Pruebas en el desarrollo de interfaces

Desarrollo de interfaces

2º DAM

Objetivos

- Definir una estrategia de pruebas que se adapte al desarrollo particular.
- Definir y evaluar diferentes tipos de pruebas en las distintas fases del desarrollo.
- Comprender la importancia de la realización de los distintos tipos de pruebas.
- Conocer algunas herramientas que se utilizan en la realización de pruebas.
- Documentar la estrategia de pruebas y los resultados obtenidos.

El proyecto de desarrollo

- La división en fases de un proyecto va a permitir organizar todo el proceso que conlleva el desarrollo de un nuevo producto, ya sea físico o digital.
- La división en fases aportará un orden lógico a la gestión del proyecto, simplificando la gestión de este en tareas más pequeñas y manejables.

Fases de un proyecto de desarrollo

- ✓ Planificación
- ✓ Diseño
- ✓ Implementación
- ✓ Evaluación
- ✓ Producción

Planificación

Fase donde se asientan las bases sobre las que se trabajará durante todo el proceso, con relación al calendario de tiempos, tareas a desarrollar, costes, etc. En esta fase se definen las estrategias de pruebas a realizar.

Diseño

Fase en que comienza la construcción de la aplicación como tal, según las pautas marcadas en la fase de planificación

Implementación

Durante esta fase se realiza el desarrollo de la aplicación propiamente dicho, a partir del lenguaje de programación y arquitectura definida en fases anteriores.

Evaluación

En esta fase se comprueba que la funcionalidad del sistema es la adecuada y que cumple el objetivo planteado en un principio. Es una fase muy importante en el desarrollo del proyecto.

Producción

Fase en la que la aplicación desarrollada se implanta definitivamente en el cliente. Aunque el funcionamiento de la aplicación ya ha sido testeado previamente, se trata de una fase crítica.

Protocolo de pruebas

- Es conveniente considerar un protocolo de pruebas en cada fase del desarrollo que ayude a detectar posibles problemas e inconvenientes antes del paso a la siguiente fase.
- Es un proceso de especial relevancia en cualquier proyecto, pero especialmente delicado en el ámbito del desarrollo de aplicaciones.
- Detectar un problema de planificación o diseño o en la fase de evaluación o producción puede suponer un aumento de costes considerable en comparación a su detección en fases más tempranas.

Protocolo de pruebas

- Podemos diferenciar las pruebas que se realizan en cada una de las fases o módulos del desarrollo del programa, de las realizadas sobre el funcionamiento general de la aplicación.

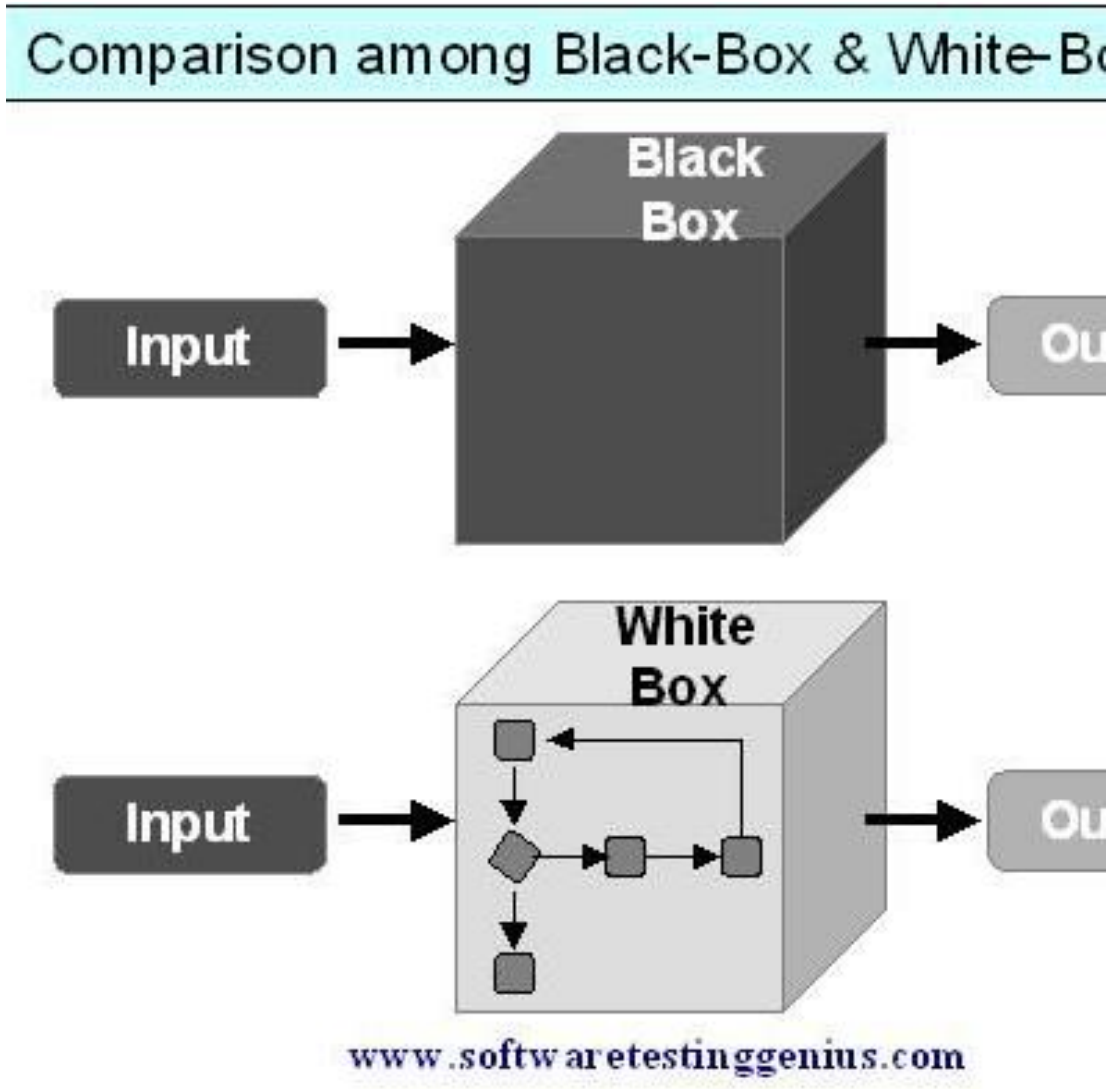
- Es muy importante diseñar un protocolo de pruebas exhaustivo, que permita garantizar el correcto funcionamiento de la aplicación y, por tanto, el éxito de esta.

