Volver

U3.1 - Pruebas de Software

Volver

Rafael Alberti
Instituto de Educación Secundaria

Volver

Indice

Volver

Introducción

Volver

¿Qué son las pruebas de software?

- Evaluación de un sistema o aplicación.
- Busca comprobar que se cumplen los requisitos.
- Asegura el funcionamiento correcto del software.
- Su objetivo es minimizar errores y asegurar calidad.
- La calidad incluye funcionalidad, rendimiento, seguridad.
- También engloba usabilidad y mantenibilidad.

Volver

Importancia de las pruebas

- Detectan errores en etapas tempranas.
- Reducen el coste de corrección de errores.
- Aumentan la satisfacción del usuario final.
- Evitan errores por malentendidos de requisitos.
- Identifican fallos de seguridad y rendimiento.
- Mejoran la calidad general del producto.

Volver

Beneficios de realizar pruebas

- Mejora de la calidad del software.
- Aumenta la confianza del usuario en el producto.
- Reduce costos de mantenimiento post-lanzamiento.
- Evita errores en producción.
- Identifica fallos desde las fases iniciales.
- Contribuye a una marca más confiable.

Volver

Verificación y prueba de programas

Volver

Definición según ISTQB

- Incluye actividades del ciclo de vida del software.
- Comprende tareas estáticas y dinámicas.
- Evalúa productos para comprobar requisitos.
- Detecta defectos y asegura cumplimiento del propósito.
- Involucra planificación, preparación y evaluación.

Volver

Principios básicos de la prueba

- Las pruebas demuestran presencia, no ausencia de errores.
- No es viable probar todo (excepto en casos triviales).
- Probar desde etapas tempranas reduce costes.
- Defectos suelen concentrarse en pocos módulos.
- Las mismas pruebas pierden eficacia con el tiempo.
- Las pruebas varían según el contexto del sistema.
- Corregir errores no basta si el sistema no es útil.

Volver

Ciclo de vida de pruebas

- Planificación y control: objetivos y gestión de pruebas.
- Análisis y diseño: definición de casos de prueba.
- Implementación y ejecución: desarrollo y ejecución.
- Evaluación de resultados: comparación con criterios.
- Cierre de pruebas: lecciones aprendidas y documentación.

Volver

Niveles de pruebas

Volver

Niveles según ISTQB

- Pruebas unitarias: testean componentes individuales.
- Pruebas de integración: validan interacción entre módulos.
- Pruebas de sistema: evalúan el sistema completo.
- Pruebas de aceptación: realizadas por el cliente final.
- Cada nivel se asocia a una fase del desarrollo.
- El modelo en V muestra esta correspondencia.

Volver

Técnicas de verificación y prueba

Volver

Clasificación de técnicas

- Dinámicas: requieren la ejecución del software.
- Estáticas: no requieren ejecución del código.
- Las técnicas se seleccionan según el objetivo.
- Complementarias entre sí para mayor eficacia.

Volver

Pruebas dinámicas: Caja negra

- No se conoce la estructura interna del código.
- Basadas en entradas/salidas del sistema.
- Tipos:
  - Partición equivalente.
  - Valores límites.
  - Transición de estado.
  - Casos de uso.
  - Basadas en experiencia.

Volver

Pruebas dinámicas: Caja blanca

- Conocimiento de la lógica interna del código.
- Se estudia el flujo de ejecución y sentencias.
- Tipos:
  - Camino básico.
  - Cobertura de sentencias.
- Asegura ejecución de todas las rutas posibles.

Volver

Técnicas estáticas

Volver

Técnicas estáticas de prueba

- No requieren ejecutar el código.
- Se basan en revisión y análisis de código.
- Permiten detectar defectos tempranamente.
- Incluyen revisión de requisitos y documentación.

Volver

Revisión de código y análisis estático

- Revisión de código: hecha por compañeros del equipo.
- Análisis estático: realizado por herramientas automáticas.
- Busca errores, malas prácticas y mejoras posibles.
- Ejemplos: SonarQube, Lint.

Volver

Otras técnicas estáticas

- Inspecciones formales con roles definidos.
- Caminatas de código dirigidas por el autor.
- Auditorías para verificar cumplimiento de normas.
- Análisis de requisitos y documentación.

Volver

Tipos de pruebas de software

Volver

Pruebas unitarias

- Evalúan unidades individuales del código.
- Pruebas aisladas del resto del sistema.
- Realizadas por desarrolladores.
- Permiten detectar errores funcionales simples.
- Verifican salidas esperadas con entradas dadas.
- Aumentan la mantenibilidad del software.

Volver

Pruebas de integración

- Verifican interacción entre distintos módulos.
- Detectan errores de comunicación entre interfaces.
- Se realizan tras las pruebas unitarias.
- Pueden usar stubs o drivers si faltan módulos.
- Evaluación de datos compartidos y lógica integrada.

Volver

Pruebas de sistema

- Evalúan el sistema como un todo integrado.
- Simulan condiciones de uso reales.
- Se verifican funcionalidades globales.
- Se prueba estabilidad, rendimiento y usabilidad.
- Son la antesala a la prueba de aceptación.

Volver

Pruebas de aceptación

- Realizadas por el cliente o usuario final.
- Verifican si el software cumple lo esperado.
- Validan funcionalidad y requisitos definidos.
- Se basan en casos de uso reales del negocio.
- Deciden si el sistema puede pasar a producción.

