4

**Pruebas de software**

Se pueden distinguir dos tipos de evaluaciones durante el proceso de desarrollo de software:

* **Verificación:** Proceso para determinar si los productos de una cierta fase del desarrollo SW cumplen o no los requisitos establecidos en la fase anterior. ¿Estamos creando el producto correctamente? Revisión del programa conforme a sus especificaciones.
* **Validación:** Proceso de evaluación del SW al final del proceso de desarrollo para asegurar el cumplimiento de necesidades del cliente. ¿Estamos creando el producto correcto? Revisión del programa conforme a las expectativas del usuario.

Las técnicas de evaluación SW se pueden agrupar en dos categorías:

* Técnicas estáticas: Buscan faltas sobre el sistema en reposo. Estas técnicas se pueden aplicar a los diferentes productos SW. Requerimientos, diseño, lista de programas, etc.
* Técnicas dinámicas: Generan entradas al sistema con el objetivo de detectar fallos cuando el sistema ejecuta dichas entradas. Se les conoce como pruebas del SW. Unitarias, de integración, del sistema, de aceptación.

**Niveles de pruebas**

* Pruebas de unidad: Verifican el funcionamiento aislado de partes del software que se pueden probar independientemente. Normalmente se realizan con acceso al código fuente y soporte de herramientas de depuración
* Pruebas de integración: Verifica la interacción entre componentes del software. Clásicos: son el arriba-abajo y abajo-arriba para software estructurado jerárquicamente. Modernos: Basados en arquitectura, integra componentes o subsistemas basándose en caminos de funcionalidad identificada. Se centra en el diseño y la construcción de la arquitectura del software
* Prueba de validación: Los requerimientos establecidos como parte de su modelado se validan confrontándose con el software que se construyó. Proporciona la garantía final de que el software cumple con todos los requerimientos informativos, funcionales, de comportamiento y de rendimiento.
* Pruebas del sistema: Comportamiento de un sistema completo. Compara el sistema con los requisitos no funcionales (seguridad, velocidad, exactitud, confiabilidad) y se evalúa interconexiones externas (Aplicaciones, utilidades, hardware, OS. Verifica que todos los elementos se mezclan de manera adecuada y que se logra el funcionamiento/rendimiento global del sistema

**Aspectos estratégicos**

* Especifican los requerimientos del producto en forma cuantificable mucho antes de comenzar con las pruebas
* Establecen de manera explícita los objetivos de las pruebas
* Entienden a los usuarios del software y desarrollan un perfil para cada categoría de usuario
* Desarrollan un plan de prueba que enfatice “pruebas de ciclo rápido
* Construyen software “robusto” que esté diseñado para probarse a sí mismo.
* Usan revisiones técnicas efectivas como filtro previo a las pruebas
* Realizan revisiones técnicas para valorar la estrategia de prueba y los casos de prueba.
* Desarrollan un enfoque de mejora continuo para el proceso de prueba.

**Pruebas de unidad**

Verificación en la unidad más pequeña del diseño de software: el componente o módulo de software. Se enfocan en la lógica de procesamiento interno y de las estructuras de datos dentro de las fronteras de un componente.

El diseño de las pruebas de unidad puede ocurrir antes de comenzar la codificación o después de generar el código fuente.

Un controlador no es más que un “programa principal” que acepta datos de caso de prueba, pasa tales datos al componente (que va a ponerse a prueba) e imprime resultados relevantes.

Los representantes sirven para sustituir módulos que están subordinados al (invocados por el) componente que se va a probar. Un representante usa la interfaz de módulo subordinado, puede realizar mínima manipulación de datos, imprimir verificación de entradas y regresar el control al módulo sobre el que se realiza la prueba

**Pruebas de validación**

* Revisión de la configuración (Auditoría): garantizar que todos los elementos de la configuración del software se desarrollaron de manera adecuada, y que se cataloga y se tiene el detalle necesario para reforzar las actividades de apoyo
* Pruebas Alfa y beta: prueba **alfa** se lleva a cabo en el sitio del desarrollador por un grupo representativo de usuarios finales. El software se usa en un escenario natural con el desarrollador “mirando sobre el hombro” de los usuarios y registrando los errores y problemas de uso. Las pruebas alfa se realizan en un ambiente controlado. Prueba **beta** se realiza en uno o más sitios del usuario final. Es una aplicación “en vivo” del software en un ambiente que no puede controlar el desarrollador. El cliente registra todos los problemas (reales o imaginarios) que se encuentran durante la prueba beta y los reporta al desarrollador periódicamente.