Pruebas de caja negra y caja blanca

- Pruebas de caja negra: se evalúa la aplicación desde un punto de vista externo, es decir, sin preocuparnos del “interior”.
- Pruebas de caja blanca: se basan en la evaluación del código interno del software. Un buen diseño para estas pruebas implica la evaluación de todos los posibles caminos que se han implementado en el diseño de un programa.

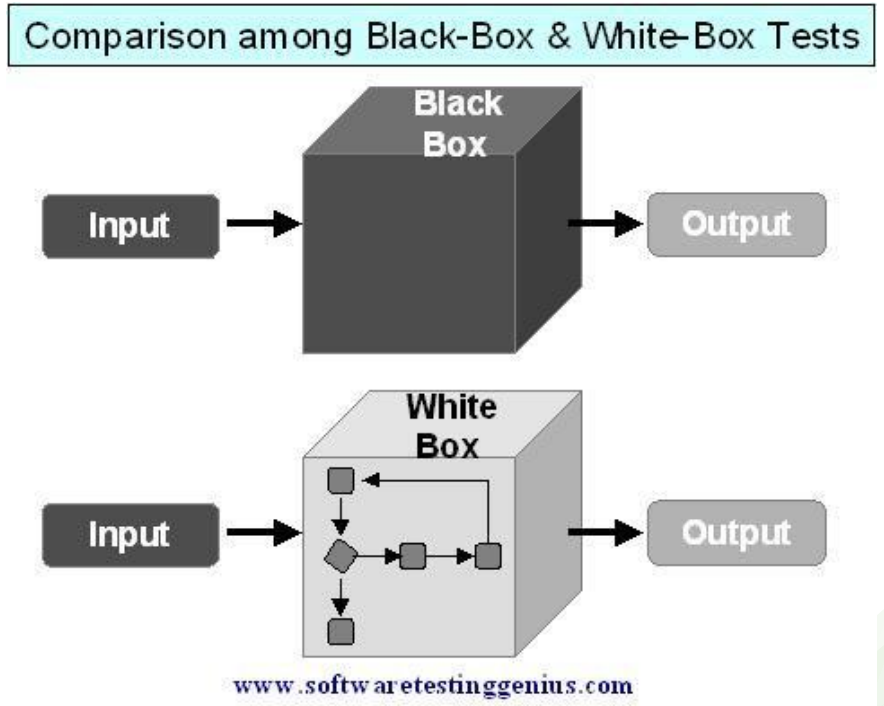
Pruebas de caja blanca

- La prueba de **caja blanca** o white-box testing, es un conjunto de técnicas que tienen como objetivo validar la lógica de la aplicación. Las pruebas se centran en **verificar la estructura interna del sistema** sin tener en cuenta los requisitos del mismo.
- Uno se centra en el código del módulo, por ejemplo, un método y se prueba en función a él.
- Como norma general se intenta que cada instrucción se ejecute al menos una vez y que cada decisión se evalúe al menos una vez a cierto y otra a falso.
- Así, algunas herramientas usadas:
  - ✓ **Control de flujo:**
    - Realización de un diagrama de flujo del algoritmo a probar, centrándose en que se recorran todos los posibles caminos al menos una vez, implicando ejecutar todas las instrucciones. Para facilitar el diseño de estas pruebas vemos cada bifurcación, cada camino, cubrir todas las instrucciones y cubrir todas las decisiones a T/F.
  - ✓ **Flujo de datos:**
    - Seguir el flujo de los datos durante todo el proceso. La depuración es útil aquí.



