2 Prueba a través del ciclo de vida del software

Objetivos de aprendizaje para probar a través del ciclo de vida del software

Los objetivos identifican lo que podrá hacer después de la finalización de cada módulo.

2.1 Modelos del desarrollo del software

- ✓ Entender la relación entre desarrollo, actividades de prueba y productos de trabajo en el ciclo de vida del desarrollo y dar ejemplos basados en las características de proyecto y producto y en el contexto.
- ✓ Reconocer el hecho que los modelos de desarrollo de software deben ser adaptados al contexto del proyecto y a las características del producto.
- ✓ Recordar las razones para diferentes niveles de prueba y características de buenas pruebas en cualquier modelo de ciclo de vida.

2.2 Niveles de prueba

- ✓ Comparar los diferentes niveles de prueba: objetivos principales, típicos objetos de prueba, típicas metas de prueba (por ejemplo, funcionales o estructurales) y productos de trabajo relacionados, gente que prueba, tipos de defectos y fallas a ser identificados.

2.3 Tipos de pruebas: los objetivos de la prueba

- ✓ Comparar cuatro tipos de prueba de software (funcionales, no funcionales, estructurales y relacionados al cambio) por ejemplo.
- ✓ Reconocer que pruebas funcionales y estructurales ocurren en cualquier nivel de prueba.
- ✓ Identificar y describir los tipos de prueba no funcionales basados en requisitos no funcionales.
- ✓ Identificar y describir los tipos de prueba basados en el análisis de una estructura o arquitectura del sistema de software.
- ✓ Describir el propósito de la prueba de confirmación y de la prueba de regresión.

2.4 Pruebas de mantenimiento

- ✓ Comprara las pruebas de mantenimiento (probando un sistema existente) para probar una nueva aplicación con respecto a los tipos de prueba, disparadores para pruebas y cantidad de pruebas.
- ✓ Identificar las razones para la prueba de mantenimiento (modificación, migración y retiro).
- ✓ Describir el rol de la prueba de regresión y del análisis de impacto en el mantenimiento.

2.5 Modelos de desarrollo de software

2.5.1 Términos

Comercial/listo para la venta (COTS), modelo de desarrollo incremental, nivel de prueba, validación, verificación, Modelo-V.

2.5.2 Introducción

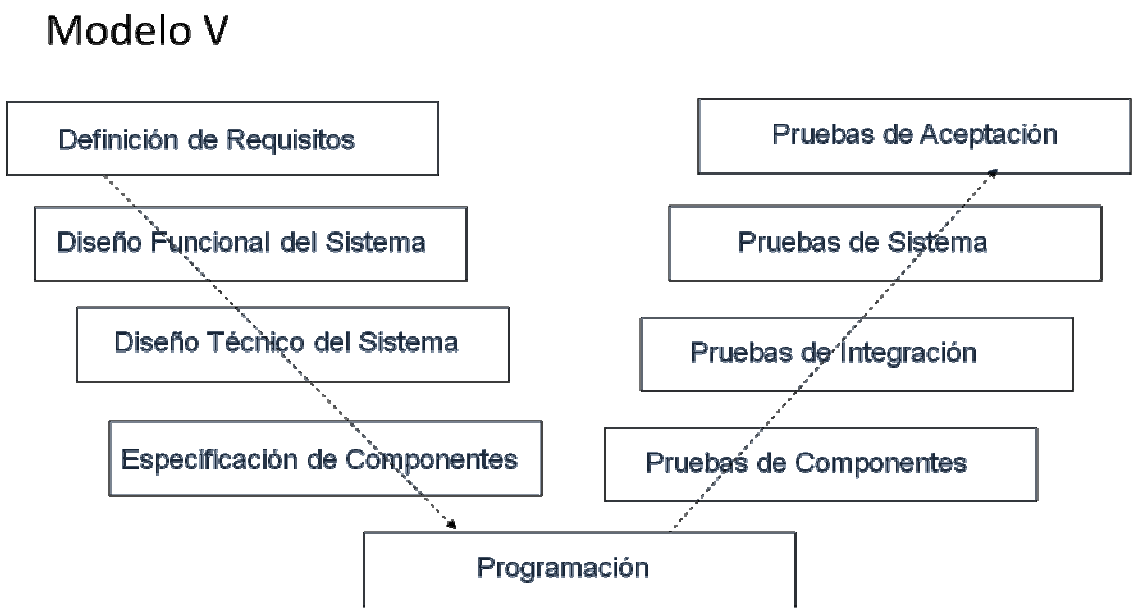
La prueba no existe en forma aislada; las actividades de prueba están relacionadas con las actividades de desarrollo de software. Los diferentes modelos de ciclo de vida de desarrollo necesitan diferentes enfoques hacia la prueba.