**Pruebas del sistema**

* Pruebas de recuperación: fuerza al software a fallar en varias formas y verifica que la recuperación se realice de manera adecuada. Si la recuperación es automática, se evalúa el reinicio, los mecanismos de puntos de verificación, la recuperación de datos y la reanudación para correcciones. Si la recuperación requiere intervención humana, se evalúa el tiempo medio de reparación (TMR) para determinar si está dentro de límites aceptables.
* Pruebas de seguridad: verificar que los mecanismos de protección que se construyen en un sistema en realidad lo protegerán de cualquier penetración impropia
* Prueba de esfuerzo: ejecuta un sistema en forma que demanda recursos en cantidad, frecuencia o volumen anormales.
* Prueba de rendimiento: pone a prueba el rendimiento del software en tiempo de corrida, dentro del contexto de un sistema integrado. La prueba del rendimiento ocurre a lo largo de todos los pasos del proceso de prueba.
* Prueba de despliegue(o de configuración): Ejercita el software en cada entorno en el que debe operar. Además, examina todos los procedimientos de instalación y el software de instalación especializado que usarán los clientes, así como toda la documentación que se usará para introducir el software a los usuarios finales.

**Depuración**

Consecuencia de las pruebas exitosas. Cuando un caso de prueba descubre un error, la depuración es el proceso que da como resultado la remoción del error.

El proceso de depuración comienza con la ejecución de un caso de prueba. Los resultados se valoran y se encuentra la falta de correspondencia entre el rendimiento esperado y el real. Intenta relacionar síntoma con causa, lo que por tanto conduce a la corrección del error. Dará como resultado que: 1) la causa se encontrará y corregirá o 2) la causa no se encontrará.

Estrategias:

* Fuerza bruta: Se toman copias de la memoria (dumps), se invocan rastreos en el tiempo de corrida y el programa se carga con enunciados de salida. La esperanza es que, en alguna parte del pantano de información que se produzca, se encontrará una pista que pueda conducir a la causa de un error.
* Seguimiento hacia atrás: Al comenzar en el sitio donde se descubrió un síntoma, el código fuente se rastrea hacia atrás (de manera manual) hasta que se encuentra la causa
* Eliminación de la causa Los datos relacionados con la ocurrencia del error se organizan para aislar las causas potenciales. Se plantea una “hipótesis de causa” y los datos anteriormente mencionados se usan para probar o refutar la hipótesis.

Una vez encontrado el error, debe corregirse. Pero, como ya se señaló, la corrección de un error puede introducir otros errores y, por tanto, hacer más daño que bien.

**Información extra**

**Objetivos de las pruebas:**

* Pruebas de aceptación/Calificación: Compara el comportamiento del sistema con los requisitos del cliente
* Pruebas de instalación: Pruebas del sistema realizadas en relación con los requisitos de la configuración del hardware
* Pruebas alfa y beta: Se proporciona el software a un grupo representativo de usuarios potenciales para usarlo en pruebas en las instalaciones del desarrollador (alpha) o externamente (beta)
* Pruebas de conformidad/funcionales/de corrección: Verifica si el comportamiento del software corresponde con las especificaciones.
* Materialización de la confiabilidad y evaluación: Genera casos de prueba aleatorios (identificar errores) para derivar aproximaciones estadísticas de confiabilidad
* Pruebas de regresión: Pruebas selectivas que se repiten en un componente para verificar que los cambios no han producido efectos indeseados.
* Pruebas de rendimiento: Verifica que se han alcanzado los requerimientos de rendimiento especificados, particularmente de capacidad y tiempo de respuesta
* Pruebas de desgaste: Hacen funcionar al software a su máxima capacidad y por encima de ella
* Pruebas de continuidad: Un grupo de pruebas se ejecuta en 2 versiones diferentes del software y se comparan
* Pruebas de recuperación: Probar capacidad del software para reiniciarse después de un desastre
* Pruebas de configuración: analiza en distintas configuraciones especificadas (Software de servicio para distintos usuarios)
* Pruebas de facilidad de uso: evalúa lo fácil que es para el usuario aprender y usar el software. Incluye documentación de usuario, efectividad de las funciones y recuperación de errores provocados por usuario
* Desarrollo dirigido por pruebas: Promociona el uso de pruebas como una parte subordinada al documento de especificación de requisitos

