Unidad 1.3 – Pruebas de software en Python

Módulo: Puesta en Producción Segura

Pruebas - Introducción

- **Definición**: Pruebas de software son el proceso de verificar y validar que un software funciona como se espera.
- **Objetivo**: Asegurar la calidad y fiabilidad del software.
- **Personal**: Desarrolladores, o mejor, personal especializado y ajeno al proyecto.

Importancia de las pruebas

- Detección Temprana de Errores: Ahorra tiempo y recursos.
- Mejora de la Calidad del Software: Garantiza un software más estable y confiable.
- Mantenimiento: Facilita la identificación y corrección de errores en el futuro.

Verificación vs Validación

- **Verificación**: Comprueba si el producto está siendo construido correctamente. Ejemplo: revisión de código.
- **Validación**: Comprueba si el producto construido cumple con los requisitos y expectativas del usuario. Ejemplo: pruebas funcionales.

Etapas o Niveles de Pruebas

1. Pruebas Unitarias: Verifican componentes individuales.

2. Pruebas de Integración: Verifican interacciones entre los componentes.

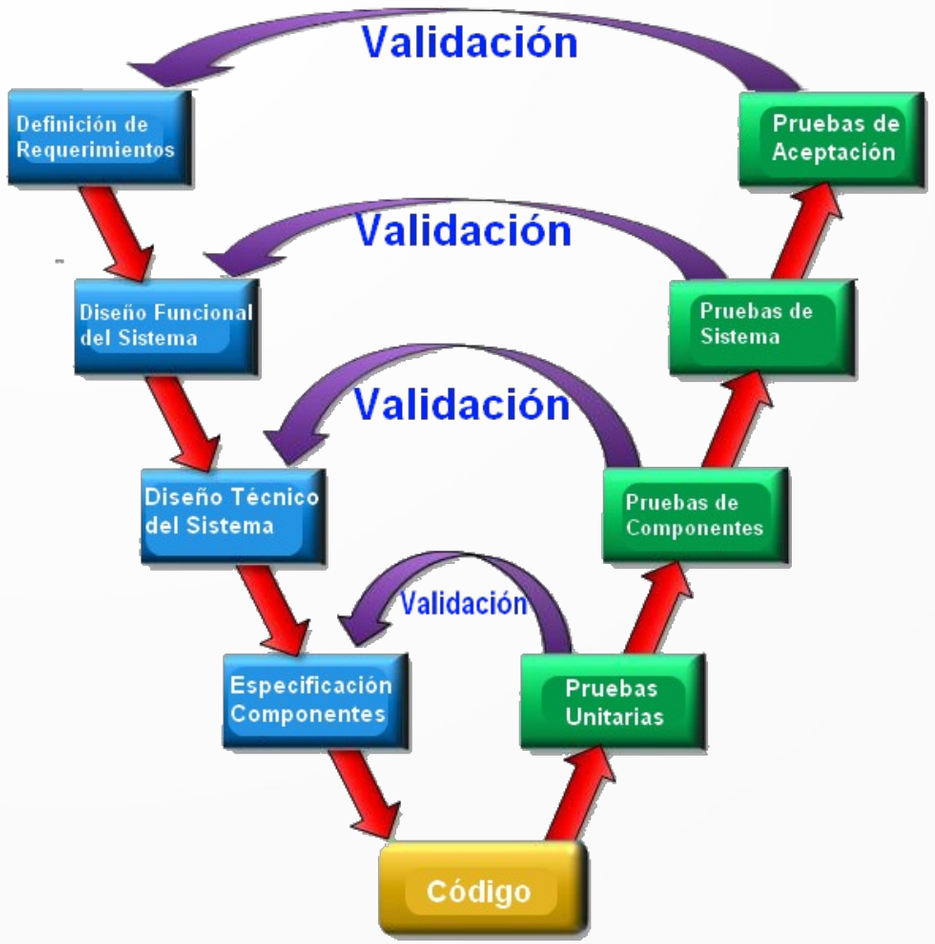
3. Pruebas del Sistema: Comprueba el funcionamiento del sistema completo.

