PRUEBAS DE SISTEMAS DE SOFTWARE

AUTORA: HELLEN CUBERO LEDEZMA

NOVIEMBRE: 2020

San Marcos

Introducción

Las pruebas son el proceso de encontrar diferencias entre el comportamiento esperado, especificado por los modelos del sistema, y el comportamiento observado del sistema. (Dutoit, 2002). y se consideran parte de un proceso más amplio de verificación y validación del software. (Sommerville, 2011).

Las pruebas de validación responden a la consulta ¿se construyó el producto correcto?), mientras que las pruebas de verificación responden a la consulta ¿se construyó bien el producto?.

Existen diferentes tipos de pruebas, como las de rendimiento, la de sistemas, las de desarrollo, las de usuario, sin embargo en la lectura se tratarán solo algunos aspectos básicos del amplio tema de pruebas de software.

Contenido

Estrategias de prueba

Una estrategia para pruebas de software es la integración de técnicas de diseño de casos de pruebas en una serie de pasos bien planificados que llevan a la evaluación correcta del software. (EcuRED, s.f)

Además, EcuRED (s.f), considera que la estrategia define:

- Técnicas de pruebas (manual o automática) y herramientas a ser usadas.
- Qué criterios de éxito y culminación de la prueba serán usados.
- Consideraciones especiales afectadas por requerimientos de recursos o que tengan implicaciones en la planificación.

Y expone las siguientes recomendaciones:

- Cada caso de prueba debe definir el resultado de salida esperado para comprarlo con el resultado obtenido.
- El programador debe evitar probar sus propios programas, siempre buscará demostrar que funcionan correctamente (ya sea de manera consciente e inconsciente).
- Inspeccionar a conciencia el resultado de cada prueba, para descubrir posibles síntomas de defectos.
- Al generar los casos de prueba, se deben incluir tanto datos de entrada válidos como no válidos.
- Considerar el tiempo necesario para realizar las pruebas, no suponer que no hay defectos en los programas.
- La experiencia parece indicar que donde hay un defecto hay otros, es decir la probabilidad de descubrir nuevos defectos en una parte del software es proporcional al número de defectos ya descubierto.
- Las pruebas son una tarea igual o más creativa que el desarrollo de software. No deben considerarse como una tarea destructiva y rutinaria.

.

OBSERVE LAS LECTURAS DE LOS AUTORES SOMMERVILLE Y DUTOIT PARA OBTENER MÁS INFORMACIÓN.

Pruebas de unidad

Las pruebas de unidad son el proceso de probar componentes del programa, como métodos o clases de objetos. Las funciones o los métodos individuales son el tipo más simple de componente.

Cuando se pone a prueba las clases de objetos, tiene que diseñar las pruebas para brindar cobertura a todas las características del objeto. Esto significa que debe:

- probar todas las operaciones asociadas con el objeto;
- establecer y verificar el valor de todos los atributos relacionados con el objeto;
- poner el objeto en todos los estados posibles. Esto quiere decir que tiene que simular todos los eventos posibles. (Sommerville, 2011, p.194)

Para obtener más información visite la sección 8.1.1 del siguiente enlace

Pruebas de integración

Las pruebas de integración hacen referencias a las pruebas que se realizan para detectar defectos al integrar varios componentes.

Detectan defectos que no se han descubierto durante las pruebas unitarias enfocándose en pequeños grupos de componentes. Dos o más componentes se integran y prueban, y después de que las pruebas ya no detectan ningún nuevo defecto se añaden componentes adicionales. (Dutoit, 2002, p.191).

