Unidad 10

Realización y sistemas de pruebas en el desarrollo de interfaces

Desarrollo de interfaces

Índice

Introducción

El control en la realización de un proyecto es esencial para que sea llevado a cabo correctamente, eso es algo que ya hemos aprendido con el estudio de la documentación en temas anteriores. El hecho de que este control sea tan exhaustivo nos debe dar una idea de la importancia que tomarán la realización de las pruebas durante el proceso.

Las pruebas durante el proceso, aún más cuanto mayor es el proyecto, permiten construir una aplicación libre, al menos en gran parte, de errores, que en otros casos permanecerían en el código obstruyendo otras secciones que si fueran correctas. Trabajar con errores tendría el mismo resultado que construir sobre malos cimientos, continuas grietas y posibilidad de colapso.

Si empleamos correctamente las pruebas podremos asegurarnos de que los fallos dentro del código sean mínimos y en su mayoría insignificantes, dándonos así un código de gran calidad que, a su vez, garantizará la calidad técnica de la aplicación.

Al finalizar esta unidad

- Conoceremos las fases de un proyecto y la importancia de las pruebas en él.
- Sabremos los distintos tipos de pruebas que podemos realizar, así como las distintas funciones y características de cada una de ellas.
- Comprenderemos cómo realizar un plan de pruebas adecuado y cuáles son las versiones previas al lanzamiento.

10.1. El proyecto de desarrollo

Un proyecto, generalmente, es demasiado extenso para poder realizarse de una sola vez o por un único equipo, por lo que es común segmentarlo con el fin de que su realización sea posible a través del cumplimiento de objetivos mucho más simples e inmediatos.

10.1.1. Fases de un proyecto de desarrollo

Las fases de un proyecte no suelen variar, salvo casos especiales donde se requieran elementos especiales, por lo que podemos citar estas como las fases de un proyecto:

1. **Planificación.**
2. **Diseño.**
3. **Implementación.**
4. **Evaluación.**
5. **Producción.**

Dado que dividimos el proceso en fases resulta más sencillo detectar problemas y en qué fase se producen, por lo que es aconsejable que se lleve a cabo un protocolo de pruebas que permita la revisión de diferentes elementos en busca de errores tras completar cada una de las fases.

Este tipo de protocolo permitirá detectar errores de forma temprana, lo que evitará que se trabaje sobre ellos, haciendo inservible el nuevo trabajo y provocando que dicho error sea difícil de localizar.

10.1.2. Objetivos principales del sistema de pruebas

Una buena fase de pruebas puede ser la diferencia entre un proyecto exitoso y otro fallido, ya que nos aporta numerosas ventajas, como son:

- Permite comprobar los resultados del software y compararlo con los objetivos propuestos inicialmente.
- Permite detectar y corregir errores en su etapa temprana, evitando que se conviertan en problemas mayores y difíciles de detectar en las siguientes fases del proyecto.
- Permite probar la aplicación en diferentes proyectos.

10.1.3. Pruebas de caja negra y caja blanca

Las pruebas de caja negra, que son las que solo comprueban el funcionamiento de la aplicación sin comprobar el código, son las siguientes:

- **Pruebas de comparación.**
- **Pruebas de clases de equivalencia de datos.**
- **Pruebas de valores límite.**

Las pruebas de caja blanca, también se denominadas clear box testing, consisten en emplear técnicas de análisis del código fuente del programa para detectar errores en las distintas partes de este.

Hay varias clases de este tipo de pruebas, como pueden ser:

- **Pruebas de flujo de datos.**
- **Pruebas de condiciones.**
- **Pruebas de bucles.**

10.1.4. Depuración de código

La depuración del código consiste en la revisión del código de la aplicación con la intención de localizar y corregir errores en el código. Generalmente no se realiza una búsqueda global, por su extenso costo, sino que se realizan en secciones en función de las pruebas llevadas a cabo.

Podemos localizar tres tipos de errores básicos al revisar el código:

- **Errores de compilación:** producidos principalmente por errores en la sintaxis del código.
- **Errores de ejecución:** producidos principalmente por operaciones no permitidas al contener elementos incorrectos.
- **Errores de lógica:** producidos por errores en el diseño del programa.

10.2. Tipos de prueba

Existen una multitud de pruebas que podemos emplear y, por lo tanto, una multitud de tipos de pruebas más haya de los conceptos de caja negra y blanca. Los tipos de pruebas más conocidos se muestran a continuación.

10.2.1. Pruebas unitarias

Estas pruebas normalmente se suelen dar durante las primeras fases del desarrollo del *software*. Es importante no retrasar las pruebas mucho porque luego deberemos de integrar todas las demás partes del proyecto, y en caso de fallos, costará más encontrarlos y repararlos.

Permite evaluar funcionalidades completas a través de los caminos posibles de cada algoritmo función o clase.

