CURSO COMPLETO ISTQB – NIVEL FOUNDATION (CTFL)

Este material está pensado como curso completo para prepararte para la certificación ISTQB Foundation Level, y también para usarlo como guía práctica en tu trabajo como QA.

- ISTQB

ISTQB

ISTQB

10/02/2026

Versión 0.1

ÍNDICE

1. Introducción al Testing de Software

    1.1 Objetivos principales del testing
    1.2 Testing vs Debugging

2. Principios Fundamentales del Testing

    2.1 Las pruebas muestran la presencia de defectos
    2.2 Las pruebas exhaustivas son imposibles
    2.3 Probar temprano
    2.4 Agrupación de defectos
    2.5 Paradoja del pesticida
    2.6 El testing depende del contexto
    2.7 Ausencia de errores no implica utilidad

3. Testing en el Ciclo de Vida del Software (SDLC)

    3.1 Niveles de prueba

4. Pruebas Estáticas

5. Técnicas de Diseño de Pruebas

6. Gestión del Testing

7. Herramientas de Soporte al Testing

8. Gestión de Defectos

    8.1 Conceptos Clave
    8.2 Objetivos
    8.3 Ciclo de Vida del Defecto
    8.4 Severidad vs Prioridad
    8.5 Buen Reporte de Defecto
    8.6 Métricas

9. Testing en Entornos Ágiles

    9.1 Rol del QA en Agile
    9.2 Testing dentro del Sprint
    9.3 Shift Left Testing
    9.4 Automatización en Agile
    9.5 Pirámide de Pruebas
    9.6 Beneficios del Testing Ágil

10. Preparación para el Examen ISTQB

1. Introducción al Testing de Software

El testing de software es un conjunto de actividades planificadas cuyo propósito es evaluar la calidad de un producto de software mediante la identificación de defectos, la verificación de requisitos y la validación del comportamiento esperado del sistema.

El testing no es solo una actividad de ejecución de pruebas, sino un proceso sistemático que incluye planificación, diseño, implementación, ejecución, evaluación de resultados y cierre.

1.1 Objetivos principales del testing

1. Detectar defectos antes de la liberación del producto.
2. Verificar que el sistema cumple los requisitos funcionales y técnicos.
3. Validar que el producto satisface las necesidades del usuario final.
4. Reducir riesgos técnicos, operativos y de negocio.
5. Proporcionar información confiable para la toma de decisiones sobre la liberación.

1.2 Testing vs Debugging

Testing	Debugging
Detecta defectos	Encuentra la causa del defecto
Lo realiza principalmente QA	Lo realiza principalmente desarrollo
Evalúa comportamiento	Corrige el problema

2. Principios Fundamentales del Testing

Los 7 principios ISTQB establecen la base teórica del testing:

2.1 Las pruebas muestran la presencia de defectos
- No demuestran que el sistema esté libre de errores.

2.2 Las pruebas exhaustivas son imposibles
- No se pueden probar todas las combinaciones de entradas y escenarios.

2.3 Probar temprano
- Detectar defectos en etapas iniciales reduce costos significativamente.

2.4 Agrupación de defectos
- La mayoría de defectos suele encontrarse en pocas áreas críticas.

2.5 Paradoja del pesticida
- Repetir los mismos casos pierde efectividad; las pruebas deben evolucionar.

2.6 El testing depende del contexto
- No se prueba igual un sistema médico que una red social.

2.7 Ausencia de errores no implica utilidad
- Un sistema puede funcionar correctamente pero no cumplir necesidades reales.

3. Testing en el Ciclo de Vida del Software (SDLC)

El testing se integra a lo largo de todo el ciclo de desarrollo.

1. Modelos de desarrollo
2. Cascada
3. V-Model (enfoque fuerte en pruebas)
4. Incremental
5. Ágil

3.1 Niveles de prueba

Nivel	Objetivo
Unitarias	Validar componentes individuales
Integración	Validar interacción entre módulos
Sistema	Evaluar el sistema completo
Aceptación	Validar contra requisitos del usuario