2.5.3 Modelo-V

Aunque existen variantes del Modelo-V, un tipo común de Modelo-V usa cuatro niveles de prueba, correspondientes a cuatro niveles de desarrollo.

Los cuatro niveles usados en este programa son:

- ✓ Prueba de componente (unidad).
- ✓ Prueba de integración.
- ✓ Prueba de sistema.
- ✓ Prueba de aceptación.



En la práctica, un Modelo-V puede tener más, pocos o diferentes niveles de desarrollo y prueba, dependiendo del proyecto y del producto de software. Por ejemplo, puede existir prueba de integración de componentes después de las pruebas de componente y prueba de integración del sistema después de la prueba del sistema.

Los productos de trabajo del software (tales como escenarios de negocio o casos de uso, especificaciones de requisito, documentos y código de diseño) producidos durante el desarrollo son a menudo la base de la prueba en uno o más niveles de prueba. Las referencias para los productos de trabajo genéricos incluyen el Modelo de Madurez de las Capacidades Integrado (CMMI) o "Procesos del ciclo de vida del software" (IEEE/IEC 12207). La verificación y validación (y diseño de prueba en forma temprana) pueden ser llevados a cabo durante el desarrollo de los productos de trabajo del software.

2.5.4 Modelos de desarrollo iterativo

El desarrollo iterativo es el proceso de establecer requisitos, diseñar, crear y probar un sistema, realizado como una serie de desarrollos más pequeños. Ejemplos son: la creación de prototipos, el desarrollo rápido de aplicaciones (RAD), el Proceso Unificado Racional (RUP) y los modelos de desarrollo ágiles. El incremento producido por una iteración puede ser probado en diversos niveles como parte de su desarrollo.
Un incremento, aumentado a otros desarrollados previamente, forma un sistema parcial creciente, el cual debería ser probado también. La prueba de regresión es progresivamente importante en todas las iteraciones después de la primera. La verificación y validación pueden ser llevadas a cabo en cada incremento.

2.5.5 Prueba dentro de un modelo de ciclo de vida

En cualquier modelo de ciclo de vida, existen varias características de buena prueba:

- ✓ Para cada actividad de desarrollo existe una actividad de prueba correspondiente.
- ✓ Cada nivel de prueba tiene objetivos de prueba específicos para ese nivel.
- ✓ El análisis y diseño de pruebas para un nivel de prueba dado deberían iniciar durante la actividad de desarrollo correspondiente.
- ✓ Los probadores deberían estar involucrados en la revisión de documentos tan pronto como los borradores estén disponibles en el ciclo de vida del desarrollo.

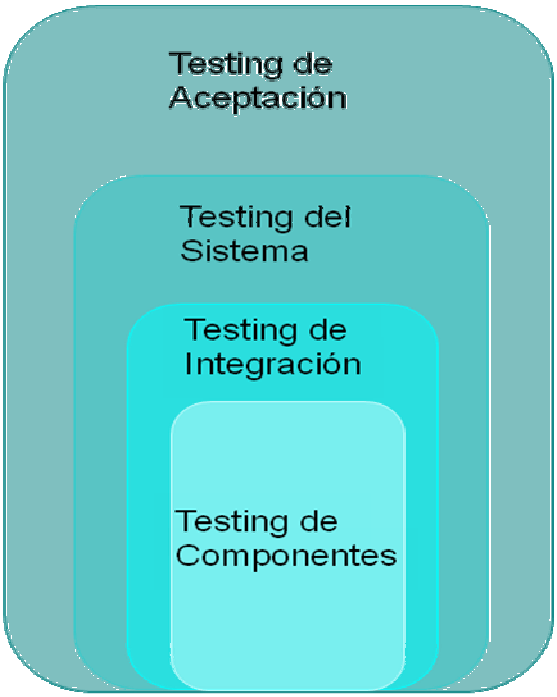
2.6 Niveles de prueba

2.6.1 Términos

Pruebas alfa, pruebas beta, pruebas de componentes (también conocidas como pruebas de unidad, módulo o de programa), pruebas de aceptación de contrato, controladores, pruebas de campo, requisitos funcionales, integración, pruebas de integración, requisitos no funcionales, pruebas operacionales (de aceptación), pruebas de aceptación de regulación, pruebas de robustez, agentes de sustitución, pruebas de sistema, desarrollo guiado por prueba, entorno de prueba, pruebas de aceptación de usuario.

