Pruebas de Software - Fontela

Contexto

La actividad de pruebas es una de las disciplinas características del desarrollo de software, tanto como la programación, el diseño o el análisis. No son las pruebas las que confieren calidad al producto. Las pruebas tienen solamente el objetivo de detectar errores.

Son una actividad de **control de la calidad (QC)**, para eso existen prácticas diversas denominadas de **aseguramiento de la calidad (QA)**, que buscan mejorar el proceso de desarrollo con la finalidad de que el producto final sea de calidad. Aunque nunca vamos a encontrar todos los errores de un programa haciendo pruebas.

Qué probamos

Verificación y validación

La **verificación** tiene que ver con controlar que hayamos construido el producto tal como pretendimos construirlo. La **validación**, en cambio, controla que hayamos construido el producto que nuestro cliente quería.

Funcionalidades y atributos de calidad

Pruebas funcionales: Son pruebas centradas en probar funcionalidades, cosas que el programa debe hacer. Estas pruebas surgen de modo natural de los requerimientos que el usuario expresa.

Pero en ocasiones ocurre que debemos probar características del sistema que no son funcionales. Decimos que estas son **pruebas de atributos de calidad**. Listado de algunos posibles tipos de pruebas de atributos de calidad:

- Pruebas de compatibilidad.
- Pruebas de rendimiento.
- Pruebas de resistencia o de estrés.
- Pruebas de seguridad.
- Pruebas de recuperación.
- Pruebas de instalación.

Alcance de las pruebas

Alcance de las pruebas de verificación (o técnicas)

Las **pruebas unitarias** verifican pequeñas porciones de código. Luego de un lapso corto de programación, se debería ejecutar alguna prueba unitaria que compruebe que vamos por el camino correcto. La ventaja está en que permiten aislarse del conjunto del sistema, y analizar que una pequeña porción de código se comporta como se espera.

Las **pruebas de integración** prueban que varias porciones de código, que trabajen en conjunto, hacen lo que pretendíamos. Estas no siempre prueban el sistema como un todo, a veces lo hacen con partes del mismo que se quieren aislar de otras.

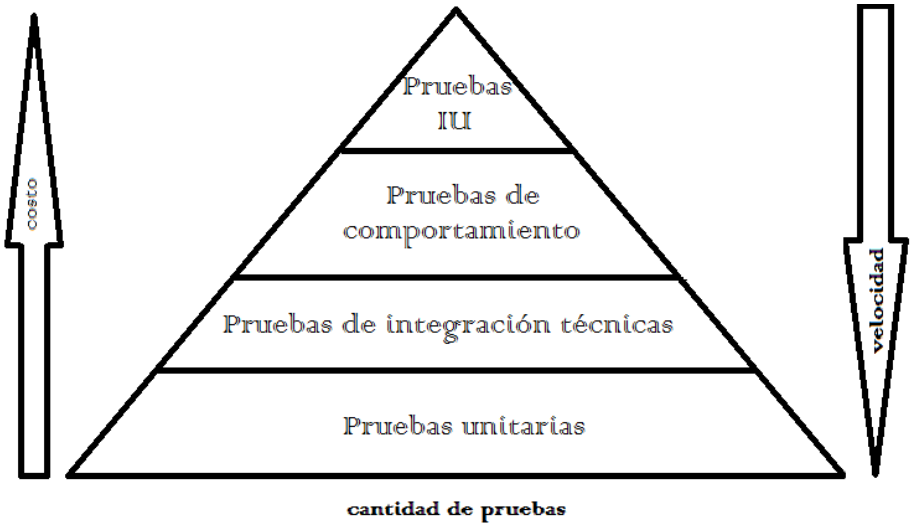
Las pruebas de verificación se pueden escribir antes del código que prueban, aunque evidentemente se ejecuten después. También, podrían ser de **caja negra**, es cuando la ejecutamos sin mirar el código que estamos probando (prueba ciega), o de **caja blanca**, es cuando analizamos el código durante la prueba. En general, se prefiere hacer pruebas de caja negra, precisamente porque se desea probar el funcionamiento y no verificar la calidad del código. Una prueba que ha caído en desuso es la llamada **prueba de escritorio**.

Alcance de las pruebas de validación

Pruebas de aceptación de usuarios (UAT): Son pruebas de aceptación diseñadas por usuarios, o al menos en conjunto con usuarios. Si las pruebas las hacemos en un entorno controlado por el equipo de desarrollo, las llamamos **pruebas alfa**. Si, en cambio, el producto se deja a disposición del cliente para que lo pruebe en su entorno, las llamamos **pruebas beta**. Estas suponen que se ejecuta el sistema completo.

Pruebas de comportamiento: Son pruebas mas limitadas, que se usan para probar si la lógica aplicada a determinado es correcta.

La pirámide de pruebas



Roles del desarrollo ante las pruebas

Visiones tradicionales

Sostiene que debe haber personas con el rol de programadores y otras con el rol de testers. Se basa en la noción administrativa de control cruzado por contraposición de intereses, que hace que el que realiza un trabajo no puede ser el mismo que lo controla.

La visión ágil

Plantea que todo el equipo de desarrollo debe trabajar de consuno y en aras de entregar un producto de calidad. Todos son desarrolladores, sin distinción entre programadores y testers.

Pruebas y desarrollo

Cuándo probamos? Cada vez que se incorporan características a un programa, podemos provocar que dejen de funcionar cosas que ya habían sido probadas y entregadas: esto es lo que se llama una **regresión**. Para evitarlas, se ejecutan cada tanto, **pruebas de regresión**, que no es otra cosa que ejecutar las pruebas de todo el sistema a intervalos regulares.

Automatización

Es el hecho de que las pruebas del programa se desarrollen en código, de forma tal que, con la sola corrida del código de pruebas, deberíamos saber si lo que estamos probando funciona bien o mal.

Ventajas:

- Nos independizamos del factor humano.
- Es más fácil repetir las mismas pruebas, con un costo ínfimo comparado con las pruebas realizadas por una persona.
- Pueden llegar a servir como comunicación.

TDD

Test-driven Development, primera práctica basada en automatización de pruebas.

Incluye tres sub-prácticas:

- Automatización: Las pruebas del programa deben ser hechas en código, y con la sola ejecución del código de pruebas debemos saber si lo que estamos probando funciona bien o mal.
- Test-First: Las pruebas se escriben antes del propio código a probar.
- Refactorización posterior: Para mantener la calidad del código, se lo cambia sin cambiar la funcionalidad, manteniendo las pruebas como reaseguro.

Ventajas:

- Las pruebas en código sirven como documentación del uso esperado de lo que se está probando.
- Las pruebas escritas con anterioridad ayudan a entender mejor lo que se está por desarrollar.
- Las pruebas escritas con anterioridad suelen incluir más casos de pruebas negativas que las que escribimos a posteriori.
- Escribir las pruebas antes del código a probar, minimiza el condicionamiento del autor por lo ya construido.