CAPÍTULO 1

Conceptos Básicos

Este capítulo presenta los conceptos fundamentales sobre pruebas de software.

1.1 ¿Qué es la prueba?

El software está presente en casi todos los aspectos de nuestra vida diaria. Desde aplicaciones móviles hasta sistemas hospitalarios, confiamos en que el software funcione correctamente. Sin embargo, cuando falla, puede causar pérdidas económicas, daños a la reputación o incluso lesiones o muertes.

Las pruebas de software ayudan a evaluar la calidad del sistema y a reducir los riesgos de fallos. Es un error común pensar que probar significa solo ejecutar código. Las pruebas incluyen también actividades de planificación, diseño de casos, criterios de finalización, control de calidad y reporte de resultados. Existen dos tipos principales:

- **Pruebas dinámicas**: implican ejecutar el código del sistema o componente.
- **Pruebas estáticas**: implican revisar productos de trabajo como requisitos, código fuente o historias de usuario sin ejecutarlos.

Las pruebas no solo verifican que el sistema cumpla especificaciones, sino que también validan que satisface las necesidades reales de los usuarios y otros actores involucrados.

Definición

"La prueba es el proceso que abarca todas las actividades del ciclo de vida, tanto estáticas como dinámicas, relacionadas con la planificación, preparación y evaluación de componentes o sistemas y sus productos asociados para determinar que satisfacen requisitos especificados, demostrar que son aptos para su propósito y detectar defectos". **ISTQB**

1.1.1 Objetivos típicos de las pruebas

Los objetivos de las pruebas pueden variar dependiendo del contexto, nivel de prueba y modelo de desarrollo. Algunos objetivos comunes incluyen:

- Prevenir defectos revisando productos como requisitos, diseño y código.
- Verificar que los requisitos se hayan cumplido.
- Validar que el sistema funciona como esperan los usuarios.
- Generar confianza en la calidad del software.
- Detectar defectos y fallos.
- Apoyar la toma de decisiones informadas por parte de los interesados.
- Cumplir requisitos contractuales o normativos.

Cada nivel de prueba puede tener objetivos distintos. Por ejemplo, en pruebas de aceptación se busca confirmar que el sistema cumpla las expectativas de los usuarios, mientras que en pruebas de integración se buscan errores en la interacción entre componentes.

1.1.2 Diferencias entre pruebas y depuración

Las pruebas y la depuración son actividades diferentes pero complementarias:

- **Probar** sirve para encontrar defectos, pero no para corregirlos.
- **Depurar** es el proceso de encontrar la causa del defecto, analizarlo y corregirlo.

Los testers se encargan de detectar y reportar defectos, mientras que los desarrolladores se encargan de depurarlos y verificar que el problema se haya solucionado.

1.2 ¿Por qué son necesarias las pruebas?

Las fallas de software pueden tener consecuencias catastróficas. El capítulo presenta varios ejemplos históricos:

- **Ariane 5 (1996)**: una conversión de número mal manejada causó la explosión de un cohete, con pérdidas de $500 millones.
- **Patriot Missile (1991)**: un error de redondeo impidió interceptar un misil, provocando 28 muertes.
- **Heathrow Terminal 5 (2008)**: errores en el sistema de equipaje causaron miles de maletas extraviadas y pérdidas millonarias.
- **Knight Capital (2012)**: una falla en software de trading causó pérdidas de $440 millones en un día.
- **Radioterapia en Panamá (2000)**: errores en software de dosis de radiación causaron 17 muertes.

Estos casos demuestran que los defectos de software pueden tener consecuencias muy graves.

1.2.1 Cómo contribuyen las pruebas al éxito

Las pruebas ayudan a evitar entregas defectuosas mediante:

- Técnicas de prueba adecuadas aplicadas por personal capacitado.
- Involucramiento de testers desde etapas tempranas como revisión de requisitos o diseño.
- Validación del software antes de su liberación.

Todo esto mejora la calidad y reduce riesgos.