Pruebas de caja negra

- La prueba de **caja negra** o black-box testing se basa en comprobar la **funcionalidad de los componentes** de una aplicación. Determinados datos de entrada a una aplicación o componente deben producir unos resultados determinados.
- Este tipo de prueba está dirigida a **comprobar los resultados** de un componente de software, no a validar como internamente ha sido estructurado.
- Se llama black-box (caja negra) porque el proceso de pruebas asume que se **desconoce la estructura interna** del sistema, solo se comprueba que los datos de salida producidos por una entrada determinada son correctos.
- Hoy en día, las **pruebas unitarias** son la técnica black-boxing más extendida



Objetivos principales del sistema de pruebas

- Permiten validar el modo en que el software funciona, comparándolo con las especificaciones y objetivos de partida.
- Permiten detectar y corregir errores en el software en fases más tempranas, facilitando así su depuración a lo largo del desarrollo.
- Permite probar y evaluar la aplicación en diferentes escenarios y situaciones.

Depuración de código

- En muchos casos al realizar las pruebas sobre un desarrollo de software, será necesario depurar el código, en función del resultado de dichas pruebas y del objeto final de la aplicación.
- En el caso de la depuración de código, el objetivo principal es detectar y corregir errores en el mismo, en muchos casos a partir de las pruebas realizadas en cada una de las fases del proyecto.

