ENTORNOS DE DESARROLLO UD-5

PRUEBAS

ÍNDICE

[PLANIFICACIÓN DE LAS PRUEBAS 3](#_Toc129865737)

[PRUEBAS UNITARIAS 3](#_Toc129865738)

[A. PROCEDIMIENTOS Y CASOS DE PRUEBAS 4](#_Toc129865739)

[B. TIPOS DE PRUEBAS 4](#_Toc129865740)

[C. PRUEBAS DE REGRESIÓN 4](#_Toc129865741)

[DOCUMENTACIÓN DE LA PRUEBA 5](#_Toc129865742)

[NORMATIVA DE VALIDACIÓN 6](#_Toc129865743)

[VALIDACIÓN 6](#_Toc129865744)

# PLANIFICACIÓN DE LAS PRUEBAS

Durante el proceso de desarrollo del software suceden multitud de errores, para cerciorarnos de que todo es correcto, debemos hacer una serie de pruebas para corroborar que lo que hemos hecho es lo que ha pedido el cliente.

Las pruebas tratan de:

* **Verificar**: Comprobar que se cumplen las condiciones impuestas. “Tiene que ser en Windows”.
* **Validar**: Comprobar que se satisface los requisitos especificados. “Tiene que sumar dos números.”.

Existen diferentes estrategias, siendo la más común el modelo en espiral: yendo desde **pruebas unitarias** (realizadas sobre cada módulo) a **pruebas de integración** (realizadas poniendo los módulos en común), pasando a **pruebas de validación** (en estas comprobamos que el software se comporta como debe comportarse) y finalizando por **pruebas de sistema** (en ellas se comprueba que el software hace lo que debe hacer según el cliente).

Su objetivo es **garantizar un software libre de errores**. Un éxito de la fase de pruebas no es que no haya errores sino encontrar errores que se puedan subsanar.

Las pruebas las debe realizar un **equipo independiente al desarrollador**. Ya que el desarrollador va predispuesto a lo que sabe que ha hecho y puede, por ejemplo, no meter un valor que conoce que va a producir un error.

# PRUEBAS UNITARIAS

Son por las que se debe comenzar y son el tipo de pruebas que afectan a cada módulo (método -en POO- o una función -en programación estructurada-) y se caracterizan por ser:

* **Automatizables**: No debe requerirse una intervención manual.
* **Completas**: Deben cubrir la mayor cantidad de código.
* **Repetibles o reutilizables**: No se deben crear pruebas que sólo puedan ser ejecutadas una vez.
* **Independientes**: Que la ejecución de una no debe afectar a la ejecución de otra.
* **Profesionales**: Que se deben realizar con rigor.

El objetivo de estas pruebas es aislar el código de la forma más pequeña posible eliminando errores de abajo hacia arriba. Tiene las siguientes ventajas:

* **Fomentan el cambio**. Al encontrar errores en códigos pequeños es fácil realizar cambios. Una vez realizados los cambios, habrá que hacer nuevas pruebas, por ello las pruebas deben ser automatizables y reutilizables.
* **Simplifican la integración**. Al eliminar los errores de los módulos independientes al integrarlos va a ser más sencillo todo.
* **Documentan el código**. Las pruebas se consideran documentación del código puesto que ahí se puede ver cómo utilizarlo. Es una parte muy importante de la documentación.
* **Separación de la interfaz y la implementación**. Se realizan pruebas para la interfaz y otras para la implementación.
* **Los errores están acotados y son más fáciles de localizar**. Dado que tenemos pruebas unitarias que pueden desenmascararlos.

## PROCEDIMIENTOS Y CASOS DE PRUEBAS

Según el IEEE, un caso de prueba es un conjunto de entradas, condiciones de ejecución y resultados esperados, desarrollados para un objetivo particular como, por ejemplo, ejercitar un camino concreto de un programa o verificar el cumplimiento de un determinado requisito, incluyendo toda la documentación asociada.

Es imposible comprobar todo el software, por ello se hace necesario crear un diseño de las pruebas que se van a realizar y, además, se llega a un compromiso de estabilidad (cuando el software llega a una estabilidad óptima se lanza).

