UT3 - DISEÑO Y REALIZACIÓN DE PRUEBAS – PROCEDIMIENTOS DE PRUEBAS Y TÉCNICAS DE DISEÑO DE CASOS DE PRUEBA – PARTE II

RESULTADOS DE APRENDIZAJE ASOCIADOS
3.- Verifica el funcionamiento de programas diseñando y realizando pruebas.
CRITERIOS DE EVALUACIÓN
a) Se han identificado los diferentes tipos de pruebas.
b) Se han definido casos de prueba
c) Se han identificado las herramientas de depuración y prueba de aplicaciones ofrecidas por el entorno de desarrollo.
d) Se han utilizado herramientas de depuración para definir puntos de ruptura y seguimiento.
e) Se han utilizado las herramientas de depuración para examinar y modificar el comportamiento de un programa en tiempo de ejecución.
f) Se han efectuado pruebas unitarias de clases y funciones.
g) Se han implementado pruebas automáticas.
h) Se han documentado las incidencias detectadas.

UT3 - DISEÑO Y REALIZACIÓN DE PRUEBAS – PROCEDIMIENTOS DE PRUEBAS Y TÉCNICAS DE DISEÑO DE CASOS DE PRUEBA – PARTE II

Índice de contenido

1.- Contenidos

En esta sección, abordaremos en profundidad la **planificación de pruebas de software**, destacando la relevancia de mantener un control meticuloso y un orden sistemático en esta fase crítica. Profundizaremos en la importancia de las pruebas unitarias, de integración, de aceptación o validación, y la automatización de pruebas, subrayando cómo cada una contribuye a la calidad y eficiencia del software. Además, se incluirá una guía detallada sobre la documentación necesaria para cada tipo de prueba, asegurando su adecuada implementación y seguimiento.

2.- Introducción

La planificación de las pruebas es un punto **importante** en la toma de decisiones de un proyecto.

> **Qué tipo de pruebas y cuándo van a realizarse son preguntas que hay que tener en cuenta desde el principio**

Cuando nos enfrentamos a **proyectos de gran magnitud**, es habitual llevar a cabo las pruebas que se describirán a continuación. Por otro lado, en **proyectos de menor escala y presupuesto limitado**, recae en los responsables técnicos del desarrollo la tarea de definir una estrategia de pruebas apropiada.

2.1.- Estrategias de pruebas del software

Las estrategias de pruebas de software son métodos sistemáticos para verificar y validar que el software cumple con los requisitos especificados. Las principales estrategias incluyen:

- **Pruebas Unitarias:** Verificación de componentes individuales o módulos del software.
- **Pruebas de Integración:** Evaluación de la interacción entre módulos o componentes combinados.

- **Pruebas de Sistema:** Verificación del software completo y sus interfaces externas.
- **Pruebas de Aceptación:** Asegurar que el software cumple con las expectativas y requisitos del usuario final.

Estas estrategias pueden ser aplicadas en diferentes modelos de desarrollo:

- Modelo en cascada
- Modelo ágil
- Modelo en espiral
- Etc

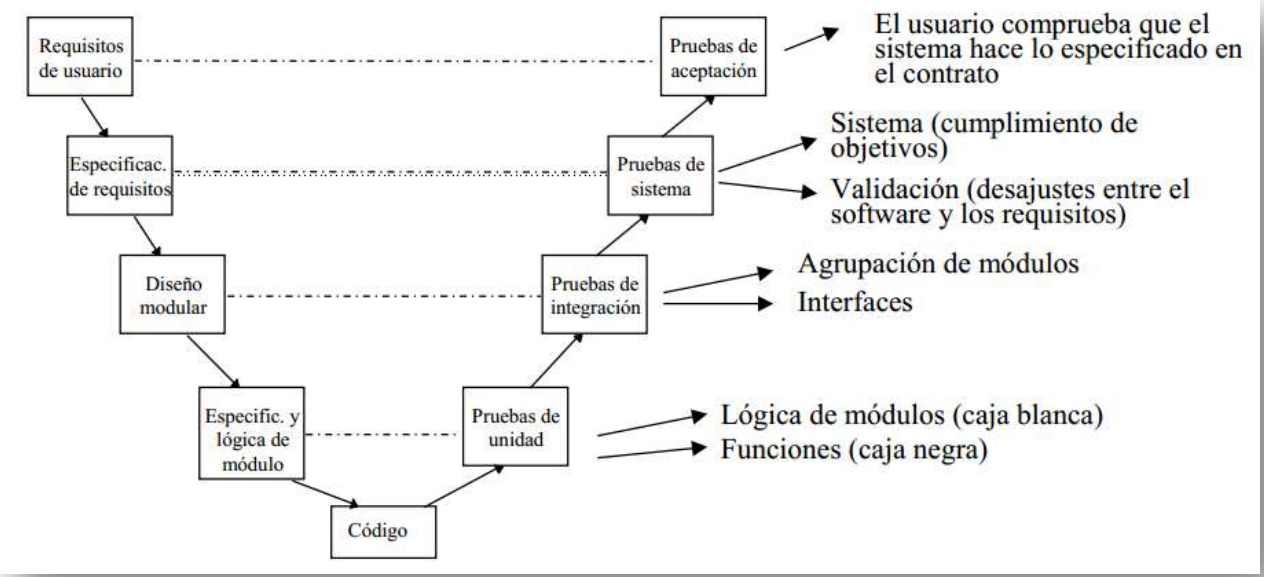
Estas pruebas suelen ser complementadas con pruebas automáticas y manuales según sea necesario.

2.1.1- Modelo V

El modelo V generalizado en las pruebas de software se estructura en cuatro niveles de pruebas, correspondientes a cuatro niveles de desarrollo:

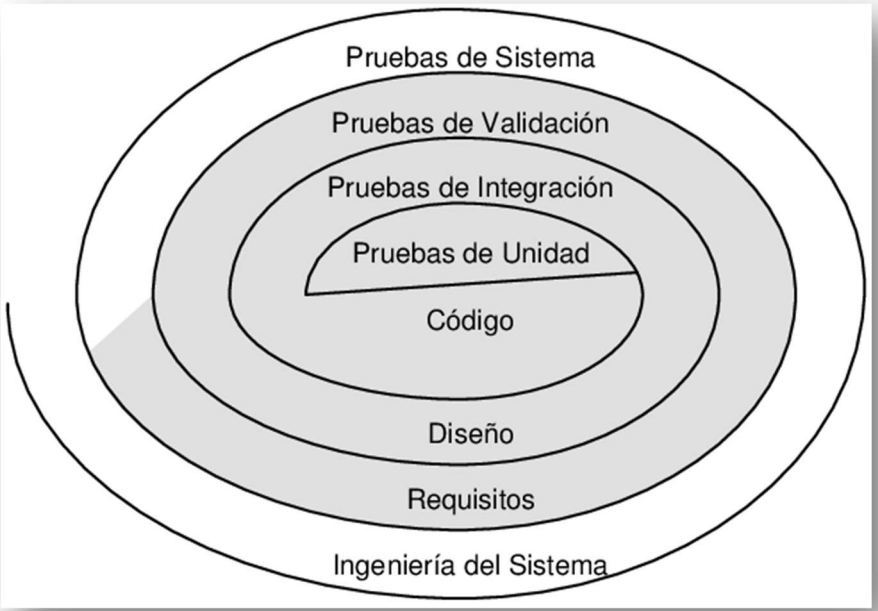
- Pruebas de componentes (unidades)
- Pruebas de integración
- Pruebas de sistema
- Pruebas de aceptación.

Este modelo puede variar según el proyecto, pudiendo incluir, por ejemplo, pruebas de integración de componentes y de sistemas en diferentes etapas del desarrollo del software.



2.1.2- Modelo en espiral

La estrategia en espiral es un modelo iterativo de desarrollo de software que combina la planificación y el prototipado. Este modelo enfatiza la evaluación y gestión de riesgos en cada etapa, permitiendo adaptaciones continuas y refinamientos basados en la retroalimentación del cliente y el análisis de riesgos. Es especialmente útil en proyectos grandes y complejos con requisitos inciertos o cambiantes.



3.- Pruebas unitarias

- **Objetivo de las Pruebas Unitarias**
  - Realizar verificaciones durante las primeras etapas de diseño y desarrollo.
- **Importancia del Tiempo**
  - Evitar demoras en la realización de estas pruebas.