Volver

Pruebas de regresión

- Aseguran que cambios no generen nuevos errores.
- Se ejecutan tras modificaciones o correcciones.
- Se automatizan para ejecuciones repetidas.
- Comparan comportamientos antiguos y actuales.
- Detectan efectos secundarios no deseados.

Volver

Pruebas de carga y rendimiento

- Evalúan comportamiento bajo demanda elevada.
- Determinan el límite de usuarios concurrentes.
- Miden tiempos de respuesta y consumo de recursos.
- Aseguran estabilidad y velocidad del sistema.
- Identifican cuellos de botella.

Volver

Pruebas de seguridad

- Evalúan resistencia ante ataques o vulnerabilidades.
- Protegen datos y sistemas ante accesos no autorizados.
- Incluyen pruebas de autenticación y cifrado.
- Simulan ataques conocidos (XSS, SQLi...).
- Imprescindibles en software que maneja información sensible.

Volver

Plan de pruebas

Volver

¿Qué es un plan de pruebas?

- Documento que organiza y define las pruebas.
- Establece objetivos, alcance y cronograma.
- Determina recursos, herramientas y responsabilidades.
- Asegura ejecución efectiva de pruebas.
- Permite evaluar resultados de forma estructurada.

Volver

Definición del alcance

- Determina qué se va a probar y qué no.
- Incluye funcionalidades críticas del sistema.
- Define límites para centrar el esfuerzo de pruebas.
- Asegura cobertura de requisitos prioritarios.

Volver

Identificación de recursos

- Se especifican los equipos necesarios.
- Se detalla el personal implicado en las pruebas.
- Se definen herramientas y datos de prueba requeridos.
- Ayuda a prever y evitar cuellos de botella.

Volver

Selección de herramientas

- Herramientas para pruebas funcionales (Selenium...).
- Herramientas para pruebas de rendimiento (JMeter...).
- Herramientas para pruebas unitarias (MockK, Kotest).
- Control de versiones para scripts de prueba (Git).

Volver

Diseño de casos de prueba

- Casos que verifican diferentes funcionalidades.
- Se deben cubrir escenarios normales y extremos.
- Incluyen datos de entrada, pasos y resultados esperados.
- Organizados por funcionalidad o módulo.

Volver

Asignación de responsabilidades

- Cada miembro tiene un rol claro.
- Se asignan pruebas a ejecutores específicos.
- Mejora el seguimiento y control del proceso.
- Fomenta la responsabilidad y la colaboración.

Volver

Cronograma de pruebas

- Se define cuándo se ejecutará cada fase.
- Incluye fechas y responsables.
- Facilita seguimiento del progreso.
- Incluye revisión de incidencias y retroalimentación.

Volver

Pruebas de integración

Volver

¿Qué son las pruebas de integración?

- Verifican interacción entre distintos componentes.
- Se realizan tras las pruebas unitarias.
- Detectan fallos en comunicación entre módulos.
- Garantizan la cohesión del sistema.

Volver

Tipos de integración

- Ascendente: de componentes bajos a altos.
- Descendente: de componentes altos a bajos.
- Híbrida: combinación de ambos enfoques.
- Cada tipo tiene ventajas según el contexto.

Volver

Estrategias de integración

- Big-Bang: todo junto, simple pero difícil de depurar.
- Por módulos: integración parcial progresiva.
- Fachada: simula módulos no disponibles.
- Stubs y Drivers: simulan componentes faltantes.

Volver

Pruebas de sistema

Volver

¿Qué son las pruebas de sistema?

- Evalúan el sistema completo e integrado.
- Validan cumplimiento de requisitos globales.
- Se realizan antes de la aceptación.
- Se centran en el producto final desde la visión del usuario.

Volver

Tipos de pruebas de sistema

- Funcionalidad: validan funciones del sistema.
- Rendimiento: evalúan tiempo de respuesta.
- Carga: verifican soporte ante muchos usuarios.
- Seguridad: evalúan resistencia a amenazas.
- Compatibilidad: en distintos entornos.
- Usabilidad: enfocadas en experiencia de usuario.

Volver

Estrategias de prueba de sistema

- Casos de uso: basados en escenarios del usuario.
- Escenarios completos: simulaciones realistas.
- Extremo a extremo: de principio a fin del flujo.
- Estrés: pruebas bajo condiciones extremas.
- Compatibilidad y seguridad.

Volver

Pruebas de aceptación

Volver

¿Qué son las pruebas de aceptación?

- Validan si el software cumple requisitos del cliente.
- Última etapa antes de entregar el sistema.
- Realizadas por usuarios o clientes.
- Determinan si se aprueba el producto.

Volver

Tipos de pruebas de aceptación

- Funcionales: validan requisitos funcionales.
- No funcionales: rendimiento, usabilidad, etc.
- Simulan escenarios reales del negocio.
- Determinan si se puede usar el sistema en producción.

Volver

Estrategias de aceptación

- Participación del cliente en la validación.
- Casos de uso centrados en el negocio.
- Pruebas manuales y automáticas combinadas.
- Definición clara de criterios de éxito.

Volver

Conclusiones

Volver

Resumen de lo aprendido

- Las pruebas aseguran calidad del software.
- Tipos y niveles permiten cubrir todo el proceso.
- Técnicas estáticas y dinámicas son complementarias.
- Un plan bien definido es esencial para el éxito.
- La colaboración entre equipos es clave.
- Las pruebas deben adaptarse a requisitos cambiantes.