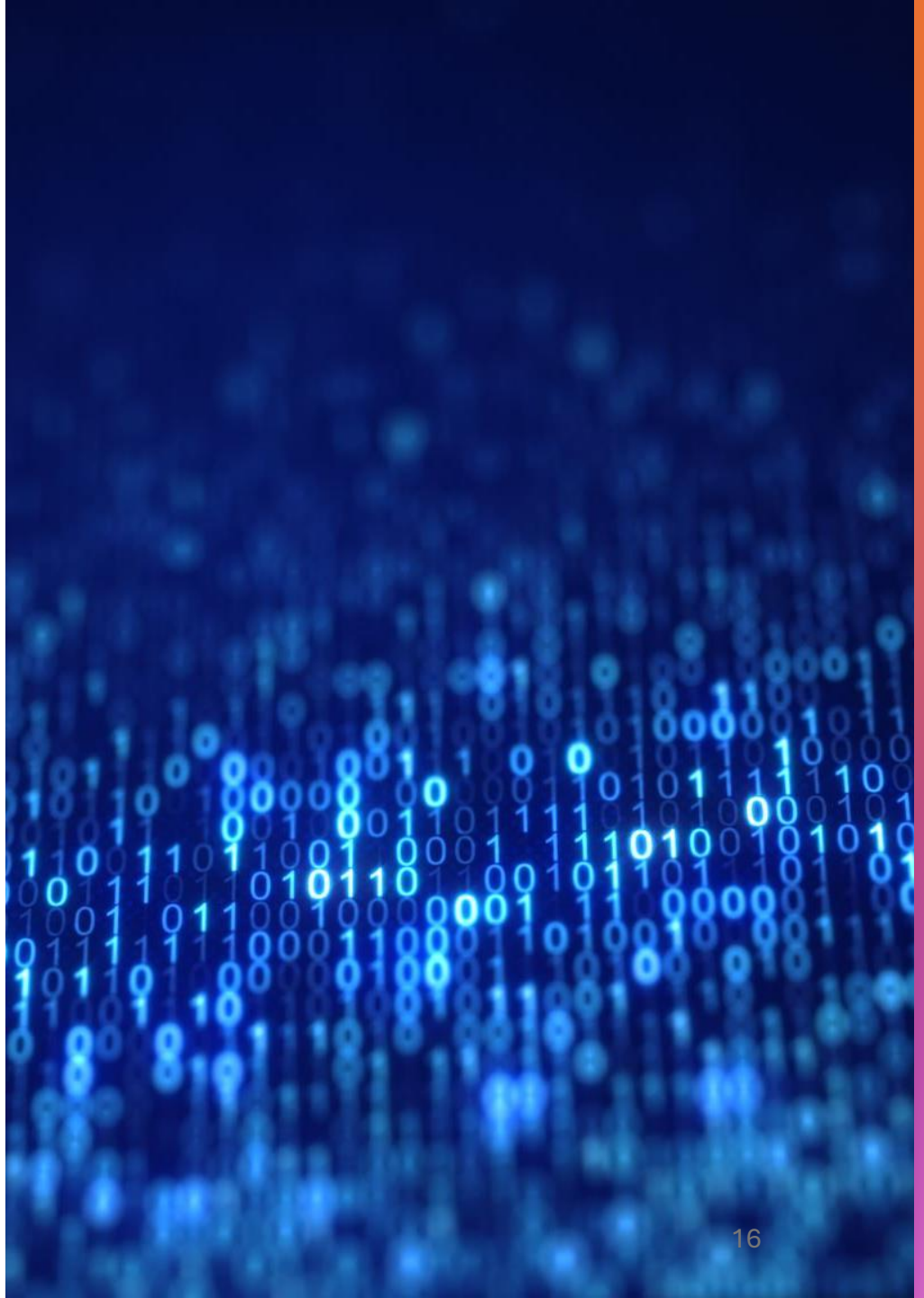
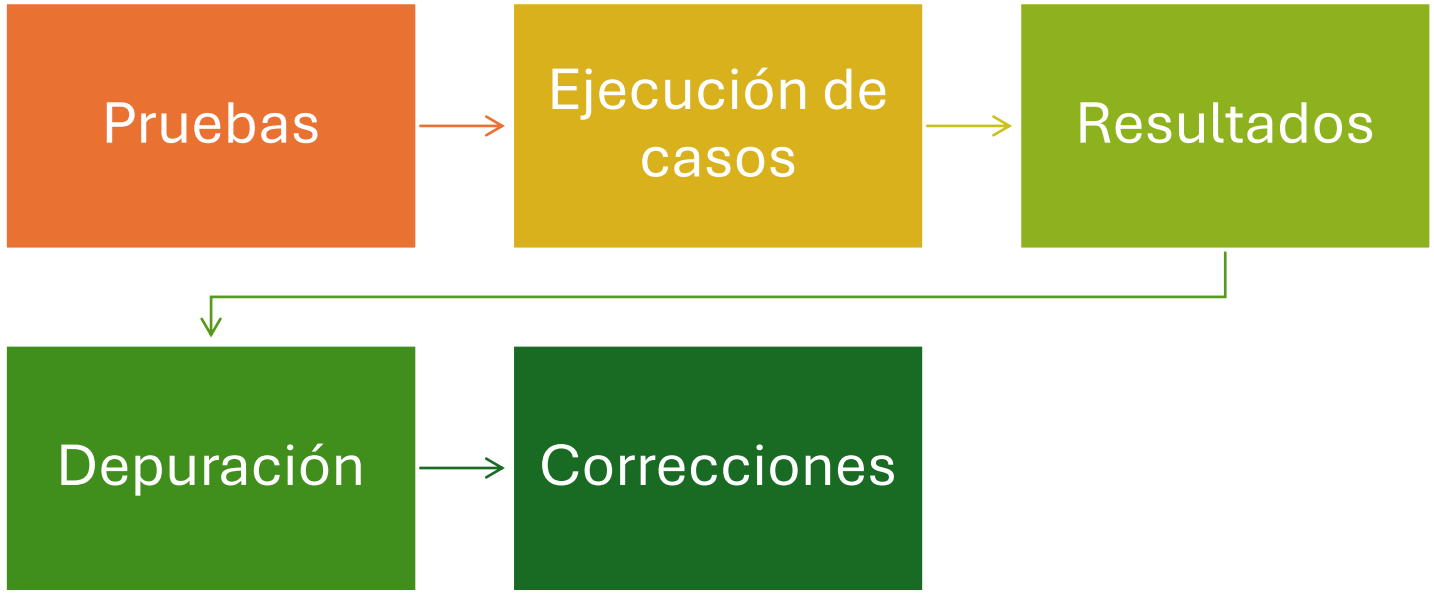