Las demoras pueden llevar a la acumulación de errores y complicar el diagnóstico y la localización de fallos

- **Enfoque en la Programación Orientada a Objetos (POO)**
  - Realizar pruebas a nivel de objeto primero.
  - Luego, proceder a nivel de paquete o librería.
  - Importancia de no probar paquetes de forma aislada sin antes haber realizado pruebas a nivel de objeto individual.

Ejemplo:

```
/**
 * Clase Calculator que proporciona operaciones matemáticas básicas.
 */
public class Calculator {

    /**
     * Suma dos números enteros.
     * @param a Primer número entero.
     * @param b Segundo número entero.
     * @return La suma de a y b.
     */
    public int add(int a, int b) {
        return a + b;
    }

    /**
```

```
    * Resta dos números enteros.
    * @param a Primer número entero.
    * @param b Segundo número entero.
    * @return La diferencia entre a y b.
    */
   public int subtract(int a, int b) {
      return a - b;
   }

   // Métodos para multiplicar y dividir con comentarios
Javadoc similares...
}

/**
 * Clase de prueba para la clase Calculator.
 * Utiliza JUnit para realizar pruebas unitarias de los métodos
de Calculator.
 */
public class CalculatorTest {

   /**
    * Prueba el método add de la clase Calculator.
    * Verifica si la suma de dos números es correcta.
    */
   @Test
   public void testAdd() {
      Calculator calculator = new Calculator();
      assertEquals(5, calculator.add(2, 3)); // Verifica que 2 + 3
es igual a 5
   }

   /**
    * Prueba el método subtract de la clase Calculator.
    * Verifica si la resta de dos números es correcta.
```

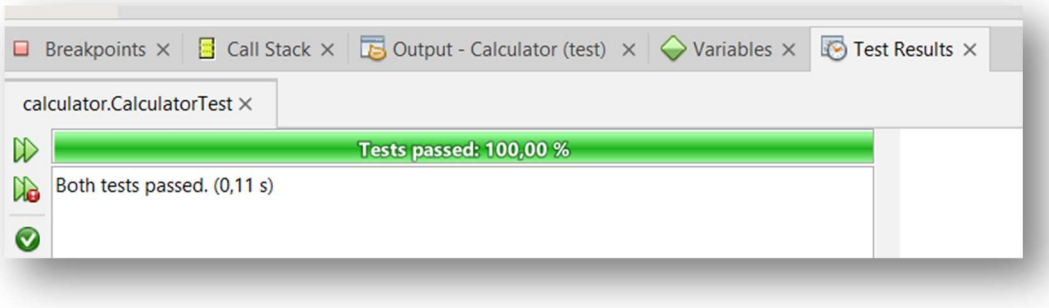
```
    */
   @Test
   public void testSubtract() {
      Calculator calculator = new Calculator();
      assertEquals(1, calculator.subtract(4, 3)); // Verifica que 4 -
3 es igual a 1
   }

   // Pruebas para los métodos de multiplicar y dividir con
comentarios Javadoc similares...
}
```

Trabajo opcional:

¿Cómo haríamos funcionar el test anterior tanto en *Netbeans* como en *Eclipse*?

TAREA: Realizar minitutorial con los pasos a seguir para que todo funcione correctamente.

RESULTADO:
Así debe mostrarse:



Hay recompensa. Hablar con el profesor.