**Técnica de pruebas:** Caja blanca (cómo se ha programado o diseñado el software), caja negra (comportamiento de entrada y salida de datos) y combinado.

* Pruebas basadas en la intuición y experiencia del ingeniero de software:

*Pruebas ad hoc:* identificar casos de prueba especiales que no se extraen fácilmente mediante técnicas formales. A partir de la habilidad, intuición y experiencia en programas similares.

*Pruebas por exploración:* aprendizaje, diseño de pruebas y ejecución al mismo tiempo. Se basa en el conocimiento del ingeniero de software acerca del comportamiento observado durante las pruebas, su familiaridad con la aplicación, la plataforma o el proceso de fallos, etc

* Técnicas basadas en la especificación:

*Particiones de equivalencia:* El dominio de la entrada de datos se subdivide se subdivide en subconjuntos o clases de equivalencias, las cuales se consideran equivalentes de acuerdo con la relación especificada.

*Análisis de los valores límite:* Casos de prueba se seleccionan en y cerca de los límites de los límites del dominio de las variable de la entrada de datos. *Prueba de robustez,* se selecciona casos de prueba que se encuentren fuera del dominio de las variables de entrada de datos. Entradas de datos erróneas e inesperadas

*Tablas de decisión:* Representan relaciones lógicas entre condiciones (Entradas) y acciones (Salidas).

*Basadas en máquinas de estado finito:* Cubren estados y transiciones

*Pruebas basadas en especificaciones formales:* Uso de oráculos

*Pruebas aleatorias:* Entradas basadas en la especificación, ya que el dominio de entradas de datos se debe conocer para ser capaces de de seleccionar elementos aleatorios del mismo

* Técnicas basadas en el código:

*Criterio basado en el flujo de control:* Cubren todos los bloques de código o líneas de código individuales o una combinación específica de los mismos. Prueba de caminos (verifica todos los caminos de control de entrada-salida del gráfico de flujo), pruebas de líneas de código, de condiciones y de decisión. Se mide en porcentajes

*Criterio basado en el flujo de datos:* El gráfico de flujo de de control tiene anotaciones con información de cómo las variables se definen, usan y destruyen.

*Modelos de referencia para pruebas basadas en código:* La estructura de control de un programa se representa usando gráficos de flujo.

* Técnicas basadas en errores:

*Conjeturar errores:* Intenta imaginar los errores más probables de un programa determinado

*Pruebas de mutación :* Se crean versiones ligeramente modificadas del programa (mutantes), a los que se le hacen pruebas, diferenciándose en un pequeño  cambio sintáctico. El caso de prueba intenta identificar la diferencia entre el mutante y el original, si lo hace, se elimina, si no lo detecta, se diseñan pruebas para eliminar a los supervivientes.

* Técnicas basadas en el uso

*Perfil operativo:* Se debe reproducir el entorno operativo tan fielmente como sea posible para deducir su confiabilidad durante su uso real

*Pruebas orientadas a la confiabilidad del software:* Diseñada y guiada por los objetivos de confiabilidad y el uso relativo esperado esperado y lo criticas que sean las funciones dentro de ese ámbito.

