GUIA DE PREGUNTAS

Material "ESTRATEGIAS DE PRUEBA DE SOFTWARE.

Velez Matias

Contenido

1. Defina estrategia de prueba de software.

Integra las técnicas de diseño de casos de prueba en una serie de pasos bien planificados que dan como resultado una correcta construcción del software. Da un mapa con los pasos a llevar a cabo, como se planifican, y los recursos que van a requerir.

2. Defina prueba de software.

Son un conjunto de actividades planificadas por adelantado y llevadas a cabo sistemáticamente, por eso se define una plantilla para las pruebas de software (con los pasos para situar los métodos de cada diseño de caso de prueba)

3. Enuncie las características generales de las estrategias de prueba del software.

1) Las pruebas comienzan a nivel modulo, y trabajan hacia la integración de todo el sistema
2) Según el momento, son apropiadas diferentes técnicas de casos de prueba
3) La prueba la lleva a cabo el responsable del desarrollo y un grupo independiente
4) La prueba y depuración son actividades diferentes, pero la depuración se incluye en cualquier estrategia de prueba

4. Defina verificación y validación

- **Verificación** es asegurarse de que el software implementa correctamente una función específica (que está bien construido).
- **Validación** es asegurarse de que lo que se construyó se ajusta a los requisitos del cliente (que se hizo lo que había que hacer)

5. Enuncie el conflicto de intereses inherente a las pruebas de software

En cualquier proyecto aparecen conflictos de intereses a la hora de pruebas. Se pide a quien lo hizo que lo pruebe, quien querrá demostrar que está libre de errores, y que funciona de acuerdo a las especificaciones. Esto intereses se convierten en inconvenientes a lo largo del proceso de prueba

6. Enuncie la responsabilidad de prueba que tiene el desarrollador del software.

Es responsable de probar todas las unidades (módulos), asegurándose de que funcionan correctamente. Y en algunos casos realiza la prueba de integración.

7. Enuncie la responsabilidad que tiene el grupo independiente de prueba del software.

El GIP se encarga de eliminar los problemas asociados con el hecho de permitir al constructor que pruebe lo que ha construido. Se ve implicado en el proceso de especificación y sigue implicado a lo largo del proyecto. También informa a la organización de garantía de calidad, que no sería posible si el GIP fuera parte de la organización que desarrollo el software.

8. Describa como la estrategia de prueba acompaña el modelo de ciclo de vida en espiral.

La prueba de unidad comienza en el vértice de la espiral, y se centra en las unidades del software.

La prueba avanza hasta llegar a la prueba de integración, donde el foco es el diseño y construcción de la arquitectura.

Dando una vuelta hacia fuera encontramos la prueba de validación, donde se validan los requisitos, y finalmente la prueba de sistema, que prueba el software como un todo

9. Defina prueba de unidad, prueba de integración, prueba de validación, prueba de alto nivel y prueba del sistema.

- **Prueba de unidad:** Se centra en cada módulo individualmente, asegurando que funcionan como unidad. Hace uso intensivo de las pruebas de caja blanca.

- **Prueba de integración:** Se dirige a todos los aspectos asociados con el doble problema de verificación y construcción. Prevalecen las pruebas de caja negra.

- **Prueba de validación:** Proporciona una seguridad final que el software satisface todos los requisitos funcionales, de comportamiento y rendimiento. Se usan pruebas de caja negra.

- **Prueba de alto nivel:** queda fuera de los límites de la ingeniería del software, entrando en el más amplio contexto de la ingeniería de sistemas de Computadora El software, una vez validado, se debe combinar con otros elementos del sistema

- **Prueba de sistema:** Verifica que todo encaja de forma adecuada y que se alcanzan la funcionalidad y rendimiento total

10) Enuncie los puntos propuestos por Gilb para implementar con éxito una estrategia de prueba de software

1) Especificar los requisitos de manera cuantificable antes de comenzar las pruebas
2) Establecer objetivos de prueba de manera explícita
3) Comprender que usuarios van a manejar el software y desarrollar un perfil para cada categoría de usuario
4) Desarrollar un plan de pruebas que haga hincapié en la prueba de ciclo rápido
5) Construir un software robusto, destinado a probarse a sí mismo
6) Usar revisiones técnicas formales efectivas como filtro antes de la prueba
7) Llevar a cabo revisiones técnicas formales para evaluar la estrategia de prueba y los propios casos de prueba
8) Desarrollar un enfoque de mejora continua al proceso de prueba, y medir la estrategia de prueba

11) Enuncie las distintas pruebas de unidad

- Se prueba la interfaz del módulo para asegurar que la información fluye de forma adecuada hacia y desde la unidad de programa.