Tipos de prueba

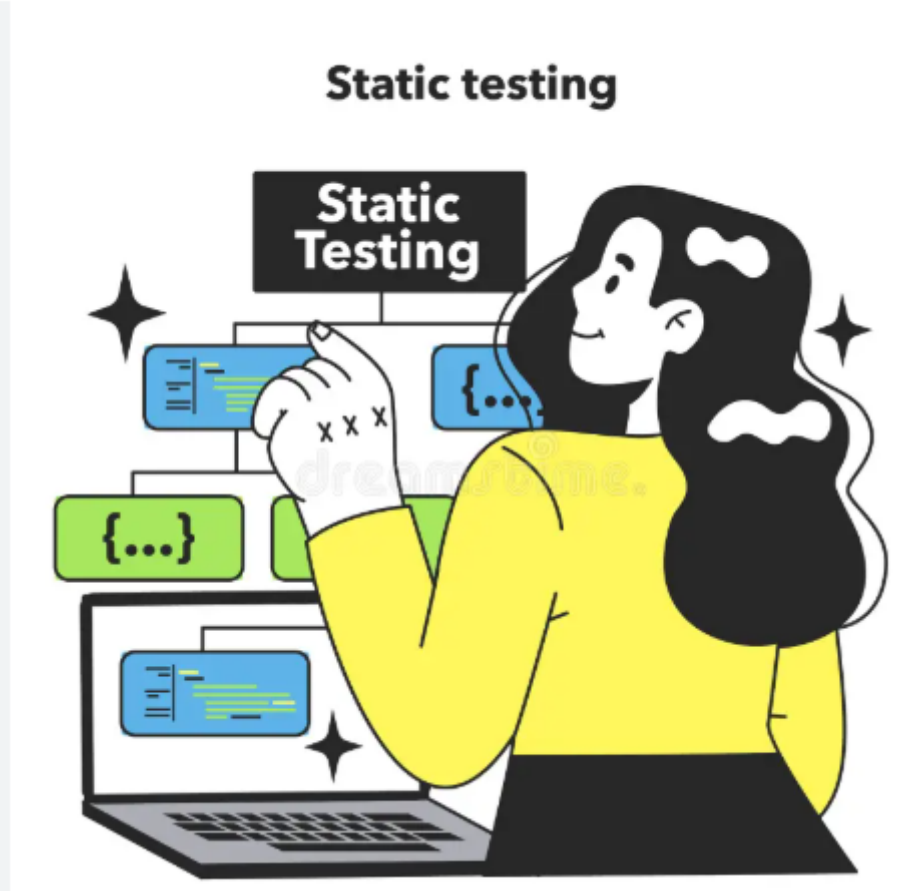
1. Funcionales
2. No funcionales (rendimiento, seguridad, usabilidad)
3. Estructurales
4. Regresión

5. Re-testing

4. Pruebas Estáticas

Son evaluaciones realizadas sin ejecutar el software.

1. Tipos de revisión
2. Revisión informal
3. Walkthrough
4. Inspección formal
5. Beneficios
6. Detección temprana de defectos
7. Menor costo de corrección
8. Mejora en la calidad de requisitos y diseño



5. Técnicas de Diseño de Pruebas

Permiten diseñar casos efectivos y optimizar cobertura.

1. Técnicas de Caja Negra
2. Clases de equivalencia
3. Análisis de valores límite
4. Tablas de decisión
5. Casos de uso
6. Técnicas de Caja Blanca
7. Cobertura de sentencias
8. Cobertura de decisiones
9. Basadas en experiencia
10. Exploratorio
11. Checklists
12. Pruebas de error guessing

6. Gestión del Testing

La gestión asegura control del proceso.

Incluye:

1. Planificación de pruebas
2. Estimación de esfuerzo
3. Identificación de riesgos
4. Monitoreo del progreso
5. Métricas de calidad
6. Ejemplos de métricas:
7. Tasa de defectos
8. Cobertura de pruebas
9. Casos ejecutados vs planificados

7. Herramientas de Soporte al Testing

Las herramientas no sustituyen al QA, pero aumentan eficiencia.

Categoría	Ejemplos
Gestión de pruebas	TestRail
Defectos	Jira
Automatización	Selenium
Performance	JMeter

8. Gestión de Defectos

La gestión de defectos es el proceso de registrar, analizar, dar seguimiento y cerrar los problemas detectados durante las pruebas. Su objetivo es asegurar la calidad del software mediante un control ordenado de incidencias.

8.1 Conceptos Clave

1. Error: equivocación humana.
2. Defecto (bug): problema en el código o diseño causado por un error.
3. Fallo: comportamiento incorrecto visible para el usuario.
4. Un error genera un defecto, y al ejecutarse produce un fallo.

8.2 Objetivos

1. Mantener trazabilidad de los problemas
2. Priorizar según impacto en el negocio
3. Facilitar la comunicación entre QA y desarrollo
4. Reducir riesgos antes de producción

8.3 Ciclo de Vida del Defecto

Estados comunes:

Nuevo → Asignado → En progreso → Resuelto → Verificado → Cerrado

También pueden existir: Reabierto, Rechazado o Duplicado.