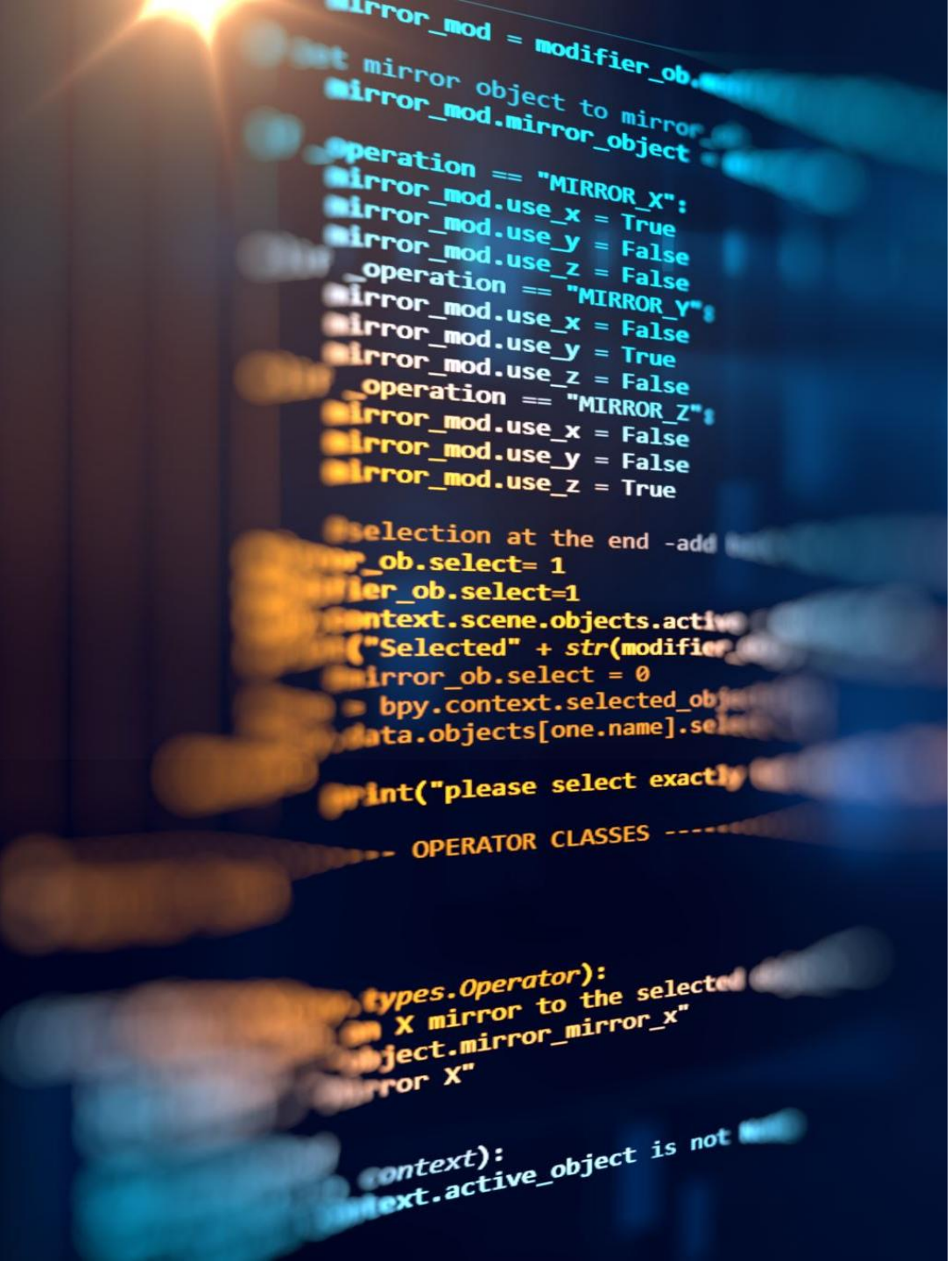


