UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS
FACULTAD DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
EAP. DE INGENIERÍA DE SISTEMAS

Estándares de calidad para pruebas de software

4. PRUEBAS DE SOFTWARE.

TESIS para optar el Título Profesional de: INGENIERO DE SISTEMAS
AUTORES
Daniel Rolando Valdivia Espinoza
Eduardo Geonias Valdivia Espinoza
ASESOR: Jorge Díaz Muñante
LIMA- PERÚ 2005

4. PRUEBAS DE SOFTWARE.

4.1. Conceptos

Las pruebas de software son un conjunto de herramientas, técnicas y métodos que evalúan la excelencia y el desempeño de un software. Las técnicas para encontrar problemas en un software son extensamente variadas y van desde el uso del ingenio por parte del personal ejecutor de las pruebas hasta herramientas automatizadas que ayudan a aliviar el peso y el costo de tiempo de esta actividad. Pero de nada serviría conocer todas las técnicas de prueba de software, si un programa carece de documentación, el código es confuso, o no se han seguido los pasos para su planificación y desarrollo. El diccionario del IEEE define a las pruebas, caso de prueba y fallo de la siguiente manera:

4.1.1. Pruebas

"*Es una actividad en la cual un sistema o uno de sus componentes se ejecuta en dos o más circunstancias previamente especificadas, los resultados se observan y registran y se realiza una evaluación de algún aspecto*" [IEEE, 1990]. Probar, por lo tanto, es el proceso de ejecutar un programa con el fin de encontrar errores o fallas.

4.1.2. Caso de prueba

Es un conjunto de entradas, condiciones de ejecución y resultados esperados desarrollados para un objetivo particular como, por ejemplo,

ejercitar un flujo de un programa o verificar el cumplimiento de un determinado requisito. También se puede "*referir a la documentación en la que se describen las entradas, condiciones y salidas de un caso de prueba*" [IEEE, 1990].

4.1.3. Fallo

"*Un fallo es la incapacidad de un sistema o de alguno de sus componentes para realizar las funciones requeridas dentro de los requisitos de rendimiento especificados*" [IEEE, 1990].

4.2. Procesos de Pruebas

En la actualidad la tendencia es iniciar los procesos de pruebas antes y dentro del proyecto, los especialistas responsables de esta actividad deben estar capacitados adecuadamente para cumplirla a cabalidad.

4.2.1. Iniciarlas antes y dentro del proyecto

Quiere decir que actualmente las especificaciones de pruebas se realizan al mismo tiempo que el diseño de software; lo que se propone es iniciar el análisis de las pruebas junto con el análisis del software, estos sondeos preventivos permitirán ejecutar las pruebas tan pronto como el software esté listo y con ello no sólo descubrir errores, sino evitarlos.

4.2.2. Capacitar a los especialistas responsables de las pruebas

Este punto se enfoca en crear conciencia acerca de la importancia de las pruebas y de la importancia que tiene un equipo de personas

específicamente dedicadas a esta actividad que puedan integrarse a un proyecto y sean responsables de su calidad.

Los objetivos actuales de las pruebas no sólo tienen que ver con corregir errores, sino con prevenirlos influyendo y controlando el diseño y desarrollo del software. Las pruebas deben ser modeladas y basadas en los requerimientos de la aplicación que se ha de construir; por tanto, en las especificaciones de software deben incluirse las especificaciones de pruebas, ambas deberán revisarse en conjunto, y en esta revisión deberá participar un especialista en pruebas.

Por otro lado, se debe reconocer que las pruebas son una especie de administrador de riesgos pues aunque se puede definir qué debe considerarse como un buen resultado, quizás lo obtenido no necesariamente sea el mejor.

Hoy en día se calcula que la fase o proceso de pruebas representa más de la mitad del costo de un programa, ya que requiere un tiempo similar al de la programación lo que obviamente acarrea un alto costo económico, de este modo si el proceso de pruebas requiere mucho más que tiempo y dinero entonces necesita una verdadera metodología la cual exige herramientas y conocimientos destinados a optimizar esta tarea.