10.2.2. Pruebas de integración

Revisan el funcionamiento adecuado de la aplicación cuando la aplicación se completa uniendo los distintos módulos. Permitiendo el estudio del trabajo conjunto de los módulos, es decir, su comportamiento tras la integración. Podemos encontrar dos tipos diferentes de pruebas:

- **Pruebas de integración ascendente:** se evalúan todos los módulos comenzando desde el más bajo y subiendo desde hay hasta el principal.
- **Pruebas de integración descendente:** se evalúan todos los módulos comenzando desde el más alto, el principal y bajando desde hay.

10.2.3. Pruebas de regresión

Son pruebas que se realizan en diversos momentos, de modo que comparativamente podamos ver si los cambios realizados han modificado los resultados de la prueba llevada a cabo con anterioridad.

Podemos encontrar los siguientes tipos:

- **Pruebas de regresión locales:** buscan errores en el código por modificaciones recientes.
- **Pruebas de regresión desenmascaradas o al descubierto:** buscan cambios que no están provocados por las modificaciones en el código, pero que se han descubierto gracias a estas.
- **Pruebas de regresión remota o a distancia:** buscan errores producidos durante la integración de las distintas partes de la aplicación.

10.2.4. Pruebas funcionales

Con este tipo de prueba se analizan las distintas funcionalidades de la aplicación **a nivel interno**, comparando los resultados con los resultados esperados o prometidos, centrándose en los requisitos de diseño.

Este tipo de prueba, siguiendo la normativa ISO 25010,sigue cuatro principios por los que debe regirse:

- Idoneidad.
- Exactitud.
- Interoperabilidad.
- Seguridad.

En caso de que no siga alguno de estos principios, o no supere exitosamente las pruebas, se debe plantear la mejora o modificación de la funcionalidad.

10.2.5. Pruebas no funcionales: de capacidad, rendimiento, uso de recursos y seguridad

Este tipo de prueba comprueba el funcionamiento del tema a nivel externo. Junto con las pruebas funcionales podemos decir que estas son de caja blanca, mientras que las no funcionales serían de caja negra.

Este tipo de prueba destaca por:

- **Prueba de capacidad:** pruebas de estrés realizadas sobre la aplicación para comprobar su capacidad.
- **Prueba de rendimiento:** pruebas que buscan conocer el rendimiento máximo de la aplicación en diferentes situaciones, especialmente comprobando elementos como la velocidad de procesamiento, su tiempo de respuesta o cualquier otro elemento que afecte a la eficiencia de la aplicación.
- **Prueba de estrés:** otro tipo de prueba de estrés, pero no centrada en lo quela aplicación es capaz de soportar, sino en su recuperación después de que se sobrecargue.
- **Prueba de volumen:** pruebas que buscan conocer la respuesta y capacidad de una aplicación ante una gran cantidad de volumen de datos.
- **Pruebas de seguridad:** evalúan la integridad de la aplicación y sus sistemas de seguridad y protección.

10.2.6. Pruebas manuales

Son pruebas desarrolladas directamente por los desarrolladores con el fin de comprobar el comportamiento correcto de la aplicación. Aunque algunas aplicaciones como Eclipse cuentan con herramientas para facilitar la tarea, este tipo de pruebas no suele realizarse de manera automática, sino manual.

Se buscan errores a elementos que puedan ser mejorados en su implementación para un mejor uso del usuario.

10.2.7. Pruebas automáticas

Son aquellas pruebas que se realizan mediante el empleo de herramientas, de hay su designación como automáticas. Permiten revisiones rápidas de las funcionalidades de la aplicación.

Debido a su rapidez y capacidad para realizar varias veces la misma acción suelen ser empleadas para la realización de las pruebas de regresión.

Este tipo de pruebas requiere de una herramienta, entre las más usadas podemos encontrar las siguientes:

- **JMeter:** desarrollada por Apache se centra en pruebas de rendimiento y carga.
- **Bugzilla:** herramienta online que busca errores de *software* en distintas versiones.
- **JUnit:** desarrolladas por Java, conjunto de librerías especializadas para el desarrollo de pruebas unitarias.
- **Cucumber:** *software* libre empleado para pruebas de aceptación.
- **Selenium:** conjunto de herramientas especializadas para pruebas en aplicaciones web.

10.2.8. Pruebas de usuario

Son las llevadas a cabo por usuarios "reales" no poseen conocimientos técnicos y probarán la aplicación con la intención de descubrir cualquier error. Suelen contar con un número reducido, con un mínimo de unos 15 usuarios de distintos perfiles. Generalmente llevarán la prueba siguiendo un guion previo.

Un mayor número de usuario y perfiles conllevará mejores resultados.