1.2.2 Aseguramiento de calidad, Control de Calidad y Testing

Aunque a veces se usa "aseguramiento de calidad" como sinónimo de pruebas, no son lo mismo:

- **QA (Quality Assurance)**: Se enfoca en mejorar los procesos de desarrollo para que los errores no ocurran. Es amplio y estratégico.
- **Control de Calidad (Quality Control)**: Se encarga de revisar el producto final para verificar que cumpla con los estándares. Está más cerca de la entrega.
- **Testing**: Es una actividad específica dentro del control de calidad enfocada en ejecutar pruebas para encontrar errores.

Analogía Simple. QA es como establecer las reglas para hornear un buen pastel (receta, ingredientes, temperatura). Control de calidad es revisar si el pastel horneado cumple con esas reglas. Testing es probar el pastel para ver si sabe bien.

Los tres son fundamentales para asegurar productos de alta calidad, pero cada uno actúa en diferentes niveles.

1.2.3 Errores, defectos y fallos

- **Error**: acción humana incorrecta, como malinterpretar un requerimiento.
- **Defecto**: imperfección en el producto, como un código mal implementado.
- **Fallo**: comportamiento incorrecto en ejecución, como mostrar un valor equivocado.

No todos los defectos causan fallos: puede depender de entradas específicas o condiciones del entorno (ej. temperatura, radiación).

1.2.4 Causas raíz y efectos

Al detectar un defecto, los testers y desarrolladores intentan identificar su **causa raíz**, como requisitos poco claros o errores lógicos en el código. Entender estas causas permite tomar acciones para prevenir futuros errores similares.

> **Ejemplo**: si el sistema calcula mal los sueldos debido a requisitos mal definidos por un analista sin experiencia, la causa raíz es la falta de conocimiento del analista. El efecto visible será el pago erróneo y la generación de reclamos.

1.3 Los siete principios de la prueba

Estos principios guían la práctica efectiva de pruebas de software:

1. **Testing shows presence of defects, not their absence**. Probar demuestra que hay errores, pero no garantiza que no haya más. Aunque no se encuentren fallos, no se puede asegurar que el software sea perfecto.
2. **Exhaustive testing is impossible**. Probar todas las combinaciones posibles de entradas y condiciones es inviable excepto en casos triviales. Por eso se priorizan pruebas según riesgo, impacto y probabilidad.
3. **Early testing saves time and money**. Detectar errores pronto en el ciclo de vida es mucho más económico que corregirlos en etapas tardías. Este principio también se conoce como "shift-left testing".
4. **Defects cluster together**. Una pequeña parte del sistema suele contener la mayoría de los errores (Principio de Pareto: 80/20). Las pruebas deben enfocarse más en esas áreas.
5. **Beware of Pesticide paradox** Si se ejecutan siempre las mismas pruebas, llegará un momento en que dejarán de encontrar errores. Es necesario revisar y actualizar los casos de prueba regularmente.
6. **Testing is context dependent**. El tipo de pruebas a realizar depende del dominio y del tipo de software. No es lo mismo probar un sistema bancario crítico que una aplicación de entretenimiento.
7. **Absence-of-errors fallacy**. Corregir todos los defectos no garantiza que el sistema sea útil o satisfactorio para los usuarios. Un sistema libre de errores puede aún fallar en satisfacer expectativas o usabilidad.

1.4 Psicología de la prueba

Factores psicológicos. Las pruebas son realizadas por personas y están sujetas a factores humanos:

- **Sesgo de confirmación**: los desarrolladores pueden no ver errores en su propio código.
- **Sesgo cognitivo**: dificultad para aceptar resultados negativos.
- **Rechazo emocional**: los errores detectados pueden tomarse como crítica personal.

  Recomendaciones:

  - Enfocar la comunicación en hechos, no en personas.
  - Fomentar colaboración entre testers y desarrolladores.
  - Recordar el objetivo común: mejorar la calidad del producto.

Mentalidad del tester vs. desarrollador El **desarrollador** busca construir soluciones. Por otro lado, el **tester** busca detectar errores y validar el producto. Estas mentalidades son diferentes pero complementarias. Testers independientes aportan una perspectiva externa que mejora la detección de defectos.