4.3. Principios Básicos de las Pruebas de Software

Los principios básicos de pruebas de software a menudo parecen obvios pero por lo general son siempre ignorados, lo cual genera serias consecuencias en el proceso. Entre los principios básicos de pruebas de software tenemos los siguientes:

a. ***La definición del resultado esperado a la salida del programa es una parte integrante y necesaria de un caso de prueba*;** muchos errores se originan por ignorar este principio. Si el resultado esperado

de un caso de prueba no ha sido predefinido, existe la posibilidad de que un resultado erróneo, se interprete como correcto.

b. ***Un programador debe evitar probar su propio programa;*** la detección de errores es un proceso "destructivo". Es por ello que es difícil para una persona adoptar la actitud mental necesaria para demostrar que lo que acaba de hacer está erróneo.

c. ***Un área de programación no debería probar sus propios programas;*** es el caso similar al anterior, aplicado a la conducta colectiva de un grupo.

d. ***Inspeccionar concienzudamente el resultado de cada prueba****;* este es probablemente el más obvio, pero, sin embargo, a menudo dejado de lado. Empíricamente se ha encontrado que muchos de los programadores fracasan en la detección de errores, aún cuando en los resultados de las pruebas eran claramente observables.

e. ***Examinar un programa para comprobar que no hace lo que se supone que debe hacer es sólo la mitad del problema;*** la otra mitad consiste en ver si el programa hace lo que no se supone que debe hacer. Esto significa que los programas deben ser revisados con respecto a efectos colaterales no deseados.

f. ***Evitar los casos de pruebas desechables a menos que el programa sea verdaderamente un programa desechable;*** una práctica muy frecuente es sentarse frente al computador e inventar casos e pruebas. Los casos de prueba deben ser reinventados con cada prueba que se realice, algo que generalmente no se hace, por lo que las pruebas siguientes no son tan rigurosas como la primera vez, esto significa que si se introdujo un error en una parte ya probada, rara vez se detecta.

g. ***No planear el esfuerzo con la suposición tácita de que no se encontrarán errores*;** este error se comete cuando se parte de la suposición de que la prueba es el proceso de mostrar que el programa funciona correctamente.

h. ***La probabilidad de encontrar errores adicionales en una sección del programa es proporcional al número de errores encontrados*;** este fenómeno tiene su base en comprobaciones empíricas, en efecto, los módulos de un programa (igualmente probados) que han presentado más errores tienen una probabilidad más alta de presentar otros errores.

i. ***La prueba de software no es una ciencia exacta*;** si bien existen métodos que ayudan en el proceso de la elaboración de casos de pruebas, ello requiere de todas maneras de una considerable cuota de creatividad.

4.4. Tipos de Pruebas de Software

Las pruebas de software se dividen en dos grupos: las *Pruebas Estáticas* (sin uso de computadoras), son básicamente manuales, utilizadas para la revisión del código y planeamiento de los objetivos y fases de las pruebas; y las *pruebas Dinámicas* (con uso de computadoras), que permiten probar aspectos como la funcionalidad, integración, resistencia al uso concurrente, etc. En la Figura Nº 1 se muestra una estructura jerárquica de los tipos de pruebas de software, a continuación se detallan los objetivos y actividades de cada uno de los tipos de pruebas.