Pista:https://github.com/junit-team/junit4/wiki/Download-and-Install

4.- Pruebas de integración

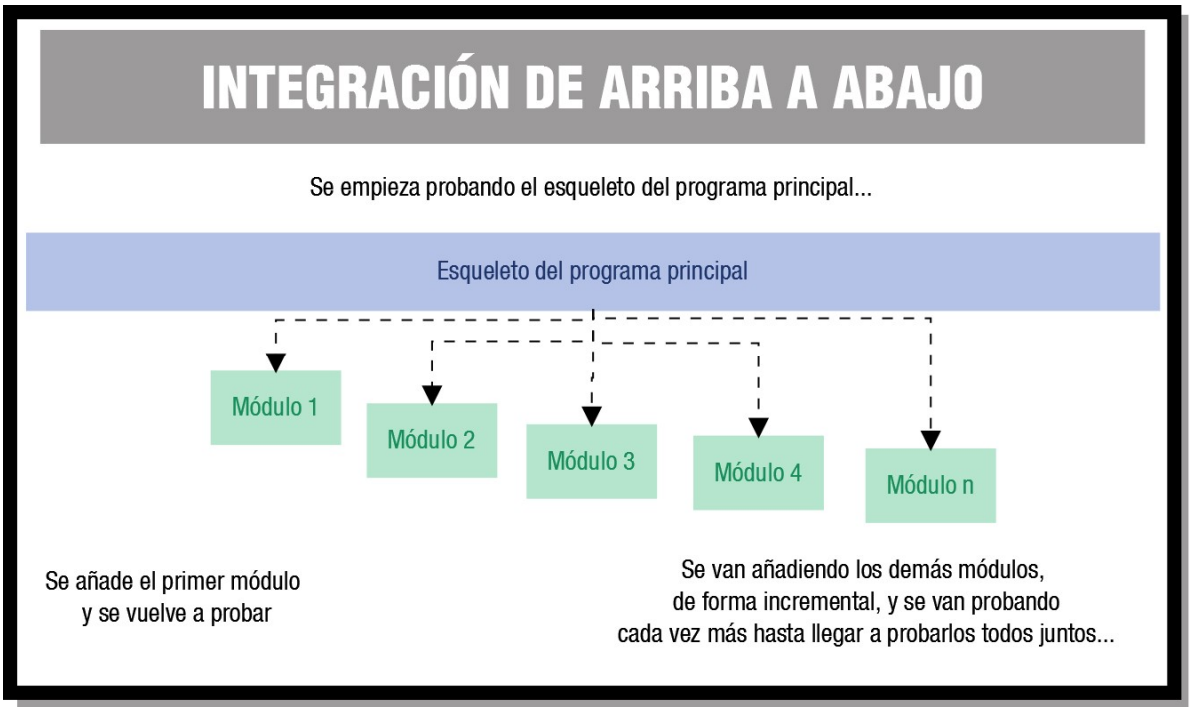
- **Objetivo de las Pruebas de Integración:**
  - Integrar y probar módulos después de haber probado los componentes individuales.
  -
- **Momentos Clave para las Pruebas de Integración:**
  - Al final de la fase de diseño:
    - Realizar pruebas para confirmar la factibilidad y eficiencia del diseño.
  - Al final de la fase de codificación:
    - Después de completar todas las pruebas individuales, integrar componentes y probarlos en conjunto.

- **Tipos de Pruebas de Integración:**
  - Integración **Ascendente**:

    - **Comienzo:** Módulos de bajo nivel.
    - **Proceso:** Iniciar con pocos módulos, ir añadiendo más progresivamente.
    - **Objetivo:** Probar la aplicación completa**.**
    - **Dirección:** De abajo hacia arriba.

  - Integración **Descendente**:

    - **Comienzo: Esqueleto del programa principal.**
    - **Proceso:** Añadir módulos subordinados incrementalmente, probando después de cada incorporación.
    - **Objetivo:** Integrar toda la jerarquía de módulos.
    - **Dirección:** De arriba hacia abajo, con el programa principal como coordinador.



ACTIVIDAD 3.4

Investiga y describe las características clave de las estrategias de integración incremental ascendente y descendente. Incluye también las ventajas y desventajas de cada una de estas estrategias.

NOTA: No olvides citar las fuentes.

5.- Pruebas del sistema

La prueba del sistema está formada por un conjunto de pruebas cuya misión es ejercitar profundamente el software. Son las siguientes:

- **Prueba de Recuperación:**

  Objetivo: Forzar un fallo del software y verificar una recuperación adecuada.

- **Prueba de Seguridad:**

Objetivo: Confirmar la protección del sistema contra accesos no autorizados.

- **Prueba de Resistencia (Stress):**

Objetivo: Someter al sistema a situaciones de alta demanda de recursos.

Ejemplos: Crear casos de prueba que maximicen el uso de memoria, aumentar la frecuencia de datos de entrada que puedan causar problemas en un sistema operativo virtual, etc.