* Técnicas basadas en la naturaleza de la aplicación:

*Orientada a objetos*

*Basada en componentes*

*Para internet*

*para GUI*

*Para programas concurrentes*

*Conformidad de protocolos*

*Para sistemas de tiempo real*

*Sistemas de seguridad crítica*

* Seleccionando y combinando técnicas

*Funcional y estructuralmente:* Basadas en especificaciones y código

*Deterministas vs aleatorias:*

**Medidas de las pruebas:** Proporcionan una evaluación del Programa que se está evaluando, basada en los resultados observados de las pruebas y aquellas que evalúan la completitud del conjunto de pruebas. Esenciales en el análisis de calidad. Optimizan la planificación y ejecución de pruebas

* Evaluación de un programa durante las pruebas:

*Medidas para ayudar en la planificación y diseño de pruebas de programas:* Basadas en el tamaño (Número de líneas de código o métodos) o estructura de un programa (complejidad)

*Tipos de errores, clasificación y estadística:* Conocer que tipos de errores se pueden encontrar en un programa que se está comprobando y la frecuencia relativa en que estos errores han sucedido antes.

*Densidad de fallos:* Razón entre el número de errores encontrados (para cada tipo) y el tamaño del programa

*Vida de las pruebas, evaluación de confiabilidad:* Evalúa el producto para decidir si las pruebas se pueden detener o no

*Modelos de crecimiento de la confiabilidad:* Predicción de confiabilidad basada en los fallos observados durante la realización y evaluación de confiabilidad. *Cuenta de fallos y tiempo de fallos*

* Evaluación de pruebas realizadas

*Medidas de la cobertura/completitud:* Se monitorizan los elementos cubiertos y su número total

*Introducción de errores:* Se introducen errores artificialmente para evaluar la efectividad de las pruebas y estimar el número restante de errores

*Puntuación de la mutación:* Razón de mutantes matados por número total de mutantes generados

*Comparación y efectividad relativa de las diferentes técnicas:* Número de pruebas realizadas para encontrar el primer fallo, razón de los errores encontrados durante y después de las pruebas.

**El proceso de las pruebas:**

* Consideraciones prácticas:

*Actitudes y programación egoless:* Colaboración respecto a las actividades de pruebas y garantía de calidad. Nadie es dueño del código

*Guías para las pruebas: Ejemplo* Prioridad de riesgos o definir casos de prueba específicos

*Gestión del proceso de pruebas:* Las pruebas se agrupan con otras actividades de evaluación como una parte integral del ciclo de vida completo.

*Documentación y productos de las pruebas:* Plan de pruebas, Especificación de Diseño de pruebas, Especificación del Procedimiento de las pruebas, Especificación de los Casos de prueba, Diario de las pruebas, Informe de Problemas o Incidentes

*Equipo de pruebas interno vs independiente:*

*Estimación coste/esfuerzo y otras medidas del proceso:* Número de casos de prueba especificados, ejecutados, superados y no superados

*Finalización:* Cuántas pruebas son suficientes y cuando finaliza la fase de pruebas. Se evalúa coste y riesgos

*Reutilización de pruebas y patrones de pruebas:* Repositorio para usar de manera sistemática

* Actividades de las pruebas

*Planificación:* Coordinación del personal, equipo disponible, gestión de instalaciones.

*Generación de casos de prueba:* bajo el control de software de gestión de configuraciones

*Desarrollo en el entorno de pruebas:* Debería ser compatible con herramientas de ingeniería de software

*Ejecución:* Todos los pasos se deben realizar y documentar de forma clara

*Evaluación de resultados:* Antes de arreglar un error, se analiza y depura para identificarlo, aislarlo y describirlo

*Notificación de problemas/Diario de pruebas:* Documentar. Configuración del software, quien ha ejecutado la prueba

*Seguimiento de defectos:* Cuando fue introducido el error, qué lo produjo y cuando debió ser observado por primera vez

