing**: Prueba la interacción entre componentes de software y normalmente es realizado después del Unit Testing.
- **System Integration Testing.** Prueba la interacción entre diferentes sistemas, paquetes o microservicios y puede ser realizado después del System Testing.

17 Integration Testing

Objetivos:

- Verificar si el comportamiento (funcional o no funcional) de **interfaces** es el esperado.
- Encontrar defectos en **interfaces.**
- Reducir el riesgo.
- Evitar que los defectos escalen a niveles superiores.

Objetos a probar:

- Subsistemas.
- Bases de datos.
- Interfaces, APIs.
- Microservicios.

18 Integration Testing

Bases de las pruebas:

- Diseño del sistema.
- Diagramas de secuencia.
- Arquitectura (componente o sistema).
- Interfaces y protocolos de comunicación.
- Workflows.

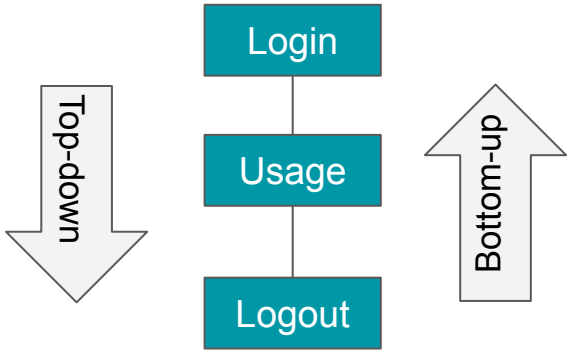
Defectos y fallas típicas:

- Problemas con los datos (datos incorrectos, faltantes, etc).
- Discrepancia entre interfaces.
- Fallas de comunicación entre componentes.
- Suposiciones incorrectas.

Integration Testing

Enfoques y responsabilidades:

- Usualmente realizado por **programadores** (component integration) y por **testers** (system integration).
- **Big bang:** Todos los componentes son integrados en un solo paso resultando en un sistema completo.
- **Top-down:** Siguiendo el flujo de control o la arquitectura (empezando por la interfaz). Los componentes son sustituidos por **stubs** (componente pasivo que es llamado por otros).
- **Bottom-up:** Siguiendo el flujo de control de abajo hacia arriba, normalmente sustituyendo los componentes por **drivers** (componente activo que llama a otros).



20 System Testing

Objetivos:

- Verificar si el comportamiento (funcional y no funcional) del **sistema** es el esperado según diseño y especificaciones.
- Encontrar defectos en el **sistema.**
- Validar que el sistema esté completo y que funcione como se espera.
- Reducir el riesgo.
- Evitar que los defectos escalen a niveles superiores.

Objetos a probar:

- Aplicaciones.
- Hardware/software.
- Sistema operativo, configuración y datos del sistema.

21 System Testing

Bases de las pruebas:

- Requerimientos funcionales y no funcionales
- Reporte de análisis de riesgo
- Casos de uso, modelos de comportamiento de sistema
- Manuales de usuario y sistema, etc

Defectos y fallas típicas:

- Comportamiento inesperado e incorrecto
- No funciona en el entorno de producción
- Fallas en los manuales o documentación
- Datos incorrectos o fallas en el flujo de control.

System Testing

Enfoques y responsabilidades:

- Usualmente realizado en el entorno de pruebas por **testers independientes**.
- Técnicas para investigar requerimientos funcionales y no funcionales: tabla de decisiones, etc.

23 Acceptance Testing

Objetivos:

- Establecer **confianza en la calidad** del sistema.
- Verificar si el comportamiento (funcional y no funcional) del **sistema** es el esperado según diseño y especificaciones.
- Validar que el **sistema** esté completo y que funcione como se espera.

Objetos a ser probados:

- El sistema a ser probado
- La configuración del sistema y los datos
- Procesos de mantenimiento y operacionales
- Formularios, reportes, etc

24 Acceptance Testing

Bases de las pruebas:	❖ Proceso de negocio ❖ Requerimientos de usuario, negocio y sistema ❖ Regulaciones, estándares y contratos ❖ Documentación, etc.
Defectos y fallas típicas:	❖ El sistema no cumple con las necesidades del usuario. ❖ Las reglas del negocio no son correctas. ❖ Problemas contractuales o regulatorios. ❖ Falla no funcionales (performance, seguridad), etc.

25 Acceptance Testing

Formas	Objetivo	Encargados y entorno
User Acceptance Testing	Validar si el sistema es apto para ser usado.	Usuario objetivo en un entorno real o simulado.
Operational Acceptance Testing	Validar si el sistema cumple con los requerimientos.	Usuarios y administradores del sistema.
Contractual Acceptance Testing	Comprobar que el software hecho a medida sigue los criterios establecidos en el contrato.	Usuarios o testers independientes.
Alpha Testing	Recolectar de retroalimentación y problemas (por observación).	Usuarios objetivo y miembros de empresa (no desarrolladores) en el entorno de la compañía.
Beta Testing	Recolectar de retroalimentación y problemas (por análisis de registros).	Usuarios objetivo en un entorno externo (fuera de la compañía).

UNIDAD 2:

- Modelos de ciclo de vida del desarrollo de software
- Niveles de prueba
- **Tipos de prueba**

27 Tipos de prueba

Los tipos de prueba se dividen según su objetivo:

- **Functional testing.** Evalúa la calidad funcional del software: completitud, correctitud, etc.
- **Non-functional testing.** Evalúa la calidad no funcional del software: confiabilidad, seguridad, usabilidad, etc.
- **White box testing.** Evalúa la arquitectura y estructura del software
- **Change related testing.** Evalúa el efecto de los cambios en el software.
  - **Confirmation testing.** Confirma que los defectos se repararon.
  - **Regression testing.** Busca si los cambios no causaron efectos secundarios.

Non-functional testing

Características a ser evaluadas:

- **Reliability**, que a su vez se pueden dividir en: robusto, tolerante a fallas y recuperabilidad.
- **Usability**, que a su vez se puede dividir en: entendibilidad, aprendizaje, operatividad, atractivo y conformidad.
- **Efficiency**, el cual es dividido en: comportamiento (performance), uso de recursos.
- **Maintainability**, que consiste en: facilidad para analizar, cambiar, estabilidad y fácil de probar.
- **Portability**, que consiste en: adaptabilidad, instalabilidad, coexistencia y fácil de reemplazar.

Tipos de pruebas

Existen más tipos de pruebas, pero en este curso nos centraremos en los tipos de pruebas mencionados anteriormente.

¿Consultas?