2.6.2 Introducción

Para cada uno de los niveles de prueba, lo siguiente puede ser identificado: sus objetivos genéricos, el(los) producto(s) de trabajo siendo referenciados para derivar casos de prueba (es decir la base de prueba), el objeto de prueba (es decir lo que está siendo probado), defectos y fallas típicas a ser encontradas, requisitos de arnés de prueba y soporte de herramienta y los enfoques y responsabilidades específicos.



Los niveles de prueba pueden ser combinados o reorganizados dependiendo de la naturaleza del proyecto o de la arquitectura del sistema. Por ejemplo, para la integración de un producto de software comercial/listo para la venta (COTS) en un sistema, el comprador podrá realizar la prueba de integración en el nivel del sistema (por ejemplo, integración a la infraestructura o a otros sistemas o despliegue del sistema) y las pruebas de aceptación (funcional y/o no funcional y prueba operacional y/o del usuario).

2.6.3 Pruebas de componente

Las pruebas de componente buscan defectos en el, y verifican el funcionamiento del, software (por ejemplo, módulos, programas, objetos, clases, etc.) que son verificables separadamente. Pueden ser realizadas en forma aislada del resto del sistema, dependiendo del contexto del ciclo de vida de desarrollo y del sistema. Los agentes de sustitución, controladores y simuladores pueden ser usados.

Las pruebas de componente pueden incluir la prueba de funcionalidad y de las características no funcionales específicas, tales como comportamiento de recursos (por ejemplo, pérdidas de

memoria) o pruebas de robustez, así como pruebas estructurales (por ejemplo, cobertura de bifurcación). Los casos de prueba están derivados de los productos de trabajo tales como una especificación del componente, del diseño del software o del modelo de datos.

Típicamente, las pruebas de componente ocurren con acceso al código que está siendo probado y con el soporte del entorno de desarrollo, tales como un marco de trabajo de prueba de unidad o una herramienta de depuración, y, en la práctica, usualmente involucran al programador que escribió el código. Los defectos son típicamente arreglados tan pronto como son encontrados, sin registrar formalmente los incidentes.

Un enfoque en las pruebas de componente es preparar y automatizar los casos de prueba antes de la codificación. Esto se llama enfoque de primera prueba o desarrollo guiado por prueba. Este enfoque es altamente iterativo y está basado en ciclos de desarrollar casos de prueba, luego crear e integrar pequeñas piezas del código y ejecutar las pruebas de componente hasta que éstas pasen.

2.6.3.1 Definición

- ✓ Prueba de cada componente tras su realización/construcción.
- ✓ Los componentes son referidos como módulos, clases o unidades.
- ✓ También denominadas pruebas de Desarrollador (developer's test).

Dadas las convenciones de cada lenguaje de programación para la asignación de nombres a sus respectivos componentes, se podrá hacer referencia a una componente como

- ✓ Prueba de modulo: En C
- ✓ Prueba de Clase: En java o C++
- ✓ Pruebas de unidad: En pascal

2.6.3.2 Alcance

Solo se prueban componentes individuales.
Cada componente es probado de forma independiente.
Los casos de prueba tienen 3 fuentes.

- ✓ Un componente puede estar constituido por un conjunto de unidades más pequeñas.
- ✓ Los objetos de prueba no siempre pueden ser probados en solitario.
- ✓ Descubriendo fallos producidos por defectos internos.
- ✓ Los efectos cruzados entre componentes quedan fuera del alcance de estas pruebas

2.6.4 Pruebas de integración

Las pruebas de integración prueban interfaces entre componentes, interacciones a diferentes partes de un sistema, tales como el sistema operativo, sistema de archivos, hardware o interfaces entre sistemas.

Puede haber más de un nivel de pruebas de integración y éste puede ser llevado a cabo sobre objetos de prueba de diverso tamaño. Por ejemplo:

- ✓ Las pruebas de integración de componentes prueban las interacciones entre los componentes del software y son realizada después de las pruebas de componente.
- ✓ Las pruebas de integración de sistema prueban las interacciones entre los diferentes sistemas y pueden ser realizada después de las pruebas de sistema. En este caso, la organización desarrolladora puede controlar solamente un lado de la interfaz, así los cambios pueden estar desestabilizando. Los procesos de negocio implementados como flujos de trabajo pueden involucrar una serie de sistemas. Los problemas de multi-plataforma pueden ser importantes.