6.- Pruebas de aceptación o validación

- **Objetivo General:**
  - Probar el sistema en su totalidad.
- **Verificación y Estabilidad:**
  - Asegurar el cumplimiento individual de cada requisito.
  - Evaluar la estabilidad técnica y comprobar la ausencia de fallos.
- **Tipos de Pruebas Realizadas:**
  - Pruebas de Rendimiento: Observar la evolución del sistema bajo diferentes cargas.
  - Pruebas de Estrés: Verificar la eficiencia del sistema bajo condiciones extremas.
- **Etapas de Pruebas:**
  - Pruebas *Alfa*: Realización en un entorno controlado con especificaciones detalladas.
  - Pruebas *Beta*: Ejecución por usuarios en un entorno no controlado, sin la intervención directa de los desarrolladores.

6.1.- Automatización de pruebas

Frecuentemente, es imprescindible automatizar las pruebas o repetirlas después de realizar mantenimientos, actualizaciones o arreglos en el software.

Resulta beneficioso conservar toda la información relevante, como datos, conjuntos de pruebas y módulos de prueba, ya que podrían ser útiles en el futuro.

Si se han empleado herramientas específicas o sistemas adicionales, es aconsejable documentarlos y guardarlos en un repositorio para facilitar su uso automático en ocasiones posteriores.

ACTIVIDAD 3.5

En esta actividad, se presentarán distintos escenarios de desarrollo de software. Tu tarea es identificar y justificar la estrategia de prueba más adecuada (pruebas unitarias, de integración, de sistema, o de aceptación) para cada escenario. Considera las características específicas y requisitos de cada proyecto al tomar tu decisión. Tu elección deberá estar acompañada de una breve explicación que fundamente por qué esa estrategia de prueba es la más apropiada para el escenario dado.

Escenarios:

- **Desarrollo de una Aplicación de Salud Móvil: Implementación de una característica de seguimiento de actividad física que sincroniza datos con *wearables*.**
- **Actualización de un Software de Contabilidad**: Mejora de un sistema existente para agregar funcionalidades de reporte de impuestos.
- **Creación de un Juego Móvil**: Desarrollo de un nuevo juego para smartphones con múltiples niveles y funciones de interacción en línea.
- **Sistema de Reservas en Línea para Hoteles**: Integración de un nuevo motor de reservas en un sitio web existente de reservas hoteleras.

7.- Documentación para las pruebas

El estándar IEEE 829-1998 describe el conjunto de documentos que pueden producirse durante el proceso de prueba. Son los siguientes:

- **Plan de pruebas:** Describe el alcance, el enfoque, los recursos y el calendario de las actividades de prueba. Identifica los elementos a probar, las características que se van a probar, las tareas que se van a realizar, el personal responsable de cada tarea y los riesgos asociados al plan

- **Especificaciones de prueba:** Están cubiertas por tres tipos de documentos:
  - Especificación del diseño de la prueba → Se identifican los requisitos, casos de prueba y procedimientos necesarios para llevar a cabo las pruebas

Se especifica la función de los criterios de pasa la prueba o no pasa la prueba.

  - Especificación de los casos de prueba → Documenta los valores reales utilizados para la entrada junto con los resultados previstos.

  - Especificación de los procedimientos de prueba → Se identifican los pasos necesarios para hacer funcionar el sistema y ejecutar los casos de prueba especificados.

- **Informes de pruebas**: Se definen cuatro tipos de documentos:
  - Informe que identifican los elementos que están siendo probados.
  - Un registro de las pruebas → donde se registra lo que ocurre durante la ejecución de las pruebas
  - Un informe de incidentes de prueba → Describe cualquier evento que se produce durante la ejecución de la prueba que requiere mayor investigación.
  - Un informe resumen de las actividades de prueba.

ACTIVIDAD 3.6

Define los estándares ISO/IEC/IEEE 29119 y IEEE 829-1998, destacando las principales diferencias entre ambos.

8.- Referencias bibliográficas

- Moreno Pérez, J.C. *Entornos de desarrollo.* Editorial Síntesis.
- Ramos Martín, A. & Ramos Martín, M.J. *Entornos de desarrollo*. Grupo editorial Garceta.