- Se prueban estructuras de datos locales para asegurar que los datos que se mantienen temporalmente se conservan durante la ejecución del algoritmo.

- Se prueban condiciones límite para asegurar que el modulo funciona bien en los límites establecidos

- Se ejercitan caminos independientes para asegurar que las sentencias se ejecutan al menos una vez, y se ejercitan los caminos de manejos de errores.

12) Defina controlador y resguardo en el contexto de pruebas de unidad

Un controlador no es más que un programa principal que acepta los datos de caso de prueba, los pasa al módulo e imprime resultados, y los resguardos reemplazan a módulos subordinados del componente que está siendo probado, que se encarga de hacer una mínima manipulación de datos, devolver una verificación, y darle de nuevo el control al módulo que lo invocó.

13) Defina el objetivo de la prueba de integración

Es una técnica sistemática para construir la estructura de un programa, mientras se están llevando a cabos pruebas de detección de errores. El objetivo es coger los módulos ya probados en la prueba de unidad y construir una estructura de acuerdo a lo que dicta el diseño.

14) Defina prueba de integración descendente

Planteamiento incremental a la construcción de la estructura de programas, donde se integran los módulos moviéndose hacia abajo por la jerarquía de control, comenzando por el modulo principal, y se van incorporando los módulos subordinados.

15) Enuncie los pasos de la integración ascendente

1) Se usa el módulo de control principal como controlador de prueba, disponiendo de resguardos para los módulos subordinados

2) Se sustituyen los resguardos con módulos reales, dependiendo del enfoque de integración

3) Se llevan a cabo pruebas cada vez que se integra un módulo nuevo

4) Luego de terminar unas pruebas, se remplaza otro resguardo con un módulo real

5) Se hace una prueba de regresión para asegurar que no se han introducido errores nuevos

16) Defina prueba de integración ascendente

En este caso se empieza desde los módulos atómicos (los más bajos del programa), y como los procesos de los módulos requeridos siempre está disponible, no hace falta utilizar resguardos.

17. Enuncie los pasos de una estrategia de integración ascendente.

1) Se combinan los módulos de bajo nivel en grupos que realicen una subfunción especifica

2) Se escribe un controlador para coordinar la entrada y salida de casos de prueba

3) Se prueba el grupo

4) Se eliminan los controladores y se combinan los grupos, moviéndose hacia arriba en la estructura del programa

18. Defina prueba de regresión.

Consiste en ejecutar un subconjunto de pruebas que se llevaron a cabo previamente para asegurar de que los cambios no propagaron efectos colaterales no deseados

19) Enuncie diferentes casos asociados a la prueba de regresión

1) Una muestra representativa de pruebas que ejercite todas las funciones del software

2) Pruebas adicionales que se centren en las funciones del software que probablemente se vean afectadas por el cambio

3) Pruebas que se centran en los componentes del software que han cambiado

20) Defina herramientas de reproducción de captura

Las herramientas de reproducción de captura permiten al ingeniero de software capturar casos de prueba y los resultados para la subsiguiente reproducción y la comparación.

21) Defina prueba de humo

Es un método de prueba de integración que se utiliza cuando se desarrolló un software empaquetado. Permite que el equipo valore el proyecto sobre una base sólida.

22) Enuncie actividades de una prueba de humo

1) Los componentes han sido traducidos y se integran a una construcción. Una construcción incluye ficheros de datos, librerías, módulos reutilizables y componentes de ingeniería que se requieren para implementar una o más funciones

2) Se diseña una serie de pruebas para descubrir errores que impiden a la construcción realizar su función, es decir, errores <bloqueantes>

3) Se integra la construcción con otras construcciones, y se aplica una prueba de humo al producto completo, haciéndose de forma ascendente o descendente

23) Enuncie beneficios de la prueba de humo sobre proyectos de IS complejos y críticos

1) Se minimizan riesgos de integración, ya que se realizan pruebas frecuentemente

2) Se perfecciona la calidad del producto final, ya que la prueba de humo está orientada a la integración

3) Se simplifican el diagnóstico y las correcciones de errores, ya que pueden aparecer errores cada vez que se agrega un módulo