Cuan mayor es el alcance de integración, más difícil se vuelve el aislar las fallas a un componente o sistema específico, lo cual puede llevar a un riesgo incrementado.

Las estrategias de integración sistemáticas pueden estar basadas en la arquitectura del sistema (tales como arriba-abajo o abajo-arriba), tareas funcionales, secuencias de proceso de transacción o algún otro aspecto del sistema o componente. Para reducir el riesgo de descubrimiento de defecto tardío, la integración debería ser normalmente incremental en lugar del tipo “big bang”.

Las pruebas de características no funcionales específicas (por ejemplo, rendimiento) pueden ser incluidas en las pruebas de integración.

En cada fase de integración, los probadores se concentran solamente en la integración en sí. Por ejemplo, si están integrando el módulo A con el módulo B, están interesados en probar la comunicación entre los módulos, no en la funcionalidad de cualquier módulo. Ambos enfoques funcionales y estructurales pueden ser usados.

Idealmente, los probadores deben comprender la planificación de integración de influencia y arquitectura. Si las pruebas de integración son planeadas antes que los componentes o sistemas sean creados, ellos pueden ser creados en el orden requerido para una prueba más eficiente.

2.6.4.1 Definición

La integración es la actividad en la cual se combinan componentes software individual en subsistemas más amplios.
Cada componente ya ha sido probado en lo referente a su funcionalidad interna (prueba de componente).

- ✓ Comprueba la interacción entre elementos de software (componentes) tras la integración del sistema.
- ✓ La integración adicional/subsiguiente de subsistemas también es parte del proceso de integración del sistema.
- ✓ Comprueban las funciones externas.
- ✓ Pueden ser ejecutadas por desarrolladores, testers o ambos.

2.6.4.2 Estrategias

Tipos fundamentales de integración

- ✓ **Integración incremental**. Se combina el siguiente módulo que se debe probar con el conjunto de módulos que ya han sido probados Integración no incremental. Se prueba cada módulo por separado y luego se integran todos de una vez y se prueba el programa completo.

  - ✓ **Ascendente.** Se comienza por los módulos hoja.
    El proceso es el siguiente:

    - o Se combinan los módulos de bajo nivel en grupos que realicen una función o subfunción específica (o quizás si no es necesario, individualmente) -> de este modo reducimos el número de pasos de integración.
    - o Se escribe para cada grupo un módulo impulsor o conductor -> de este modo permitimos simular la llamada a los módulos, introducir datos de prueba y recoger resultados.
    - o Se prueba cada grupo mediante su impulsor.
    - o Se eliminan los módulos impulsores y se sustituyen por los módulos de nivel superior en la jerarquía.

  - ✓ **Descendente.** Se comienza por el módulo raíz.

  - ✓ Comienza por el módulo de control principal y va incorporando módulos subordinados progresivamente.
  - ✓ No hay un orden adecuado de integración, pero unos consejos son los siguientes:
    - o Si hay secciones críticas (especialmente complejas) se deben integrar lo antes posible.
    - o El orden de integración debe incorporar cuanto antes los módulos de entrada/salida para facilitar la ejecución de pruebas.

  - ✓ Existen dos formas básicas de hacer esta integración:

    - o Primero en profundidad: Se van completando ramas del árbol (A B E C F G D)
    - o Primero en anchura: Se van completando niveles de jerarquía (A B C D E F G)

ETAPAS FUNDAMENTALES DE LA INTEGRACION DESCENDENTE

- o El módulo raíz es el primero: Se escriben módulos ficticios que simulan los subordinados.
- o Una vez probado el módulo raíz (sin detectarse ya ningún defecto), se sustituye uno de los subordinados ficticios por el módulo correspondiente según el orden elegido.
- o Se ejecutan las correspondientes pruebas cada vez que se incorpora un módulo nuevo.
- o Al terminar cada prueba, se sustituye un ficticio por su correspondiente real.
- o Conviene repetir algunos casos de prueba de ejecuciones anteriores para asegurarse de que no se ha introducido ningún defecto nuevo.

- ✓ **Integración no incremental.** Se prueba cada módulo por separado y luego se integran todos de una vez y se prueba el programa completo.