4. Pruebas de aceptación: El cliente comprueba si satisface sus necesidades.

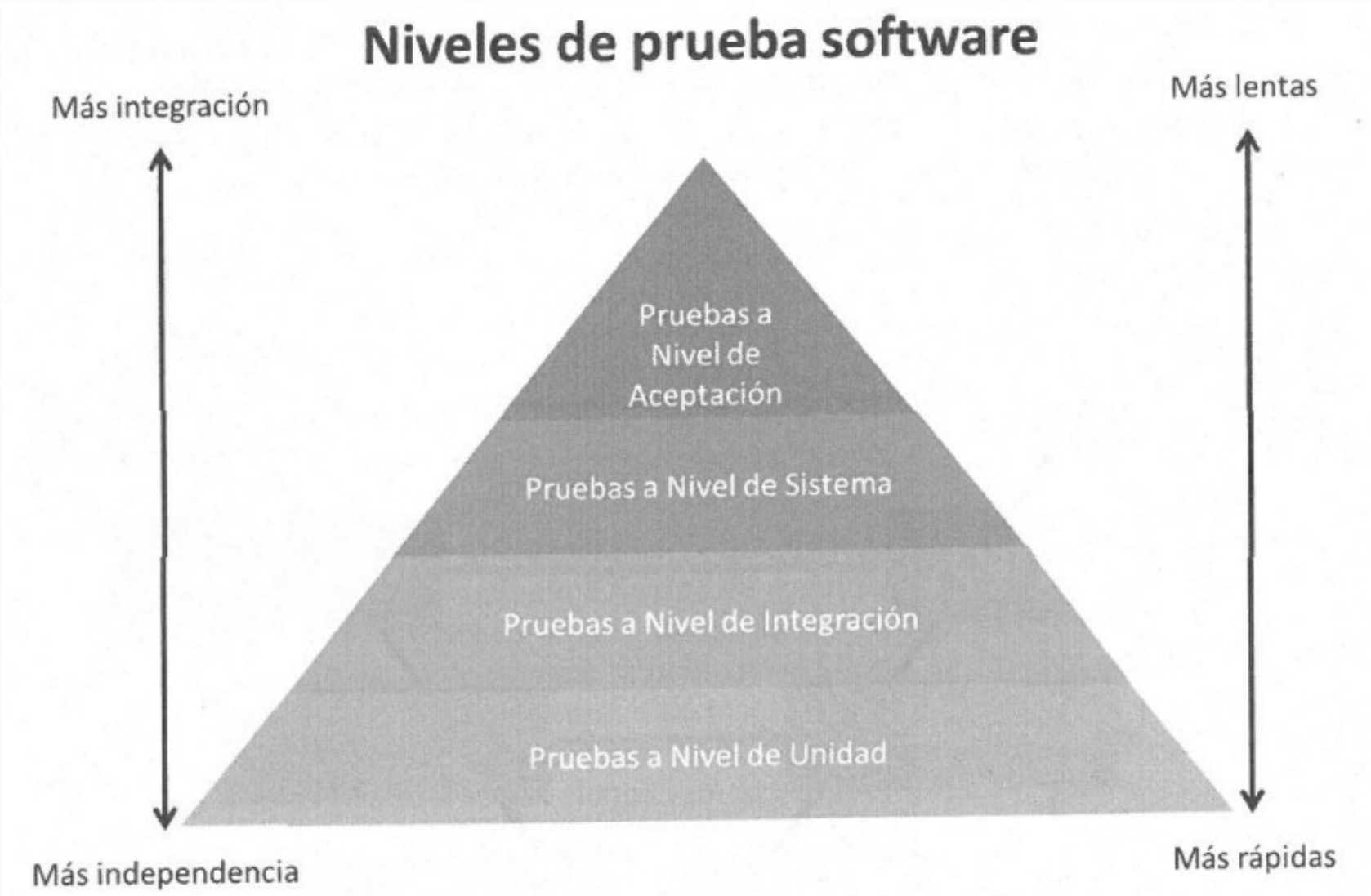
Etapas o Niveles de Pruebas

Modelo en V:

Indica cómo se debe de llevar a cabo la verificación y validación.



Etapas o Niveles de Pruebas



Pruebas Unitarias

- **Definición**: Verifican componentes individuales en el nivel más bajo de la aplicación.
- **Objetivo**: Asegurar que cada unidad de código funcione correctamente de manera aislada.
- **Ejemplos:** Junit, PHPUnit, Nunit,Mocha -Chai

Herramientas de Pruebas en Python

- **Unittest**: Biblioteca estándar de Python para pruebas unitarias.
- **PyTest**: Framework de pruebas robusto y flexible.
- **Nose**: Herramienta de pruebas que extiende unittest.

Pruebas Unitarias con Unittest

- Ejemplo de Código:

Python Copiar

```python
import unittest

def suma(a, b):
    return a + b

class TestSuma(unittest.TestCase):
    def test_suma_positiva(self):
        self.assertEqual(suma(1, 2), 3)

    def test_suma_negativa(self):
        self.assertEqual(suma(-1, -2), -3)

if __name__ == '__main__':
    unittest.main()
```

Pruebas Unitarias con PyTest

- Ejemplo de Código:

Python Copiar

```python
import pytest

def multiplica(a, b):
    return a * b

def test_multiplica():
    assert multiplica(2, 3) == 6
    assert multiplica(-1, 5) == -5
```

Pruebas de Integración

- **Definición**: Verifican interacciones entre componentes y módulos del sistema.
- **Objetivo**: Asegurar que los componentes integrados funcionen correctamente en conjunto.
- **Ejemplo**: Pruebas de integración de APIs, pruebas de interacción entre módulos de software.

Metodologías de Pruebas de integración

Pruebas no incrementales: Se integran todos los componentes y se prueba el sistema de una vez.