Para llegar a esta estabilización se debe llegar a un compromiso entre recursos, material humano, tiempo de software parado… y la probabilidad de que se detecten los errores existentes. Se debe de llegar a un nivel de equilibrio entre aceptabilidad de errores y recursos utilizados.

## TIPOS DE PRUEBAS

Existen principalmente las siguientes:

* + **Pruebas de caja negra o funcionales**. No sabemos cómo funciona el software pero sí cuál es la entrada y la salida. Por ejemplo, sé que 3 + 3 (entrada) da como resultado 6 (salida) pero no sé cómo lo hace el software durante el proceso.

Diagrama

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

* + **Pruebas de caja blanca o estructurales**. Se debe saber la lógica interna del software y lo que se va a valorar es la eficiencia, es decir, localizar estructuras incorrectas o ineficientes del código.

Diagrama

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

* + **Enfoque aleatori**o. Son menos sistemáticas y sólo están recomendadas para usuarios que sólo se dediquen a comprobar software y tengan mucha experiencia en ello.

## PRUEBAS DE REGRESIÓN

Se llevan a cabo cuando se ha producido un cambio en el código tanto para corregir un error o para implementar una mejora.

Existen tres clases de pruebas:

1. Hacer una **diversificación de las funciones** del software.
2. **Pruebas concretas** en las **funciones** que se van a ver afectadas por el cambio.
3. **Pruebas concretas** en los **componentes software** que han cambiado.

# DOCUMENTACIÓN DE LA PRUEBA

Es indispensable en el desarrollo software y viene definida por metodologías estándar como por ejemplo Métrica v. 3 que se basa en los estándares ANSI / IEEE.

Los documentos son:

* **Plan de pruebas**. Planificación de las pruebas a nivel del sistema. Pertenece a la fase de análisis del sistema.
* **Especificación del diseño de pruebas**. Determina pautas de prueba para los bloques en los que se ha dividido el programa. Se hace en la fase de diseño.
* **Especificación de un caso de prueba**. Los casos de prueba se concretan a partir de la especificación del diseño de pruebas. Fundamentalmente, se nutre de los casos obtenidos para las pruebas de caja blanca y caja negra. Se utilizan durante la fase de implementación.
* **Especificación de procedimiento de prueba**. Una vez determinados los casos de prueba, será preciso detallar el modo en que van a ser ejecutados cada uno de ellos, quedando recogidos en el documento "Especificación del procedimiento de prueba". También deberá incluir los resultados esperados en cada caso de prueba. Determina las pruebas unitarias a realizar.
* **Registro de pruebas**. El qué ha pasado con las pruebas, es decir, los resultados obtenidos en las pruebas.
* **Informe de incidente de pruebas o informe final**. Para cada incidente, defecto detectado, solicitud de mejora, etc.

# NORMATIVA DE VALIDACIÓN

La norma ISO/IEC 29119 que reúne al completo todo el proceso de pruebas.

* Parte 1. Conceptos y vocabulario que se deben utilizar.
  + Introducción a la prueba.
  + Pruebas basadas en riesgo.
  + Fases de prueba (unidad, integración, sistema, validación) y tipos de prueba (estática, dinámica, no funcional, ...). Prueba en diferentes ciclos de vida del software.
  + Roles y responsabilidades en la prueba.
  + Métricas y medidas.
* Parte 2. Procesos de prueba.
  + Política de la organización.
  + Gestión del proyecto de prueba.
  + Procesos de prueba estática.
  + Procesos de prueba dinámica.
* Parte 3. Documentación.
  + Contenido.
  + Plantilla.
* Parte 4. Técnicas de prueba que se van a utilizar (descripción, ejemplos, tipos…).
  + Descripción y ejemplos.
  + Estáticas: revisiones, inspecciones, etc.
  + Dinámicas: caja negra, caja blanca, técnicas de prueba no funcional (seguridad, rendimiento, usabilidad, etc).

# VALIDACIÓN

Consiste en de forma colaborativa con el cliente comprobar que lo que se ha hecho es correcto.

Tanto el plan como el procedimiento estarán diseñados para asegurar que se satisfacen todos los requisitos: funcionales, de rendimiento, de portabilidad, de compatibilidad, de recuperación de errores, de facilidad de mantenimiento, etc.