COMPARACION DE LOS DISTINTOS TIPOS DE INTEGRACION PRUEBAS DEL SOFTWARE

- ✓ **Ventajas de la no incremental:**
  - o Requiere menos tiempo de máquina para las pruebas, ya que se prueba de una sola vez la combinación de los módulos.
  - o Existen más oportunidades de probar módulos en paralelo.

- ✓ **Ventajas de la incremental:**
  - o Requiere menos trabajo, ya que se escriben menos módulos impulsores y ficticios
  - o Los defectos y errores en las interfaces se detectan antes, ya que se empieza antes a probar las uniones entre los módulos.
  - o La depuración es mucho más fácil, ya que si se detectan los síntomas de un defecto en un paso de la integración hay que atribuirlo muy probablemente al último módulo incorporado.
  - o Se examina con mayor detalle el programa, al ir comprobando cada interfaz poco a poco.

VENTAJAS Y DESVENTAJAS DE LAS INTEGRACIONES ASCENDENTE Y DESCENDENTE

Ascendente	**Descendente**
Ventajas	
- Es un método ventajoso si aparecen Grandes fallos en la parte inferior del programa, ya que se prueba antes. - La entradas para las pruebas son más módulos inferiores tienen funciones. - Es más fácil observar los resultados de la prueba, ya que es en los módulos inferiores donde se elaboran los datos (los superiores suelen ser módulos de control).	- Es ventajosa si aparecen grandes defectos en los niveles superiores del programa, ya que se prueban antes. - Una vez incorporadas las funciones de entrada/salida, es fácil manejar los casos de prueba. - Permite ver antes una estructura previa del programa, lo que facilita el hacer demostraciones y ayuda a mantener la moral.
Desventajas	**Desventajas**
Se requieren módulos impulsores, que deben codificarse. • El programa, como entidad, sólo aparece cuando se agrega el último módulo.	- Se requieren módulos ficticios que suelen ser complejos de crear. - Antes de incorporar la entrada/salida resulta complicado el manejo de los casos de prueba. - Las entradas para las pruebas pueden ser difíciles o imposibles de crear, puesto que, a menudo, se carece de los módulos inferiores que proporcionan los detalles de operación. - Es más difícil observar la salida, ya

	que los resultados surgen de los módulos inferiores. • **Pueden inducir a diferir la terminación de la prueba de ciertos módulos.**

- ✓ **Ad-hoc:** Los componentes serán probados, si fuera posible, inmediatamente después de haber sido finalizada su construcción y se hayan finalizado las pruebas de componente.

2.6.4.3 Consideración al momento de la elección

La estrategia de integración determina la magnitud del esfuerzo requerido para las pruebas
La finalización de la construcción de componentes determina, para todos los tipos de estrategias, el intervalo temporal en el cual el componente estará disponible.
La estrategia de desarrollo influye en la estrategia de integración.

2.6.5 Pruebas de sistema

Las pruebas de sistema están relacionadas con el comportamiento de un sistema/producto completo como está definido por el alcance de un proyecto o programa de desarrollo.

En las pruebas de sistema, el entorno de prueba debe corresponder al objetivo o entorno de producción final tanto como sea posible para minimizar el riesgo de fallas de entorno específico que no están siendo encontradas en la prueba.

Las pruebas de sistema pueden incluir pruebas basadas en riesgos y/o en especificaciones de requisitos, procesos de negocio, casos de uso, u otras descripciones de alto nivel del comportamiento del sistema, interacciones con el sistema operativo y recursos del sistema.

Las pruebas de sistema deben investigar tanto los requisitos funcionales como los requisitos no funcionales del sistema. Los requisitos pueden existir como texto y/o modelos. Los probadores también necesitan tratar con requisitos incompletos o no documentados. Las pruebas de sistema de requisitos funcionales comienzan al usar las técnicas más apropiadas basadas en especificación (caja negra) para el aspecto del sistema a ser probado. Por ejemplo, una tabla de decisión puede ser creada para combinaciones o efectos descritos en reglas de negocio. Las técnicas basadas en estructura (caja blanca) pueden ser usadas para evaluar la minuciosidad de la prueba con respecto al elemento estructural, tales como estructura del menú o navegación de página web.

Un equipo de prueba independiente a menudo lleva a cabo pruebas de sistema.