- **Pruebas incrementales:** Se realiza integración gradual.
- **Integración desdendente:** Es incremental y se realiza desde el componente de más alto nivel.
- I**ntegración asdendente:** Es incremental y se realiza desde el componente de más bajo nivel.
- **Integración Mixta:** Conjuntamente varias estrategias.

Pruebas del Sistema

- **Definición**: Verifican el sistema completo.
- **Objetivo**: Asegurar que el sistema cumpla con los requisitos especificados y funcione como un todo.
- **Ejemplo**: Pruebas de extremo a extremo (end-to-end), pruebas funcionales completas..

Pruebas del Sistema

- Pruebas de integración del sistema completo que comprueba el funcionamiento global del Sistema.
- No es necesario realizarlas por alguien que conozca el sistema.
- Pruebas desde el punto de vista del usuario.
- Permiten validar arquitectura y requisitos.

Pruebas de Aceptación

- **Definición**: Validan que el sistema cumple con los requisitos y expectativas del usuario final.
- **Objetivo**: Obtener la aprobación del cliente o usuario final antes de la implementación.
- **Ejemplo**: Pruebas de usuario, pruebas de aceptación del cliente.

Tipos de pruebas de Aceptación

- **Pruebas alpha:** En el lugar del desarrollo por un cliente.
- **Pruebas beta:** Por el cliente en entornos de trabajo.

Tipos de Pruebas

1. **Pruebas Funcionales**: Verifican que el sistema funciona como se espera.

2. **Pruebas no funcionales:** Controlar calidad de los sistemas implementados.

3. **Pruebas de Regresión**: Aseguran que cambios no introduzcan nuevos errores.

Pruebas Funcionales

- **Definición**: Verifican que el sistema funcione de acuerdo con las especificaciones y requisitos funcionales.
- **Objetivo**: Asegurar que cada función del software opere correctamente.
- **Ejemplo**: Pruebas de casos de uso, exploratorias, pruebas de regresión, compatibilidad de entorno, de humo, de sanidad, de mono...

Pruebas No Funcionales

- **Definición**: Verifican aspectos no funcionales del sistema como rendimiento, seguridad, usabilidad.
- **Objetivo**: Asegurar que el sistema cumpla con los requisitos de rendimiento y otros criterios no funcionales.

Ejemplo: Pruebas de carga, de estrés, de estabilidad, Pruebas de rendimiento, Recuperación, instalación, estructurales, configuración y usabilidad.

Técnicas de prueba

- **Enfoque funcional o de caja negra.** Estudia especificaciones del producto.
- **Enfoque estructural o de caja blanca.** Analiza la estructura del componente
- **Enfoque aleatorio.** Utiliza modelos (pueden ser estadísticos) sobre las posibles entradas.

Pruebas de Caja Negra

- **Definición**: Pruebas en las que el tester no conoce la estructura interna del sistema y se enfoca en la funcionalidad externa.
- **Objetivo**: Verificar que las entradas generen las salidas esperadas sin conocer el código interno.
- **Ejemplo**: Pruebas basadas en los requisitos y especificaciones, pruebas funcionales.

Técnicas de pruebas de Caja Negra

- **Técnica de las Clases o Particiones de equivalencia:** Identificamos los tipos de valores diferentes y probamos uno de cada.
- **Técnica de análisis de valores límite:** Pruebas para los extremos de las diferentes clases de equivalencia.

Pruebas de Caja Blanca

- **Definición**: Pruebas en las que el tester conoce la estructura interna del sistema y se enfoca en el código y lógica interna.
- **Objetivo**: Verificar el flujo de control y los caminos lógicos dentro del código.
- **Ejemplo**: Pruebas de cobertura de código, pruebas de rutas lógicas, revisiones de código.

Pruebas de Caja Blanca

- **Definición**: Pruebas en las que el tester conoce la estructura interna del sistema y se enfoca en el código y lógica interna.
- **Objetivo**: Verificar el flujo de control y los caminos lógicos dentro del código.
- **Ejemplo**: Pruebas de cobertura de código, pruebas de rutas lógicas, revisiones de código.

Técnicas de pruebas de Caja Blanca

- **Cobertura de sentencias**
- **Cobertura de Decisión.**
- **Cobertura de condiciones.**
- **Cobertura de Decisión/condición.**
- **Cobertura de Condición Múltiple.**
- **Cobertura de Caminos:**

Cobertura de Caminos.

- **Técnica del camino básico.**

  Halla la complejidad lógica de un diseño que sirve para definición de un conjunto básico.

- Se dibuja el grafo de flujo asociado.
- Se calcula la complejidad ciclomática del grafo
- Se determina un conjunto básico de caminos independientes.
- Se preparan casos de prueba para cada uno de los caminos.

Mejores Prácticas

- Escribe Pruebas Claras y Simples: Facilita la comprensión y el mantenimiento.
- Cubre Casos Límites: Asegura que todos los posibles escenarios están cubiertos.
- Automatiza las Pruebas: Utiliza CI/CD para ejecutar pruebas automáticamente

Conclusión

- Resumen: La importancia de las pruebas de software y cómo Python ofrece diversas herramientas y frameworks para facilitar el proceso.