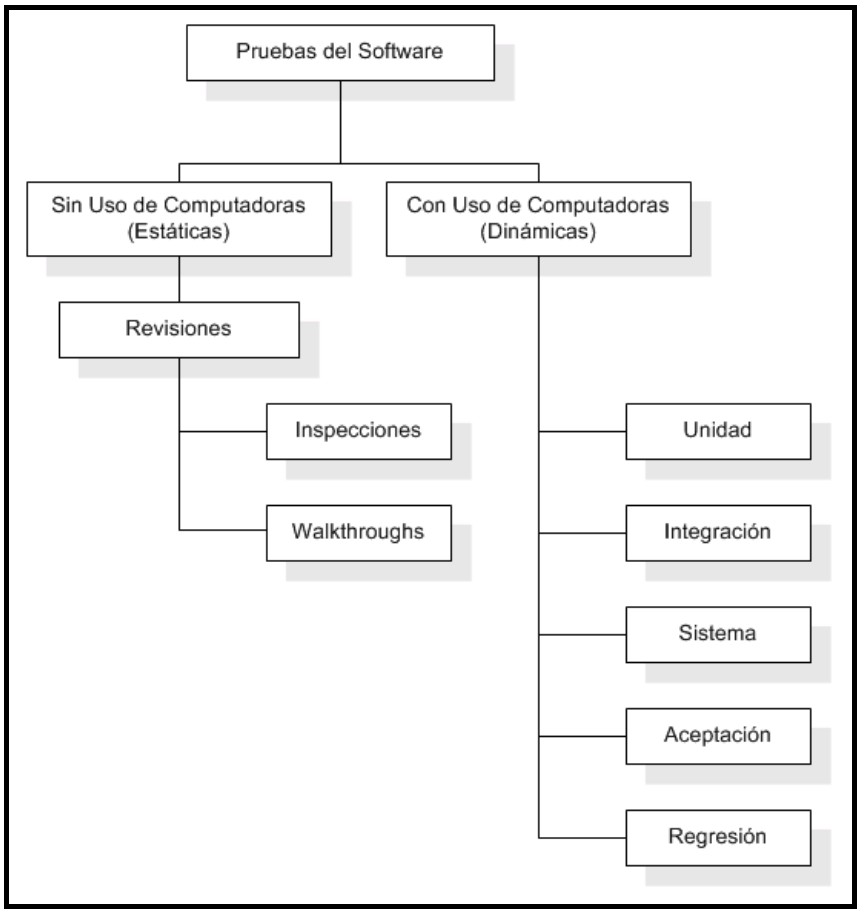

Figura Nº 1: Tipos de Pruebas de Software

4.4.1. Pruebas sin Uso de Computadoras o Estáticas

4.4.1.1. Inspecciones

Una inspección formal es un proceso con objetivos bien definidos, y como todo proceso, está conformado por fases, dichas fases son las siguientes:

a. ***Planificación*;** se determina si el producto cumple los criterios para ser inspeccionado, se selecciona el equipo de trabajo, se asignan los roles, se distribuye el material, se coordina fecha, hora y lugar de la reunión de inspección.

b. ***Orientación inicial*;** es una fase opcional. El equipo se familiariza con el tema en cuestión.

c. ***Preparación individual*;** cada miembro del equipo analiza el producto, detectando potenciales defectos que serán luego discutidos.

d. ***Reunión de inspección*;** el equipo revisa el producto para detectar, categorizar y registrar, pero no corregir, los defectos.

e. ***Corrección*;** se corrigen los defectos detectados en el producto.

f. ***Reinspección*;** el proceso recomienza cuando se detectaron gran cantidad de defectos.

g. ***Seguimiento*;** Se determina si los defectos detectados han sido corregidos y se verifica que no se han introducido nuevos defectos.

4.4.1.2. Walkthroughs

Es una técnica más aplicada en control de calidad que en pruebas, consiste en sentar alrededor de una mesa a los desarrolladores y a una serie de críticos, bajo las órdenes de un moderador. El método consiste en que los críticos leen el programa línea a línea y piden explicaciones de todo lo que no está suficientemente claro. Puede ser que simplemente falte un comentario explicativo, o que detecten un error auténtico o que simplemente el código sea tan complejo de entender o explicar que más vale que se rehaga de forma más simple. Para un sistema complejo pueden hacer falta muchas sesiones.

Esta técnica es muy eficaz para encontrar errores de naturaleza local pero falla estrepitosamente cuando el error deriva de la interacción entre dos partes alejadas del programa. Nótese que no se está ejecutando

el programa, sólo se hace un análisis como con una lupa, y de esta forma sólo se ve en cada instante un parte del listado del código fuente.

4.4.2. Pruebas con uso de Computadoras o Dinámicas

4.4.2.1. Pruebas de Unidad

Se focaliza en ejecutar cada módulo (o unidad mínima a ser probada, ejemplo: una clase) lo que provee un mejor modo de manejar la integración de las unidades en componentes mayores. Busca asegurar que el código funciona de acuerdo con las especificaciones y que el módulo lógico es válido.

Características:

- Particionar los módulos en unidades lógicas fáciles de probar, por cada unidad hay que definir los casos de prueba (pruebas de caja blanca).
- Los casos de prueba deben diseñarse de forma tal que se recorran todos los caminos de ejecución posibles dentro del código bajo prueba; por lo tanto el diseñador debe construirlos con acceso al código fuente de la unidad a probar.

- Los aspectos a considerar son los siguientes: rutinas de excepción, rutinas de error, manejo de parámetros, validaciones, valores aceptados, valores límites, rangos, mensajes posibles.

Técnica:

- Comparar el resultado esperado con el resultado obtenido.
- Si existen errores, reportarlos.

4.4.2.2. Pruebas de Integración

Se focaliza en verificar que las interfaces entre las componentes de software funcionan correctamente, así como determinar el enfoque para avanzar desde un nivel de integración de los componentes al siguiente y las acciones a tomar cuando se descubren problemas

Características:

- Identificar errores introducidos por la combinación de programas probados unitariamente.
- Determina cómo la base de datos de prueba será cargada.
- Verificar que las interfaces entre las entidades externas (usuarios) y las aplicaciones funcionan correctamente. Verificar que las especificaciones de diseño sean alcanzadas.
- Determina el enfoque para avanzar desde un nivel de integración de los componentes al siguiente.
- Describe cómo verificar que las interfaces entre los componentes de software funcionan correctamente.
- Decide qué acciones tomar cuando se descubren problemas.