**Preguntas**

1. Proceso para determinar si los productos de una cierta fase del desarrollo del software cumplen o no los requisitos establecidos en la fase anterior.
2. **Verificación**
3. Validación
4. Evaluación
5. Seguimiento
6. Proceso de evaluación del software al final del proceso de desarrollo para asegurar el cumplimiento de necesidades del cliente.
7. Verificación
8. **Validación**
9. Evaluación
10. Seguimiento
11. Las técnicas de evaluación estáticas buscan faltas sobre el sistema en reposo. Estas técnicas se pueden aplicar a los diferentes productos del software como los requerimientos, diseño, lista de programas, etc.
12. **Verdadero**
13. Falso
14. Este tipo de técnicas de evaluación genera entradas al sistema con el objetivo de detectar fallos cuando el sistema ejecuta dichas entradas. Se les conoce como pruebas del Software.
15. Estáticas
16. **Dinámicas**
17. Son ejemplos de pruebas del software, excepto:
18. Pruebas unitarias
19. Pruebas de integración
20. Pruebas del sistema
21. **Prueba de diseño**
22. En este nivel de pruebas, se verifica el funcionamiento aislado de partes del software que se pueden probar independientemente. Normalmente se realizan con acceso al código fuente y soporte de herramientas de depuración.
23. **Prueba de Unidad**
24. Prueba de integración
25. Prueba de validación
26. Prueba del sistema
27. Este nivel de pruebas se centra en el diseño y la construcción de la arquitectura del software. Se verifica la interacción entre componentes del software.
28. Prueba de Unidad
29. **Prueba de integración**
30. Prueba de validación
31. Prueba del sistema
32. En este nivel de pruebas, los requerimientos establecidos como parte de su modelado se validan confrontándose con el software que se construyó.
33. Prueba de Unidad
34. Prueba de integración
35. **Prueba de validación**
36. Prueba del sistema
37. En este nivel de pruebas, se evalúa el comportamiento de un sistema completo. Compara el sistema con los requisitos no funcionales (seguridad, velocidad, exactitud, confiabilidad) y se evalúa interconexiones externas (Aplicaciones, utilidades, hardware, OS)
38. Prueba de Unidad
39. Prueba de integración
40. Prueba de validación
41. **Prueba del sistema**
42. Esta prueba de validación se lleva a cabo en el sitio del desarrollador por un grupo representativo de usuarios finales. El software se usa en un escenario natural con el desarrollador “mirando sobre el hombro” de los usuarios y registrando los errores y problemas de uso.
43. **Prueba alfa**
44. Prueba beta
45. Esta prueba de validación se realiza en uno o más sitios del usuario final. Es una aplicación “en vivo” del software en un ambiente que no puede controlar el desarrollador. El cliente registra todos los problemas (reales o imaginarios) que se encuentran durante la prueba y los reporta al desarrollador periódicamente.
46. Prueba alfa
47. **Prueba beta**
48. En esta prueba del sistema se fuerza al software a fallar en varias formas y verifica que la recuperación se realice de manera adecuada. Si es automática, se evalúa el reinicio, los mecanismos de puntos de verificación, la recuperación de datos y la reanudación para correcciones. Si requiere intervención humana, se evalúa el tiempo medio de reparación (TMR) para determinar si está dentro de límites aceptables.
49. Prueba de seguridad
50. **Prueba de recuperación**
51. Prueba de despliegue
52. Prueba de rendimiento
53. En esta prueba del sistema se verificar que los mecanismos de protección que se construyen en un sistema en realidad lo protegerán de cualquier penetración impropia.
54. **Prueba de seguridad**
55. Prueba de esfuerzo
56. Prueba de despliegue
57. Prueba de rendimiento
58. En esta prueba del sistema se ejecuta un sistema en forma que demanda recursos en cantidad, frecuencia o volumen anormales.
59. Prueba de rendimiento
60. Prueba de recuperación
61. **Prueba de esfuerzo**
62. Prueba de seguridad
63. En esta prueba del sistema se pone evalúa el rendimiento del software en tiempo de corrida, dentro del contexto de un sistema integrado. Ocurre a lo largo de todos los pasos del proceso de prueba.
64. **Prueba de rendimiento**
65. Prueba de recuperación
66. Prueba de despliegue
67. Prueba de seguridad
68. En esta prueba del sistema, se ejercita el software en cada entorno en el que debe operar. Además, examina todos los procedimientos de instalación y el software de instalación especializado que usarán los clientes, así como toda la documentación que se usará para introducir el software a los usuarios finales.
69. Prueba de recuperación
70. Prueba de seguridad
71. Prueba de esfuerzo
72. **Prueba de despliegue**
73. Como consecuencia de las pruebas exitosas. Cuando un caso de prueba descubre un error, es el proceso que da como resultado la remoción del error.
    1. **Depuración**
    2. Reinicio
    3. Formateo
    4. Recuperación