8.4 Severidad vs Prioridad

1. Severidad: impacto técnico del defecto (lo define QA).
2. Prioridad: urgencia de solución (la define negocio o PM).

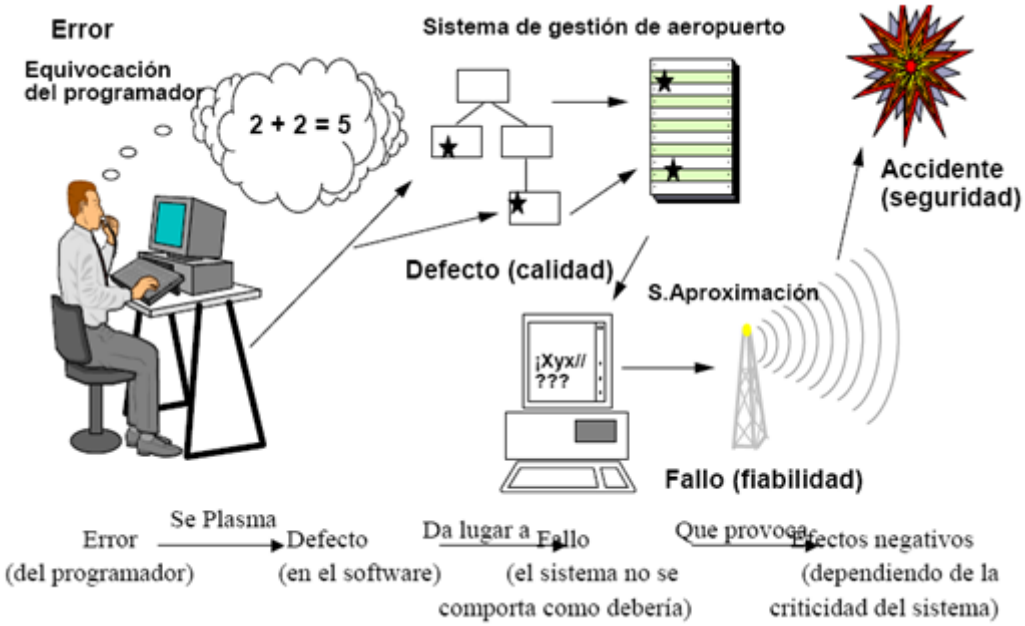
8.5 Buen Reporte de Defecto

Debe ser claro, reproducible y con evidencia.

Incluye:
ID, título, pasos, resultado esperado, resultado actual, entorno, evidencia, severidad y prioridad.

8.6 Métricas

1. Defectos por módulo
2. Defectos reabiertos
3. Tiempo de resolución



9. Testing en Entornos Ágiles

En metodologías ágiles, el testing es una actividad continua dentro de cada Sprint, no una fase final. El QA trabaja desde el inicio junto al equipo para asegurar calidad en ciclos cortos y detectar defectos lo antes posible.

9.1 Rol del QA en Agile

1. El QA cumple un rol activo y preventivo.
2. Participa en historias de usuario
3. Define y valida criterios de aceptación

4. Identifica riesgos funcionales
5. Diseña casos de prueba
6. Apoya la automatización
7. Verifica el cumplimiento de la Definition of Done (DoD)
8. El QA es un facilitador de calidad, no solo un ejecutor de pruebas.

9.2 Testing dentro del Sprint

1. El testing ocurre durante todo el Sprint:
2. Planificación: análisis de historias y escenarios de prueba
3. Desarrollo: pruebas continuas y exploratorias
4. Cierre de historia: validación funcional y regresión
5. Review: confirmación de estabilidad y calidad

9.3 Shift Left Testing

Consiste en probar lo antes posible en el ciclo de desarrollo.

Beneficios:

1. Menor costo de corrección
2. Menos retrabajo
3. Mejor comprensión de requisitos
4. Mayor estabilidad

9.4 Automatización en Agile

Es clave para mantener la velocidad:

1. Regresión
2. APIs
3. Funcionalidades críticas
4. Reduce tiempos y permite validar cambios frecuentes.

9.5 Pirámide de Pruebas

Distribución recomendada:

1. Base: pruebas unitarias
2. Medio: integración / APIs
3. Punta: UI / End-to-End

Se busca depender menos de pruebas lentas de interfaz.

9.6 Beneficios del Testing Ágil

Detección temprana de defectos

1. Mejor calidad
2. Entregas frecuentes
3. Menor riesgo en producción
4. Mayor alineación con el negocio

10. Preparación para el Examen ISTQB

1. 40 preguntas
2. 60 minutos
3. 65% mínimo
4. Se evalúa comprensión conceptual, no memorización literal.

11. Glosario

Incluye términos clave del estándar ISTQB.

12. Aplicación en el Trabajo Real

Los fundamentos del ISTQB se aplican directamente en las actividades diarias del QA

12.1 Diseño de Casos de Prueba

Permiten crear casos estructurados, cubrir escenarios positivos y negativos, y usar técnicas como equivalencia y valores límite para mejorar la cobertura.