Para obtener más información visite el siguiente enlace

Pruebas de validación

En las pruebas de validación se espera que el sistema se desempeñe de manera correcta mediante el conjunto dado de casos de prueba, que refleje el uso previsto del sistema. (Sommerville, 2011,p.224).
Se descubren defectos en el sistema.
Los procesos de verificación y validación son indispensables en la producción de sistemas, como lo menciona Sommerville (2011), buscan comprobar que el software por desarrollar cumpla con sus especificaciones, y brinde la funcionalidad desea por las personas que pagan por el software.
La finalidad de la verificación es comprobar que el software cumpla con su funcionalidad y con los requerimientos no funcionales establecidos. Sin embargo, la validación es un proceso más general. La meta de la validación es garantizar que el software cumpla con las expectativas del cliente. La validación es esencial, porque las especificaciones de requerimientos no siempre reflejan los deseos o necesidades reales de los clientes y usuarios del sistema.

Métodos de prueba

Los métodos de prueba orientada a objetos es una secuencia de operaciones para probar los estados de la clase.

Según Blanco (s.f), los métodos se presentan a nivel de clase e interclase, los cuales consisten en:

Nivel de clase

Se realiza verificación al azar, que consiste en probar en orden aleatorio diferentes secuencias válidas de operaciones. Por ejemplo, en un sistema bancario, un orden aleatorio de operaciones sería abrir - configurar-depositar-retirar-cerrar.

También se realizan pruebas de partición basada en:

- **Estados:** clasifica las operaciones de clase en base a su habilidad de cambiar el estado de la clase. Por ejemplo, las operaciones depositar o retirar cambian de estado, las de consultas de saldo no.
- **Atributos:** clasifica las operaciones basadas en los atributos que usa. por ejemplo, las operaciones que hace uso del atributo "LimiteCredito" y las que no.

Categorías: clasifica las operaciones de acuerdo a la función genérica que llevan a cabo. Por ejemplo, operaciones de inicialización (abrir, configurar), operaciones de consulta (saldo, resumen).

Nivel entre clases

Es necesario verificar las colaboraciones entre clases.
Al igual que a nivel de clase, se realiza técnica de selección al **azar** y técnica de **partición**, pero adicionalmente se realizan:

Técnica basada en el **escenario**.

- Para cada clase cliente, generar secuencias de operaciones al azar.
- Para cada mensaje determinar la clase colaboradora y la operación correspondiente en el servidor.
- Para cada operación invocada por el servidor determinar los mensajes que transmite.
- Para cada mensaje determinar el siguiente nivel de operaciones invocadas e incorporarlas a la secuencia de pruebas.

Técnica basada en los **modelos de comportamiento**

- Los diagramas de transición de estados pueden derivar una secuencia de pruebas.
- Se persigue alcanzar una cobertura total de estados.
- La secuencia de operaciones debe causar que la clase haga transiciones por todos los estados. Por ejemplo, abrir-preparar cuenta-depositar-retiro-cerrar.

Conclusiones y recomendaciones

Las pruebas de software son tan importantes como cualquier otra etapa del ciclo de vida de la producción de software, debe planificarse y gestionarse con el tiempo requerido para finalizar de manera correcta todas las etapas del proceso de pruebas (planificación, diseño, implementación y evaluación).

Las pruebas solo pueden mostrar la presencia de errores en un programa, sin embargo, no pueden garantizar que no surjan fallos posteriores. (Sommerville, 2011).

A modo de recomendación, observe las lecturas indicadas en cada sección, de esta manera podrá comprender varios tipos de pruebas que no se han mencionado, pero que son utilizadas en el ámbito de gestión de proyectos de software.

Referencias bibliográficas

Dutoit, A. H. y H. Dutoit, A. (2002). Ingeniería de software orientado a objetos. Pearson Educación. https://elibro.net/es/ereader/usanmarcos/74078?page=351

Sommerville, I. (2011). Ingeniería de software (9a. ed.). Pearson Educación. https://elibro.net/es/ereader/usanmarcos/37857?page=01

Blanco, C. (s.f). Construcción y pruebas de software. Recuperado de https://ocw.unican.es/pluginfile.php/1408/course/section/1803/tema1-pruebasSistemasSoftware.pdf.

EcuRED (s.f). Estrategia de pruebas de software. Recuperado de https://www.ecured.cu/Estrategia_de_pruebas_de_software

San Marcos

www.usanmarcos.ac.cr

San José, Costa Rica