1.5 Glosario de Términos Relevantes

- **Cobertura**: Nivel en que los elementos del software han sido ejercitados por un conjunto de pruebas, expresado en porcentaje.
- **Defecto**: Imperfección o deficiencia en un producto de trabajo que impide que cumpla con sus requisitos o especificaciones.
- **Depuración**: Proceso de localizar, analizar y corregir las causas de los fallos en el software.
- **Error**: Acción humana que da lugar a un resultado incorrecto.

- **Criterios de salida**: Conjunto de condiciones que deben cumplirse para considerar finalizada una tarea definida.
- **Prueba exhaustiva**: Enfoque de prueba que contempla todas las combinaciones posibles de valores de entrada y precondiciones.
- **Fallo**: Evento en el cual un componente o sistema no realiza una función requerida dentro de los límites establecidos.
- **Prueba funcional**: Prueba orientada a verificar si un componente o sistema satisface los requisitos funcionales especificados.
- **Prueba no funcional**: Prueba orientada a evaluar si un componente o sistema cumple con los requisitos no funcionales (como rendimiento, seguridad, usabilidad, etc.).
- **Prueba de regresión**: Prueba relacionada con cambios en el software, orientada a detectar defectos introducidos o descubiertos en áreas no modificadas del código.
- **Requisito**: Condición o capacidad que debe ser cumplida por un sistema o componente.
- **Riesgo**: Factor que podría derivar en consecuencias negativas, generalmente expresado en términos de impacto y probabilidad.
- **Pruebas basadas en riesgo**: Estrategia de pruebas donde las actividades se priorizan y ejecutan de acuerdo con los tipos y niveles de riesgo involucrados.
- **Causa raíz**: Origen principal de un defecto; si se elimina, disminuye o elimina la recurrencia de dicho defecto.
- **Calidad**: Grado en que un componente o sistema satisface las necesidades explícitas e implícitas de sus diversos interesados.
- **Aseguramiento de calidad**: Actividades orientadas a generar confianza en que los requisitos de calidad serán cumplidos.
- **Control de calidad**: Conjunto de actividades destinadas a evaluar la calidad de un componente o sistema.
- **Gestión de calidad**: Actividades coordinadas para dirigir y controlar una organización con respecto a la calidad, incluyendo planificación, control, aseguramiento y mejora.
- **Base de prueba**: Conjunto de información utilizada como base para el análisis y diseño de las pruebas.
- **Caso de prueba**: Conjunto de precondiciones, entradas, acciones, resultados esperados y postcondiciones, desarrollado a partir de condiciones de prueba.
- **Datos de prueba**: Información necesaria para la ejecución de pruebas.
- **Ejecución de pruebas**: Actividad que implica ejecutar un caso de prueba sobre un componente o sistema para producir resultados reales.
- **Registro de prueba**: Registro cronológico de detalles relevantes relacionados con la ejecución de las pruebas.
- **Objeto de prueba**: Componente o sistema que está siendo probado.
- **Objetivo de prueba**: Propósito o razón específica para realizar una prueba.
- **Oráculo de prueba**: Fuente utilizada para determinar los resultados esperados con los que se comparan los resultados reales del sistema probado.
- **Procedimiento de prueba**: Secuencia de casos de prueba con su orden de ejecución y cualquier acción adicional necesaria antes o después de la prueba.
- **Proceso de prueba**: Conjunto de actividades interrelacionadas que incluye planificación, monitoreo, análisis, diseño, implementación, ejecución y cierre de pruebas.
- **Suite de pruebas**: Conjunto de scripts o procedimientos de prueba a ejecutarse en una sesión determinada.
- **Artefactos de prueba (testware)**: Productos generados durante el proceso de prueba, utilizados para planificar, diseñar, ejecutar, evaluar y reportar las pruebas.

- **Pruebas**: Actividades del ciclo de vida destinadas a evaluar un componente o sistema y detectar defectos.
- **Trazabilidad**: Grado en que puede establecerse una relación entre dos o más productos de trabajo.
- **Validación**: Confirmación, mediante examen y evidencia objetiva, de que se han cumplido los requisitos para un uso o aplicación específica.
- **Verificación**: Confirmación, mediante examen y evidencia objetiva, de que se han cumplido los requisitos especificados.