Diagrama de pruebas y depuración de código





Tipos de errores que aparecen durante el desarrollo

- **Errores de compilación:** en gran parte de los casos ocurren debido a la sintaxis, que varía en función del lenguaje de programación que se use.
- **Errores de ejecución:** suelen aparecer cuando existen operaciones incorrectas, que no están permitidas. Habitualmente el sistema generará un mensaje de error indicando el motivo.
- **Errores de lógica:** motivados por un error en el diseño del programa, ya que el resultado que se produce no es el esperado. Es más complicado detectar y, por tanto, de corregir, ya que en este caso el programa se ejecuta de forma normal, sin arrojar ningún error.

Tipos de pruebas

- Adicionalmente a la clasificación de pruebas como de caja negra o caja blanca, se plantea otro tipo de clasificaciones, según el tipo de funcionalidad que se evalúe en cada caso.

Pruebas unitarias

- Son utilizadas para evaluar funcionalidades concretas, examinando todos los caminos posibles implementados en el desarrollo de un algoritmo, función o clase.
- Una prueba unitaria es aquella que permite comprobar el funcionamiento de uno de los módulos que forman el programa.
- Tras evaluar el funcionamiento unitario de la aplicación, se procede con las pruebas en las que se engloban el resto de módulos.

✓ Ejemplo de pruebas unitarias

Pruebas de integración

- Se utilizan con el fin de aportar una garantía relacionada con el funcionamiento adecuado de la aplicación, una vez se han integrado todos los módulos que componen la misma, de ahí el nombre de estas pruebas.
- Este tipo de pruebas ofrecen la posibilidad de analizar el correcto funcionamiento de todos los módulos de una forma conjunta, ya que es posible que se hayan aplicado otro tipo de pruebas sobre estos, pero de manera separada.
- Por lo tanto, la importancia de este tipo de pruebas en el entorno de desarrollo es crucial, ya que es habitual que una aplicación de cierta complejidad esté compuesta por varios módulos.

Pruebas de integración: tipos

- **Pruebas de integración ascendente:** consisten en evaluar en primer lugar los niveles o módulos más bajos de la aplicación, para ir subiendo en los mismos de manera gradual.
- **Pruebas de integración descendente:** las pruebas se realizarán de manera inversa respecto a las pruebas de integración ascendente, es decir, se comienza desde el módulo principal y se va descendiendo gradualmente.

Pruebas de regresión

- Conjunto de pruebas que se habían ejecutado anteriormente sobre el sistema, y que se utilizan para buscar evidencias relacionadas con modificaciones y/o cambios que se haya podido producir sobre el código del programa, con el fin de detectar si han podido ocasionar nuevos errores o fallos, que anteriormente no habían aparecido.
- En definitiva, estas pruebas consisten en volver a probar las partes del sistema que ya hemos probado anteriormente cada vez que realizamos cambios.

