PECHAKUCHA

Pruebas unitarias

1) Las pruebas unitarias se basan en la prueba de componentes individuales del software. Este tipo de pruebas se realizan generalmente durante el Desarrollo del sistema. Su objetivo es aislar una sección de código para evaluar si es correcto. Una unidad puede ser una función o procedimiento, clase, método u objeto. Las pruebas de unidad son llevadas a cabo por los desarrolladores y son una técnica de caja blanca

2)

3)

Pruebas de Regresión

1)Se definen como como un tipo de prueba de software para confirmar que una código o programa recientemente desarrollado no ha afectado a funcionalidades ya existentes. Es una selección completa o parcial de casos de prueba que se re ejecutan para asegurar que las funcionalidades existentes funcionan normalmente.

2)Las pruebas de regresión son necesarias cuando:

* Existe algún cambio de en los requerimientos y por ende el código se ve modificado
* Cuando se agrega una nueva funcionalidad
* Cuando se quiere arreglar algún defecto
* Cuando se quiere arreglar algún problema de performance

3)Algunas formas de hacer test de regresión son:

* Retestear todo, puede llegar a ser muy costoso debido a la magnitud de la pruba
* Selección de casos de prueba, los cuales pueden ser reusables u obsoletos.
* Priorizando casos de prueba dependiendo el impacto en el negocio y la frecuencia de uso y que tan críticas son las funcionalidades

Pruebas Exploratorias

1)

2)

3)

TDD

1. Test Driven Development (Desarrollo Dirigido por Pruebas) es una disciplina de diseño y programación, donde cada nueva línea de código que escribe un programador es en respuesta a una prueba que ha fallado, también escrita por el programador. Es parte medular del desarrollo ágil de código derivado de XP
2. Es aplicado mediante microiteraciones: El desarrollo es guiado por pruebas unitarias escritas por el propio programador donde se corren las pruebas y se refactoriza el código con errores. El objetivo es implementar solo lo imprescindible solicitado por el cliente, eliminando ambigüedades, reduciendo errores y creando software con gran aceptación a los cambios.  
   114
3. Ventajas:
   1. Al elaborar las pruebas primero el programador no siente que la prueba destruye su código.
   2. Al elaborar sus propias pruebas no somete su código a pruebas ajenas.
   3. Si introduce mejoras en el código o nuevas funcionalidades sabrá rápidamente si pasan todos las pruebas.

Desventajas:

Los ciclos extremadamente cortos y el cambio entre pruebas y codificación de producción hacen que muchos programadores sientan desgaste. Exige además disciplina, responsabilidad y coraje.

ATDD

1. ATDD (Acceptance Test Drive Develoment) o Desarrollo dirigido por Test de Aceptacion es una práctica en la que todo el equipo, destacando, e incluyendo, a los usuarios y/o al product owner, desarrolladores y tester, analiza conjuntamente los criterios de aceptación, antes de que comience el desarrollo.
2. Se aplica mediante conversaciones frecuentes acerca del alcance de las User Stories, se establecen sus criterios de aceptación entre cliente y desarrolladores. Asi es posible estimar mejor el tiempo y la complejidad de las tareas. Posteriormente se realiza el desarrollo como en TDD
3. Mediante las Pruebas de Aceptación es posible definir con ejemplos los criterios de aceptación de un desarrollo (lo que se conoce como “especificación mediante ejemplos”), de modo que los desarrolladores saben qué desarrollar y los usuarios saben qué tendrán.

Las Pruebas de Aceptación sirven también como documentación viva del proyecto a la vez que sirven de puente entre el equipo de desarrollo y la parte de negocio implicada, mediante un entendimiento común de los requisitos del desarrollo.

CONTROL DE PRUEBAS DE VERSIONES

1. Debido a que en cada iteración el software cambia también deben cambiar y actualizarse las pruebas que se realizan. En teoría, después de cada corrección uno debe ejecutar el conjunto completo de casos de prueba antes de ejecutar contra el sistema, para asegurarse de que no ha sido dañado de forma oscura. En la práctica, tales pruebas de regresión deben aproximarse a esta idea teórica, y es muy costoso
2. En el área de ingeniería de software y en particular las pruebas de software (Software Testing), existen nuevas herramientas para registrar información y presentarnos reportes de estatus en las diferentes fases de ciclo de vida, según las metodologías de desarrollo de software utilizadas. Asimismo, tenemos nuevas herramientas para automatizar los Tests.
3. Hay herramientas Open Source que nos facilitan esta tarea
   1. Para organizar las pruebas de software, Testlink permite
      1. Definir proyectos de pruebas (Test Project).
      2. Crear casos de prueba y su información (Test Case).
      3. Organizar los casos de pruebas en “conjuntos de pruebas” (Test Suite).
      4. Ejecutar los casos de prueba y registrar resultados.
      5. Visualizar los resultados de las pruebas (Test Results).
   2. Para automatizar las pruebas vamos a mencionar Selenium
   3. Para hacer un seguimiento del proyecto y gestionar incidencias Redmine es una aplicación de software para la gerencia de proyectos, puede funcionar en diversas plataformas y bases de datos de distintos proveedores.