Técnica:

- Utilizar la técnica top-down. Se empieza con los módulos de nivel superior, y se verifica que los módulos de nivel superior llaman a los de nivel inferior de manera correcta, con los parámetros correctos.

- Utilizar la técnica down-top. Se empieza con los módulos de nivel inferior, y se verifica que los módulos de nivel inferior llaman a los de nivel superior de manera correcta, con los parámetros correctos.

4.4.2.3. Pruebas de Sistema

Las pruebas del sistema deben enfocarse en requisitos que puedan ser tomados directamente de casos de uso, reglas y lógica del negocio. El objetivo de estas pruebas es verificar el ingreso, procesamiento y recuperación apropiado de datos, y la implementación apropiada de las reglas de negocio. Este tipo de pruebas se basan en técnicas de caja negra, esto es, verificar el sistema (y sus procesos internos), la interacción con las aplicaciones que lo usan vía GUI y analizar las salidas o resultados. Esencialmente, el encargado de la prueba intenta hacer fallar el programa. Algunos tipos de pruebas de sistema importantes para sistemas basados en software son:

a. ***Prueba de recuperación*;** es una prueba del sistema que obliga al fallo del software de muchas formas y verifica que la recuperación se lleva a cabo apropiadamente. Si la recuperación es automática hay que evaluar la corrección de la reinicialización, de los mecanismos de recuperación de datos y del rearranque. Si la recuperación requiere la intervención humana, hay que evaluar los tiempos medios de reparación para determinar si están dentro de los límites aceptables.

b. ***Prueba de seguridad*;** intenta verificar que los mecanismos de protección incorporados en el sistema lo protegerán de accesos inapropiados.

c. ***Prueba de resistencia*;** consiste en realizar pruebas que demanden recursos en cantidad, frecuencia o volúmenes anormales. Esto se logra a través de:

- Casos de prueba que requieran el máximo de memoria o de otros recursos.
- Casos de prueba que puedan dar problemas con el esquema de gestión de memoria virtual.
- Casos de prueba que produzcan excesivas búsquedas de datos residentes en disco.

d. **Prueba de rendimiento.** Esta diseñada para probar el rendimiento del software en tiempo de ejecución dentro del contexto de un sistema integrado. Esta prueba se da durante todos los pasos del proceso de prueba, incluso al nivel de unidad, se debe asegurar el rendimiento de los módulos individuales a medida que se realizan las pruebas de caja blanca.

Técnica:

- Ejecutar cada caso de prueba, flujo básico o función utilizando datos válidos e inválidos, para verificar que:

  - Los resultados esperados ocurren cuando se utiliza un dato válido.
  - Los mensajes de error o de advertencia aparecen en el momento adecuado, cuando se utiliza un dato inválido.
  - Cada regla de negocios es aplicada adecuadamente.

4.4.2.4. Prueba de Aceptación

La prueba de aceptación es ejecutada antes de que la aplicación sea instalada dentro de un ambiente de producción. La prueba de aceptación es generalmente desarrollada y ejecutada por el cliente o un especialista de la aplicación y es conducida a determinar como el sistema satisface sus criterios de aceptación validando los requisitos que han sido levantados para el desarrollo, incluyendo la documentación y procesos de negocio. Basado en esta prueba el cliente determina si acepta o rechaza el sistema.

Estas pruebas están destinadas a probar que el producto está listo para el uso operativo, suelen ser un subconjunto de las Pruebas de Sistema y sirven para que el usuario pueda validar si el producto final se ajusta a los requisitos fijados, es decir, si el producto está listo para ser implantado para el uso operativo en el entorno del usuario.

Técnica:

- Realización de los documentos de planes de prueba de aceptación y especificación de los mismos, basados en los criterios de aceptación del cliente.

- Los casos prueba de aceptación han de ser planificados, organizados y formalizados de manera que se determine el cumplimiento de los requisitos del sistema. Para la realización de estas pruebas se necesita disponer de los siguientes documentos:

    - Especificación de requisitos del sistema.
    - Manual de usuario.
    - Manual de administrador.

4.4.2.5. Prueba de Regresión

En esta prueba se vuelve a probar el sistema a la luz de los cambios realizados durante el mantenimiento o desarrollo de la nueva versión del sistema buscando efectos adversos en otras partes.

Técnica:

- Realizar una nueva corrida de casos de prueba previos.

- Se requiere de políticas para ser creada la prueba de regresión y decidir qué casos de prueba incluir, para probar eficientemente.

- La prueba de viejas funcionalidades es más importante que la de nuevas funcionalidades.

- Aquellos casos de uso (y los casos de prueba asociados) que descubren defectos tempranamente deben ser incluidos en la prueba de regresión.