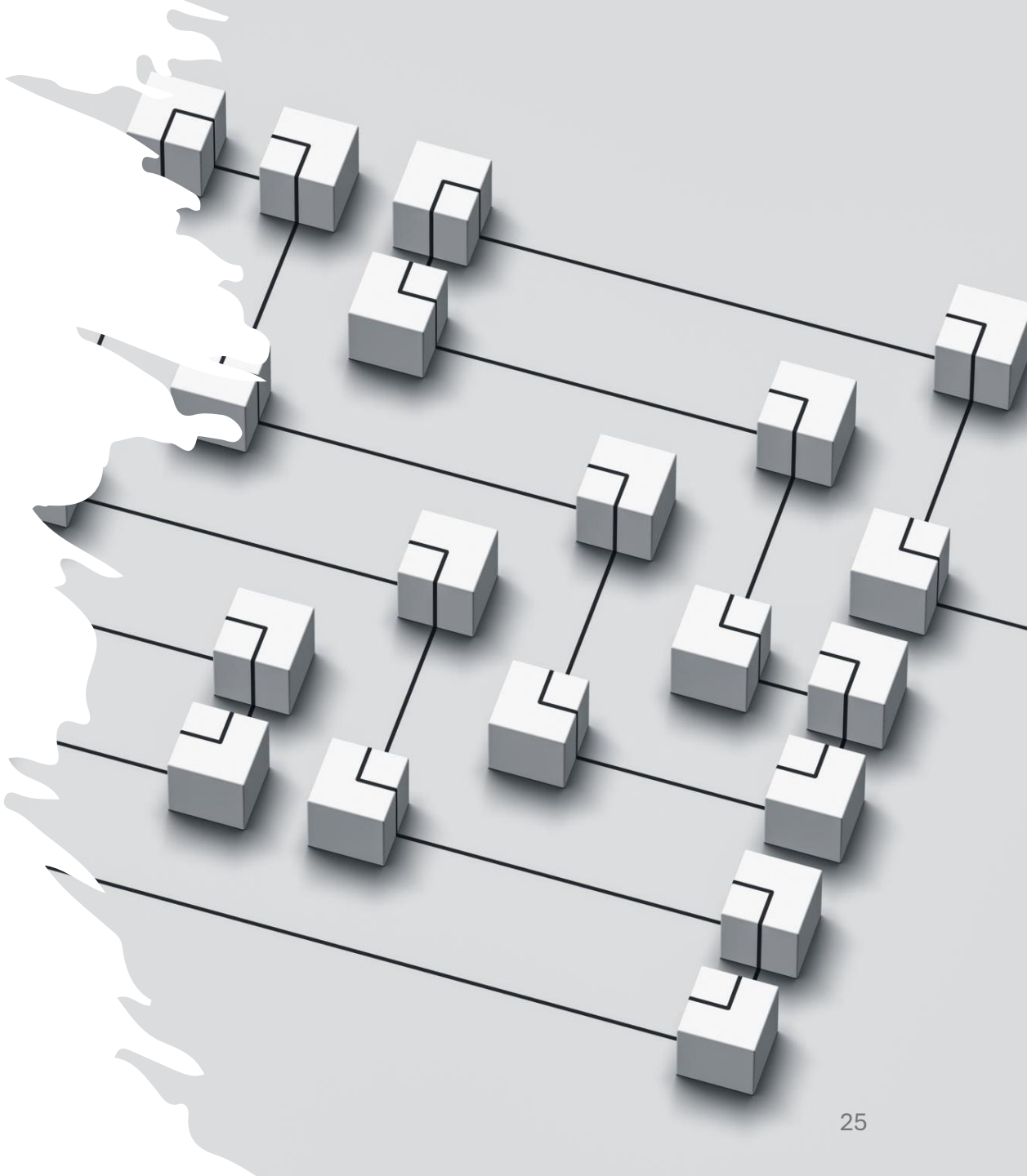
Tipos de pruebas de regresión

- **Pruebas de regresión locales:** tienen el fin de encontrar errores que hayan sido ocasionados por cambios o modificaciones recientes en el código.
- **Pruebas de regresión desenmascaradas o al descubierto:** estas pruebas tienen lugar cuando la realización de cambios ocasiona problemas que no tienen ninguna relación con los mismos, pero que se han detectado a raíz de su inclusión.
- **Pruebas de regresión remota o a distancia:** están relacionadas con la aparición de problemas ocasionados al integrar las distintas partes de una aplicación, considerándose que dichas partes por separado no arrojaban ese mismo problema.

Este tipo de pruebas son fundamentales para poder validar de manera correcta todos los cambios que se hagan en el código de la aplicación.

Pruebas funcionales

- Se caracterizan por evaluar al completo las funcionalidades que se indican en la especificación de la aplicación o herramienta desarrollada, concretamente con relación a sus requisitos de diseño.
- Al igual que sucede con el resto de las pruebas que se realicen, es muy importante que el proceso se documente de manera completar, con el fin de facilitar su análisis y comprensión.
- Estas pruebas deben realizarse por personal con experiencia en su desarrollo para que los resultados puedan ser interpretados de manera adecuada y correcta.



Pruebas no funcionales

- Las pruebas funcionales se basan en analizar cómo se comporta la aplicación a nivel interno. A diferencia de estas, las pruebas no funcionales suelen utilizarse con el fin de evaluar cómo se comporta la aplicación a nivel externo.
- Según lo que hemos estudiado en este tema, se podría afirmar que las pruebas funcionales son pruebas de caja blanca, mientras que las pruebas no funcionales son pruebas de caja negra.

Tipos de pruebas no funcionales

- **Prueba de capacidad.** Su uso suele estar ligado a evaluar el comportamiento de la aplicación ante una situación de estrés. Se utilizan para comprobar la respuesta del sistema ante el aumento de las peticiones o carga de trabajo que recibe, aplicándose una situación de estrés sobre el mismo.
- **Prueba de rendimiento.** El fin de este tipo de pruebas es comprobar cómo se comporta la aplicación desde el punto de vista de su eficiencia, teniendo en cuenta parámetros especialmente importantes como el tiempo de respuestas y la velocidad de procesado. Son muy importantes para decidir si es necesario o no optimizar procesos (y, por tanto, código).

Tipos de pruebas no funcionales

- **Prueba de estrés.** Suelen estar relacionadas con las pruebas de capacidad, ya que se basan en lanzar varias peticiones al sistema, con la salvedad de que en este caso lo que se pretende evaluar es cómo se recupera la aplicación ante una situación de sobrecarga, más que la respuesta que se da.
- **Prueba de volumen.** También se suelen desarrollar de manera paralela a las anteriores, ya que persiguen evaluar el sistema desde el punto de vista de su capacidad para procesar grandes volúmenes de datos.
- **Pruebas de seguridad.** Tienen como objetivo final aportar una garantía sobre la integridad de los datos de la aplicación. Así mismo, evalúan distintos mecanismos o formas de protección para la aplicación, con el fin de mejorar su robustez y seguridad.

Pruebas manuales

- Son ejecutadas por parte del encargado de desarrollar el código, con el fin de poder probar su funcionamiento y, en caso de ser necesario, modificar su implementación.
- El propio desarrollador es el que las ejecuta, evaluando por sí mismo si la respuesta de la aplicación a una determinada entrada es la correcta o no, en función del código implementado con anterioridad.
- Para su ejecución no se emplea una herramienta determinada si no que depende de cada programador. Habitualmente los IDE más habituales incorporan herramientas y/o funcionalidades para facilitar la realización de este tipo de pruebas.

Pruebas automáticas

- Requieren el uso de herramientas de pruebas para poder ejecutarse.
- Tanto las pruebas manuales como las automáticas son complementarias entre sí y tienen una gran importancia para conseguir un desarrollo de software correcto.
- Su ejecución suele ser más rápida que en el caso de las manuales y permiten comprobar el funcionamiento de la aplicación ante diversas variaciones en los datos y repetir las pruebas de manera rápida y sencilla.
- Se suelen utilizar para la realización de pruebas de regresión, ya que consiguen una optimización del procedimiento.