2.6.5.1 Definición

Tienen que ver con el comportamiento de todo un sistema/producto, definida por el alcance de un proyecto de desarrollo o programa.

Los casos de prueba pueden ser obtenidos de:

- ✓ Evaluación de Riesgos
- ✓ Especificaciones funcionales
- ✓ Casos de Uso
- ✓ Procesos de Negocios

2.6.5.2 Enfoque para las pruebas funcionales

Basadas en Requisitos

Los casos de prueba se derivan de la especificación de requisitos.
El número de casos de prueba variará en función del tipo/profundidad de las pruebas basadas en la especificación de requisitos.

Basadas en procesos de Negocio

Cada proceso de negocio sirve como fuente para derivar/generar pruebas.
El orden de relevancia de los procesos de negocio puede ser aplicado para asignar prioridades a los casos de prueba.

Basadas en Casos de Uso

Los casos de prueba se derivan/generan a partir de las secuencias de usos esperados o razonables
Las secuencias utilizadas con mayor frecuencia reciben una prioridad más alta.

2.6.6 Pruebas de aceptación

Las pruebas de aceptación son a menudo la responsabilidad de los clientes o usuarios de un sistema; otras partes interesadas pueden estar involucradas también.

La meta en las pruebas de aceptación es establecer la confianza en el sistema, las partes del sistema o las características no funcionales específicas del sistema. Encontrar defectos no es el foco principal en las pruebas de aceptación. Las pruebas de aceptación pueden evaluar la predisposición del sistema para el despliegue y uso, aunque no es necesariamente el nivel final de la prueba. Por ejemplo, una prueba de integración del sistema a gran escala puede venir después de las pruebas de aceptación para un sistema.

Las pruebas de aceptación pueden ocurrir como más que apenas un simple nivel de prueba, por ejemplo:

- ✓ Un producto de software COTS puede ser probado para aceptación cuando es instalado o integrado.
- ✓ Las pruebas de aceptación de la usabilidad de un componente pueden ser realizadas durante las pruebas de componente.
- ✓ Las pruebas de aceptación de una nueva mejora funcional pueden venir antes de las pruebas de sistema.

Las formas típicas de las pruebas de aceptación incluyen lo siguiente:

2.6.6.1 Pruebas de aceptación del usuario

Verifica típicamente la aptitud para el uso del sistema por los usuarios del negocio.

2.6.6.2 Pruebas operacionales (de aceptación)

La aceptación del sistema por los administradores de sistema, incluyendo:

- ✓ Pruebas de respaldo/restauración;
- ✓ Recuperación de desastre;
- ✓ Gestión del usuario;
- ✓ Tareas de mantenimiento;
- ✓ Chequeos periódicos de las vulnerabilidades de seguridad.

2.6.6.3 Pruebas de aceptación de contrato y de regulación

Las pruebas de aceptación de contrato son realizadas contra unos criterios de aceptación del contrato para producir software desarrollado a la medida. Los criterios de aceptación deben ser definidos cuando el contrato es acordado. Las pruebas de aceptación de regulación son realizadas contra cualquier regulación que debe ser adherida, tales como regulaciones gubernamentales, legales o de seguridad.

2.6.6.4 Pruebas alfa y beta (o de campo)

Los desarrolladores de software de mercado, o de COTS, a menudo quieren obtener información de clientes existentes o potenciales en su mercado antes que el producto de software sea puesto a la venta comercialmente. La prueba alfa es realizada en el sitio de la organización desarrolladora. La prueba beta, o prueba de campo, es realizada por la gente en sus propios lugares. Ambas son realizadas por clientes potenciales, no por desarrolladores del producto.

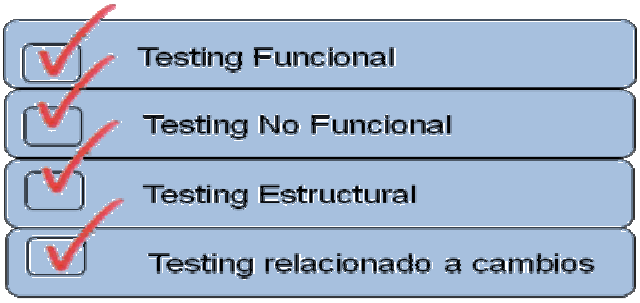
Las organizaciones pueden utilizar otros términos también, tales como pruebas de aceptación de fábrica y pruebas de aceptación del sitio para los sistemas que son probados antes y después de ser movidos al sitio del cliente.