4) El progreso es fácil de observar, ya que cada día que pasa más software se integra, y más funciona

24) Defina que son y en qué momento puede comenzar la prueba de validación

La validación se consigue cuando el software funciona de acuerdo a las expectativas razonables del cliente. Comienza una vez que el software fue ensamblado porque termino la prueba de validación

25) Defina criterios de la prueba de validación

La validación se consigue mediante una serie de pruebas de caja negra que demuestran conformidad con los requisitos. Se hace un plan de prueba con las pruebas a llevar a cabo, y un procedimiento de prueba define los casos de prueba específicos, para descubrir errores de acuerdo con los requisitos. Luego de esto pueden pasar dos cosas:

1) Las características de funcionamiento se dan de acuerdo a las especificaciones
2) Hay una desviación y se crea una lista de deficiencias

26) Defina el propósito de la revisión de la configuración

Es asegurarse de que todos los elementos de la configuración se han desarrollado apropiadamente, catalogado y están suficientemente detallados para soportar la fase de mantenimiento. También se la llama auditoría.

27) Defina pruebas alfa y beta Prueba alfa:

Se lleva a cabo por un cliente en el lugar de desarrollo. Se usa el software de forma natural con el desarrollador como observador, quien registra errores y problemas de uso.

- **Pruebas alfa** se hacen en un entorno controlado.

- **Prueba beta**: Se lleva a cabo por usuarios finales del software, y el desarrollador no está presente. La prueba beta es una aplicación <en vivo> del software, en un entorno que no puede ser controlado. Con los problemas informados, el desarrollador lleva a cabo las modificaciones necesarias.

28) Defina pruebas del sistema

Está constituida por una serie de pruebas diferentes cuyo propósito primordial es ejercitar profundamente el sistema basado en computadora. Todas trabajan para verificar que se han integrado adecuadamente los elementos del sistema, y que se realizan las funciones apropiadas

29) Enuncie estrategias de anticipación a problemas de interacción

1) Diseñar caminos de manejo de errores, que prueben toda la información precedente de otros elementos del sistema
2) Llevar a cabo pruebas que simulen la presencia de datos en mal estado u otros errores en la interfaz
3) Registrar resultados de pruebas como evidencia en caso de que se señale con el dedo
4) Participar en la planificación y el diseño de pruebas del sistema, para asegurarse de que el software se prueba de forma adecuada

30. Defina prueba de recuperación, prueba de seguridad, prueba de resistencia (stress), prueba de sensibilidad y pruebas de rendimiento.

- **Prueba de recuperación:** Fuerza el fallo del software de muchas formas, y verifica que la recuperación sea adecuada. Si esta es automática, se evalúa correcta inicialización de los mecanismos de recuperación.

- **Prueba de seguridad:** Intenta verifica que los mecanismos de seguridad implementados lo defienden de accesos impropios. El responsable de prueba intenta entrar por cualquier método, bloquear el sistema, negando el servicio a otras personas, etc.

- **Prueba de resistencia o stress:** Ejecuta un sistema que demande recursos en cantidad, por ejemplo, diseñar pruebas que generen diez interrupciones por segundo, incrementar la frecuencia de datos de entrada, para comprobar cómo responden las funciones de entrada, diseñar casos de prueba que puedan dar problemas en un sistema operativo virtual, o diseñar casos de prueba que produzcan excesivas búsquedas de datos del disco.

- **Prueba de sensibilidad:** Intenta descubrir combinaciones de datos dentro de una clase de entrada valida que pueda producir inestabilidad o un proceso incorrecto.

- **Prueba de rendimiento:** Es para probar el rendimiento del software en un tiempo de ejecución dentro del contexto de un sistema integrado, y se lleva a cabo durante todos los pasos del proceso de prueba.

31) Enuncie características de errores que son identificados durante el proceso de depuración

1) El síntoma puede desaparecer temporalmente al corregir otro error

2) El síntoma puede realmente estar producido por algo que no es un error

3) El síntoma y la causa pueden ser remotos entre sí, osea que uno esté en una parte, y la causa en otra más alejada

4) El síntoma puede estar causado por un error humano que no sea de fácil detección

5) El síntoma puede ser el resultado de problemas de temporización y no de proceso

6) Puede ser difícil reproducir las condiciones de entrada

7) El síntoma puede aparecer de forma intermitente (suele pasar en sistemas empotrados)

8) El síntoma puede ser por causas distribuidas por una serie de tareas en diferentes procesadores