PRÁCTICAS DE TESTING ÁGIL

UTN FACULTAD REGIONAL CÓRDOBA

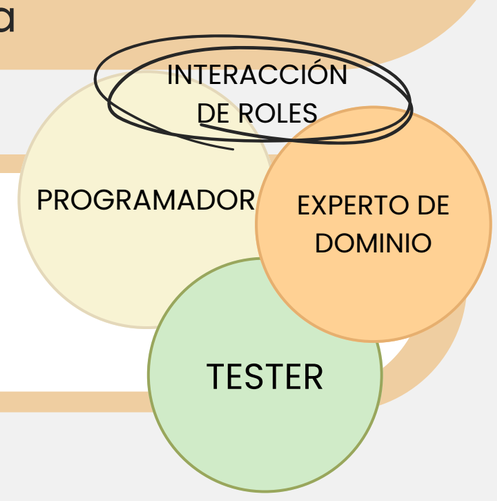
INGENIERÍA y CALIDAD de SOFTWARE

Grupo 5: Almendra, Arrascaeta, Dionisio, Hunziker, Menendez, Somavilla

¿QUÉ SON?

Testing Ágil es una metodología de trabajo colaborativo que se basa en dos pilares fundamentales: la **retroalimentación conjunta y la producción de calidad**. Para llevarlo a cabo fueron surgiendo diferentes prácticas de pruebas de software como herramientas o técnicas que se pueden utilizar y que siguen los principios de desarrollo ágil.

Desde este punto de vista, hay que comprender que se debe tener entendimiento sobre el **punto de vista del cliente**, como así también sobre las **complejidades técnicas de la implementación**.

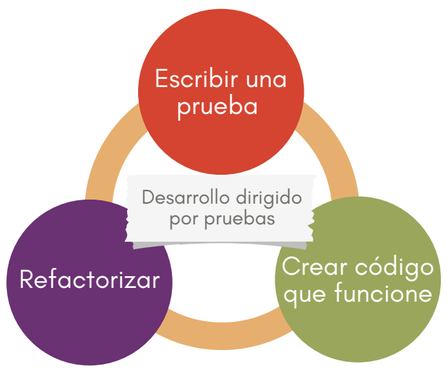


1

Desarrollo Dirigido por Pruebas (TDD)

El desarrollo dirigido por pruebas es un enfoque al diseño de programas donde se entrelazan el desarrollo de pruebas y el de código.

El programador escribe una prueba para un poco de funcionalidad, ve que falla, escribe el código que lo hace pasar, y luego pasa a la siguiente pequeña parte de la funcionalidad.



ESENCIA

El código se desarrolla incrementalmente, junto con una prueba para ese incremento. No se avanza hacia el siguiente incremento sino hasta que el código diseñado pasa la prueba.

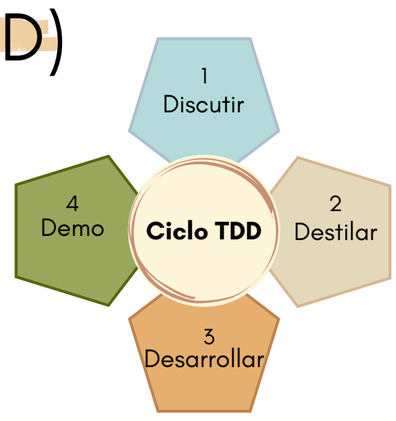
BENEFICIOS

1. COBERTURA DE CÓDIGO
2. REDUCIR COSTOS EN PRUEBAS DE REGRESIÓN
3. DEPURACIÓN SIMPLIFICADA
4. DOCUMENTACIÓN DEL SISTEMA

2

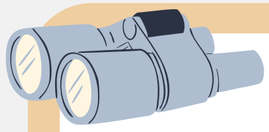
Desarrollo Dirigido por Pruebas de Aceptación (ATDD)

ATDD busca generar **colaboración entre el equipo**, para poder crear un entendimiento compartido de los requerimientos del sistema. Las especificaciones se convertirán en casos de uso para aceptación automatizados.
Busca resolver el problema de la implementación de features que no resuelven las necesidades del cliente por parte del equipo de desarrollo.



Tiene un ciclo de vida similar a TDD, consistente en crear las pruebas previo a desarrollar el código. En este caso en pruebas de aceptación

3



Pruebas Exploratorias

Son una estrategia de pruebas de software que suelen describirse como aprendizaje simultáneo, diseño de prueba y ejecución. Se centran en la detección y dependen de que el tester descubra defectos que no son fáciles de cubrir con otras pruebas.

¿PARA QUÉ SIRVEN?

Hoy en día los equipos deben adoptar la integración continua y satisfacer la demanda del mercado para así satisfacer también las crecientes expectativas de los clientes.
Aunque la velocidad de salida del mercado es importante, hay errores que cuestan millones o desastres en términos de experiencia de usuario sencillos, pero muy costosos

1. Probar el software
2. Aprender a manejar el sistema
3. Experiencia y creatividad
4. Generar nuevas pruebas a ejecutar

Caja Negra
Métodos basados en la experiencia.

4

Pruebas de Unidad e Integración Automatizadas

AUTOMATIZACIÓN

Las pruebas automatizadas se realizan a través de una máquina o framework que ejecuta un script de la prueba escrito con antelación.
Las pruebas automatizadas son mucho más potentes y fiables que las manuales, pero su calidad depende de lo bien que se hayan escrito los scripts de las pruebas.

UNIDAD

Estas pruebas verifican el funcionamiento de una sola unidad de código de forma aislada del resto de componentes.

Permite la detección de errores en...

- Funciones/Métodos
- Clases/Objetos
- Modulos/Librerias
- Algoritmos

- Incompatibilidad de interfaces o datos
- Problemas como cuello de botella
- Errores de lógica compartida
- Problemas de configuración

INTEGRACIÓN

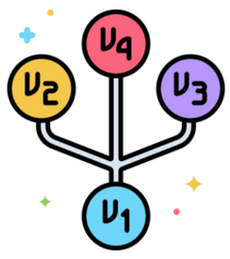
Destinadas a comprobar el funcionamiento de dos o más componentes en el sistema y la forma que funcionan en conjunto.

5

Control de versión de las pruebas con el código

El control de versión de pruebas con código es una herramienta que permite a los desarrolladores y equipos de pruebas rastrear y gestionar cambios en el código fuente de un proyecto de software a lo largo del tiempo. Se dejan registradas todas las modificaciones, quién las realizó y cuándo se hicieron, lo que facilita el seguimiento de la evolución del proyecto y la colaboración entre miembros del equipo.

IMPLICA



1. Seguimiento de cambios
2. Colaboración
3. Revertir cambios
4. Ramificaciones (Branches)
5. Integración continua

6

Pruebas de Regresión

La prueba de regresión es el proceso de probar un producto de software después de que se hayan introducido nuevos errores como resultado de los cambios.
Es fundamental para entregar un producto de alta calidad.
Se utiliza para confirmar que los cambios más recientes, no han afectado las funciones existentes de manera adversa.
Consiste principalmente en volver a ejecutar los casos de prueba para confirmar si la aplicación funciona correctamente.

CONCLUSIONES

- Mejora la calidad de software
- Mejora la eficiencia del equipo
- Detección temprana de errores al adoptar enfoques como
  - Integración continua
  - Automatización de pruebas
  - Colaboración estrecha entre desarrolladores y testers.
- Aceleran los ciclos de desarrollo
- Se garantiza que el producto cumpla con los requisitos del cliente

REFERENCIAS