2.7 Tipos de pruebas: los objetivos de la prueba

2.7.1 Términos

Automatización, pruebas de caja negra, cobertura de código, pruebas de confirmación, pruebas funcionales, pruebas de interoperabilidad, pruebas de carga, pruebas de mantenibilidad, pruebas de rendimiento, pruebas de portabilidad, pruebas de regresión, pruebas de fiabilidad, pruebas de

seguridad, pruebas basadas en especificación, pruebas de estrés, pruebas estructurales, conjunto de pruebas, pruebas de usabilidad, pruebas de caja blanca.



2.7.2 Introducción

Un grupo de actividades de prueba puede estar dirigido a la verificación del sistema del software (o a una parte de un sistema) basado en una razón u objetivo específico para la prueba.

Un tipo de la prueba se centra en un objetivo particular de prueba, el cual podría ser la prueba de una función a ser realizada por el software; una característica de calidad no funcional, tales como fiabilidad o usabilidad, la estructura o arquitectura del software o del sistema; o relacionado con los cambios, es decir confirmar que los defectos han sido arreglados (pruebas de confirmación) y buscar cambios involuntarios (pruebas de regresión).

Un modelo del software puede ser desarrollado y/o usado en pruebas estructurales y funcionales. Por ejemplo, en pruebas funcionales un modelo de flujo de proceso, un modelo de transición de estado o una especificación de lenguaje simple; y para pruebas estructurales un modelo de flujo de control o un modelo de estructura de menú.

2.7.3 Prueba de función (prueba funcional)

Las funciones, que un sistema, un subsistema o un componente realizará, pueden estar descritas en productos de trabajo tales como una especificación de requisitos, casos de uso o una especificación funcional, o podrían estar no documentados. Las funciones son “lo que” el sistema hace.

Las pruebas funcionales están basadas en estas funciones y rasgos (descritos en documentos o comprendidos por los probadores), y pueden ser realizadas en todos los niveles de prueba (por ejemplo, las pruebas para componentes pueden estar basadas en una especificación de componente).

Las técnicas basadas en especificación pueden ser usadas para derivar condiciones de prueba y casos de prueba de la funcionalidad del software o del sistema. Las pruebas funcionales consideran el comportamiento externo del software (pruebas de caja negra).

Un tipo de pruebas funcionales, pruebas de seguridad, investiga las funciones (por ejemplo, un firewall) relacionadas con la detección de amenazas, tales como virases, de desconocidos maliciosos.

2.7.4 Prueba de características del producto de software (prueba no funcional)

Las pruebas no funcionales incluyen, mas no están limitadas a, pruebas de rendimiento, pruebas de carga, pruebas de estrés, pruebas de usabilidad, pruebas de interoperabilidad, pruebas de mantenibilidad, pruebas de fiabilidad y pruebas de portabilidad. Es la prueba de "como" funciona el sistema.

Las pruebas no funcionales pueden ser realizadas en todos los niveles de prueba. El término pruebas no funcionales describe las pruebas requeridas para medir las características de sistemas y software que pueden ser cuantificadas en una escala diversa, tales como tiempos de respuesta para pruebas de rendimiento. Estas pruebas pueden ser referenciadas a un modelo de calidad tales como aquel definido en "Ingeniería de Software – Calidad del Producto de Software" (ISO 9126).

2.7.5 Prueba de estructura/arquitectura de software (prueba estructural)

Las pruebas estructurales (de caja blanca) pueden ser realizadas en todos los niveles de prueba. Las técnicas estructurales son usadas mejor después de las técnicas basadas en especificación, para ayudar a medir la minuciosidad de las pruebas a través de la evaluación de cobertura de un tipo de estructura.

La cobertura está en la medida en que una estructura ha sido ejercida por un conjunto de pruebas, expresada como un porcentaje de elementos que están siendo cubiertos. Si la cobertura no es del 100%, entonces más pruebas podrán ser diseñadas para probar aquellos elementos que faltaron, y por lo tanto, incrementar la cobertura.

En todos los niveles de prueba, mas especialmente en las pruebas de componente y en las pruebas de integración de componente, herramientas pueden ser usadas para medir la cobertura de código de los elementos, tales como sentencias o decisiones. Las pruebas estructurales pueden estar basadas en la arquitectura del sistema, tales como una jerarquía de llamada.
Los enfoques de pruebas estructurales pueden ser también aplicados en niveles de sistema, de integración de sistema o de pruebas de aceptación (por ejemplo, a modelos de negocio o estructuras de menú).