Pruebas automáticas

- El número de herramientas existentes en el mercado para llevar a cabo este tipo de pruebas es muy amplio, por lo que escoger una u otra dependerá del objetivo final, del tipo de prueba, de las preferencias del desarrollador y, por último, de la aplicación que se esté desarrollando.
- Algunas de las herramientas más utilizadas son:
  - **JMeter:** herramienta desarrollada por Apache que facilita la realización de pruebas de rendimiento y de carga sobre la aplicación.
  - **Bugzilla:** aplicación ejecutada online, que se usa para el seguimiento de errores en los módulos del software en cada una de sus versiones.
  - **JUnit:** se trata de un grupo de librerías desarrolladas en Java y que se utilizan para realizar pruebas unitarias sobre aplicaciones que han sido desarrolladas en este lenguaje.
  - **Cucumber:** es una herramienta de software libre, que facilita realizar pruebas de aceptación sobre aplicaciones web.
  - **Selenium:** se trata de un grupo de herramientas que se utilizan habitualmente para realizar pruebas de aplicación sobre desarrollos de aplicaciones web

Pruebas de usuario

- Son ejecutadas por uno o varios usuarios reales del sistema o aplicación que se está desarrollando, que no tienen por qué tener conocimientos de programación.
- Se utilizan de manera complementaria a otros tipos de pruebas que son ejecutadas por expertos, ya que es posible que estos no hayan tenido en cuenta ciertas situaciones que pueden darse en un entorno real de uso de la aplicación.
- Son pruebas basadas en la experiencia del usuario en la realización de sus tareas.
- Las pruebas se ejecutan según un guion previo y es aconsejable que participen en las mismas distintos tipos de perfiles, para probar a su vez diferentes funcionalidades y procedimientos.

Pruebas de aceptación

- Se ejecutan sobre una aplicación determinada para corroborar que su funcionamiento cumple con los requerimientos iniciales de diseño, es decir, su funcionamiento es el esperado en el momento del diseño de la aplicación.
- Se realizan desde el aspecto del rendimiento de la aplicación, así como desde su funcionalidad, dado que en ambos casos se plantean requerimientos en la fase de diseño del desarrollo.
- Evidentemente, la aplicación resultante debe satisfacer las necesidades planteadas en cuanto a funcionamiento, pero también ante tiempo de respuesta, comportamiento, estabilidad, etc., con el fin de que la experiencia de usuario sea grata y adecuada.

Pruebas de aceptación. Características

- Es habitual que estas pruebas sean definidas por el cliente para el que se está desarrollan do la aplicación. Es evidente que el grado de satisfacción sobre el producto dependerá de las expectativas existentes sobre el mismo.
- Suelen ejecutarse de manera previa a la implantación definitiva de la aplicación.
- Al igual que ocurre con otros tipos de pruebas, es muy importante que se documenten de manera correcta, recogiéndose todas las evidencias posibles sobre las mismas.

Desarrollo del plan de pruebas

El flujo que se sigue para ejecutar estas pruebas es el siguiente:

- ✓ Unitarias
- ✓ Integración
- ✓ Sistema
- ✓ Regresión
- ✓ Funcionales
- ✓ Capacidad
- ✓ Recursos
- ✓ Seguridad
- ✓ Usuario
- ✓ Aceptación

Versiones alfa y beta

- Estas pruebas se utilizan para detectar errores que solo pueden ser descubiertos por parte del usuario final, y que no pueden ser detectados por todas las pruebas realizadas de manera previa a la puesta en producción de la aplicación.
- Su fin es recoger posibles errores que no hayan sido detectados anteriormente, y que solo pueden recogerse cuando se testea la aplicación por usuarios reales, en los diferentes entornos y condiciones en los que se vaya a utilizar la misma.

Versión ALFA

- Primera versión de la misma, que será probada inicialmente por un grupo de personas, cuyo objetivo es simular el uso que le daría el cliente o usuario final.
- Evidentemente, se trata de una versión que aún no está del todo finalizada para poder implantarse en producción, sino que debe ser previamente depurada y evaluada.
- Habitualmente se ejecutan en las mismas oficinas del cliente final para reproducir más fielmente el escenario real de uso.

Versión BETA

- Versión prácticamente final de la aplicación.
- Se trata de la primera versión del desarrollo que será probada directamente por el cliente o usuarios finales de la misma, hecho que en sí mismo es la principal diferencia respecto a las pruebas alfa.
- Es muy común que se realicen en un entorno de aplicación lo más parecido al real para evitar que el mismo condicione el resultado de su uso.
- Los usuarios que prueban estas versiones se conocen como “beta tester”.