10.2.9. Pruebas de aceptación

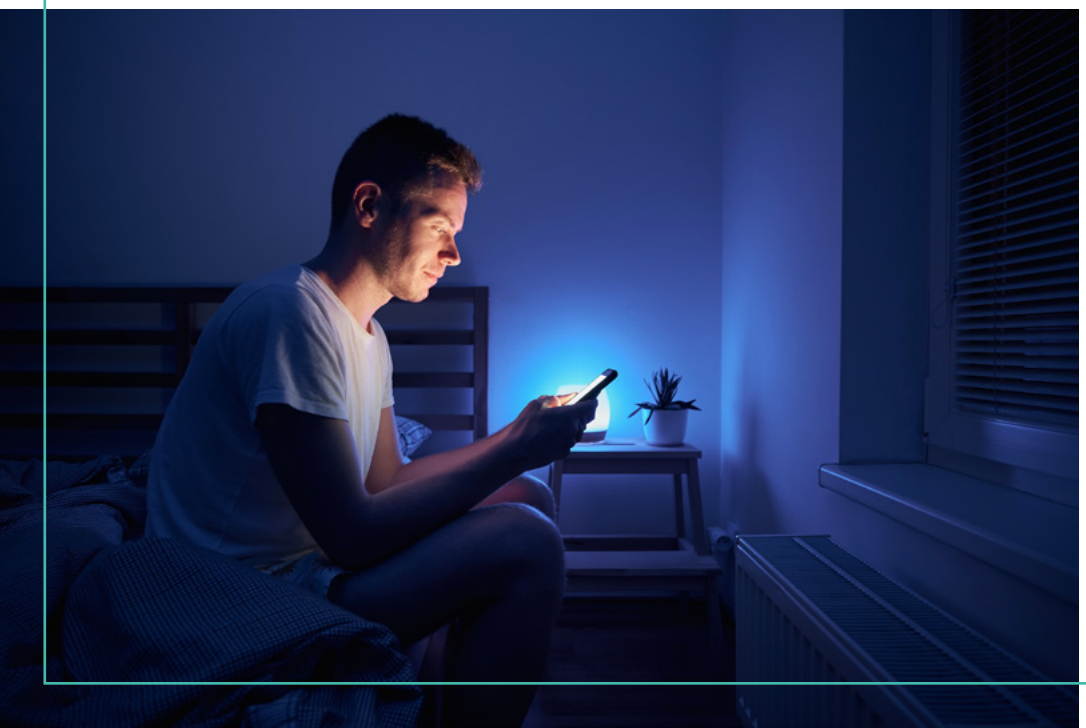
En las pruebas de aceptación se comprueba el programa al completo. No solo se comprobarán los requisitos del programa y su funcionamiento individualmente, también se observará si se cuenta con algún fallo técnico y que el funcionamiento del programa en general se mantiene en el tiempo.

Todo esto sirve para comprobar si la aplicación creada realmente cumple con los requisitos que se fijaron en el inicio del proyecto. Aunque la aplicación funcione correctamente, si no cumple los objetivos, es posible que se declare un fracaso.

Entre las características de este tipo de prueba podemos destacar el hecho de que se suelen realizar o preparar por el cliente, no por los desarrolladores.

10.2.10. Desarrollo del plan de pruebas

Tan importante pueden ser las pruebas que se realizan como el orden en el que son realizadas, ya que la realización de pruebas como las de esfuerzo, sin haber revisado antes fallos e incompatibilidades, solo nos proporcionara una cantidad ingente de fallos. Dependiendo del tipo de aplicación y sus necesidades especiales este podrá cambiar, pero se suele seguir un mismo plan. Este plan, generalmente seguido, es el siguiente:



10.3. Versiones alfa y beta

Las pruebas realizadas hasta ahora habrán detectado y eliminado la mayoría de los errores que podemos encontrar, pero aún quedarán errores, muchos de ellos errores que tan solo pueden ser detectados por el usuario.

Con el fin de detectar todos estos errores es común crear pruebas donde sean los propios usuarios los que detecten los errores a través de testeos, siempre que puedan masivos.

10.3.1. Versión ALFA

La primera versión de la aplicación que se lanza completamente funcional se denomina versión alfa, obviamente contendrá numerosos errores. Esta se emplea como muestra para el cliente y para ser empleada por un primer conjunto de usuarios, aún no reales y desde la oficina generalmente, en busca de errores.

10.3.2. Versión BETA

Es la primera versión probada por usuarios reales, generalmente de forma masiva, en busca de errores o defectos en la aplicación. En este tipo de pruebas los usuarios son llamados "beta tester".

En la actualidad, gracias a las facilidades que nos ofrece Internet y a las actualizaciones de las aplicaciones, se ha vuelto común lanzar la prueba beta directamente al público, e ir corrigiendo los errores mediante parches de actualizaciones, lo que permite sacar al mercado el producto antes y que la prueba beta se realice realmente de manera masiva por los usuarios.

www.universae.com