2.7.6 Pruebas relacionadas con cambios (pruebas de confirmación y pruebas de regresión)

Cuando un defecto es detectado y arreglado, entonces el software debe ser probado de nuevo para confirmar que el defecto original ha sido retirado exitosamente. Esto se llama prueba de confirmación. La depuración (arreglo de defectos) es una actividad de desarrollo, no una actividad de pruebas.

Las pruebas de regresión son las pruebas repetidas de un programa ya probado, después de la modificación, para descubrir cualquier defecto introducido o no descubierto como un resultado de el(los) cambio(s). Estos defectos pueden estar ya sea en el software que está siendo probado o en cualquier componente del software relacionado o no relacionado. Son realizadas cuando el software, o su entorno, es cambiado.
La medida de las pruebas de regresión está basada en el riesgo de no encontrar defectos en el software que estaba funcionando previamente.

Las pruebas deben ser repetibles si serán usadas para las pruebas de confirmación y para asistir a las pruebas de regresión.

Las pruebas de regresión pueden ser realizadas en todos los niveles de prueba, y se aplican a las pruebas funcionales, no funcionales y estructurales. Los conjuntos de pruebas de regresión son corridos muchas veces y generalmente evolucionan lentamente, así las pruebas de regresión son un fuerte candidato para la automatización.

Pruebas típicas:

- Pruebas de confirmación: repetición de pruebas, después de la corrección de errores
- Pruebas de regresión: pruebas para descubrir defectos introducidos en funcionalidad previamente sin fallos

Criterio de selección:

- Casos de prueba de prioridad alta
- Probar solamente la funcionalidad estándar, saltarse casos y variaciones especiales
- Probar solamente la configuración utilizada con mayor frecuencia.
- Probar solamente subsistemas/Zonas seleccionadas del objeto de pruebas

Si durante fases tempranas del proyecto resulta evidente que ciertas pruebas son adecuadas para las pruebas de regresión, se deberá considerar la automatización de estas pruebas.

2.8 Pruebas de mantenimiento

2.8.1 Términos

Análisis de impacto, pruebas de mantenimiento, migración, modificaciones, retiro.

2.8.2 Introducción

Una vez desplegado, un sistema de software a menudo está en servicio durante años o décadas. Durante este tiempo el sistema y su entorno son a menudo corregidos, cambiados o extendidos. Las pruebas de mantenimiento son realizadas en un sistema operacional existente, y son activadas por modificaciones, migración o retiro del software o del sistema.

Las modificaciones incluyen cambios de mejora planeados (por ejemplo, basados en lanzamiento), cambios correctivos y de emergencia, y cambios de entorno, tales como actualizaciones planeadas del sistema operativo o de la base de datos, o parches a vulnerabilidades recientemente expuestas o descubiertas en el sistema operativo.

Las pruebas de mantenimiento para migración (por ejemplo, de una plataforma a otra) deben incluir pruebas operacionales del nuevo entorno, así como del software cambiado.

Las pruebas de mantenimiento para el retiro de un sistema pueden incluir las pruebas de migración de datos o de almacenamiento si periodos de largos de retención de datos son requeridos.

Además de probar que ha sido cambiado, las pruebas de mantenimiento incluyen pruebas de regresión extensivas a las partes del sistema que no han sido cambiadas. El alcance de las pruebas de mantenimiento está relacionado con el riesgo del cambio, del tamaño del sistema existente y con el tamaño del cambio. Dependiendo de los cambios, las pruebas de mantenimiento pueden ser realizadas en cualquiera o en todos los niveles de prueba y para cualquier o para todos los tipos de prueba.

Determinar como el sistema existente puede ser afectado por los cambios se llama análisis de impacto y es usado para ayudar a decidir cuantas pruebas de regresión se harán.

Referencias

2.1.3 CMMI, Craig, 2002, Hetzel, 1998, IEEE 12207
2.2 Hetzel, 1998
2.2.4 Copeland, 2004, Myers, 1979
2.3.1 Beizer, 1990, Black, 2001, Copeland, 2004
2.3.2 Black, 2001, ISO 9126
2.3.3 Beizer, 1990, Copeland, 2004, Hetzel, 1998
2.3.4 Hetzel, 1998, IEEE 829
2.4 Black, 2001, Craig, 2002, Hetzel, 1998, IEEE 829