12.2 Pruebas de Regresión

Ayudan a identificar áreas críticas, priorizar pruebas según riesgo y apoyar la automatización para evitar que nuevos cambios afecten funciones existentes.

12.3 Análisis de Riesgos

El QA enfoca esfuerzos en los módulos con mayor impacto en el negocio, optimizando el tiempo de prueba.

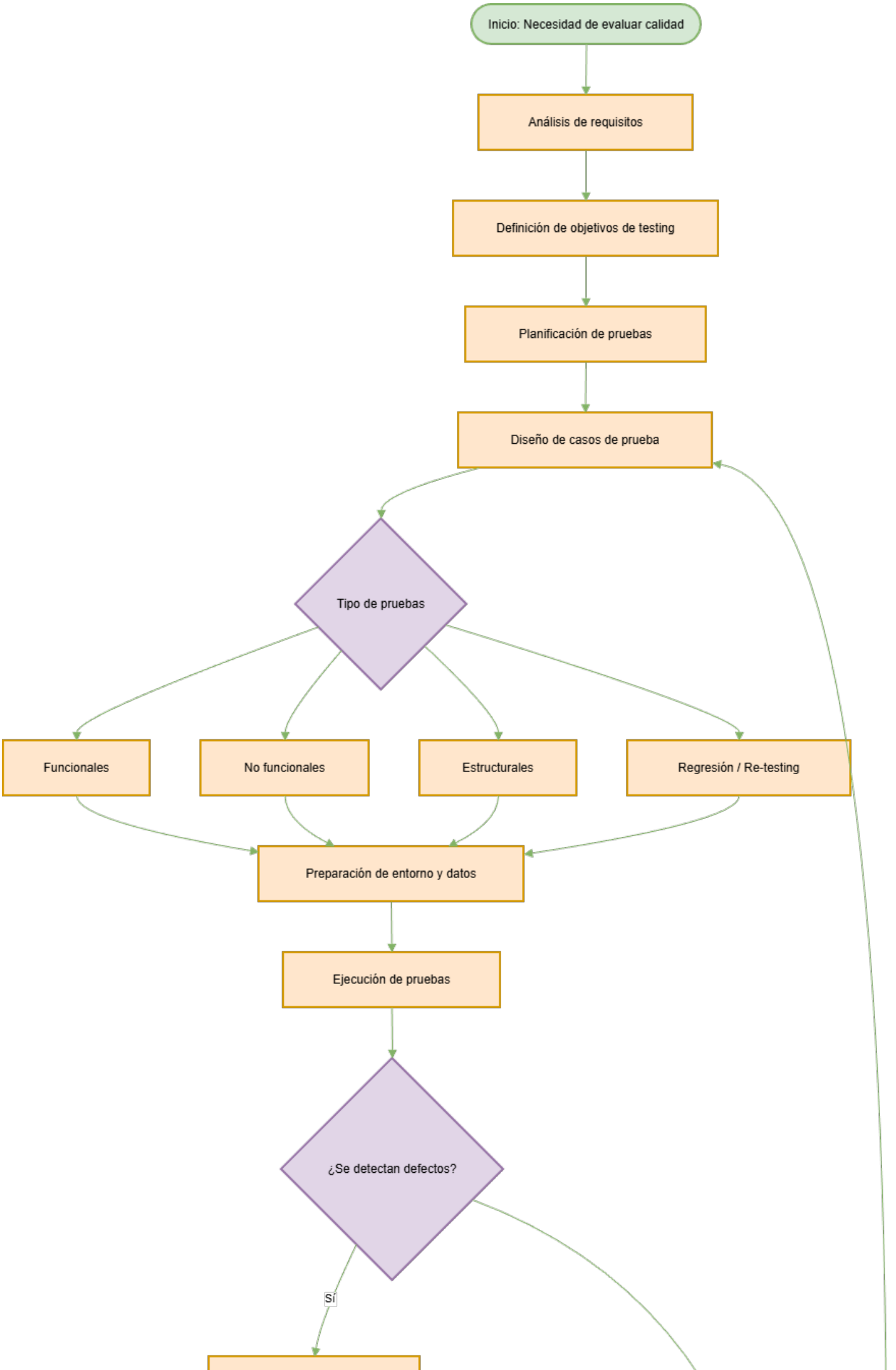
12.4 Reporte Profesional de Defectos

Se aplican buenas prácticas: pasos claros, evidencias, correcta severidad/prioridad y comunicación objetiva con el equipo.

12.5 Mejora Continua

Se ajustan estrategias, se optimizan procesos y se fortalecen estándares de calidad de forma constante.

13. Diagrama de Flujo


Inicio: Necesidad de evaluar calidad
Análisis de requisitos
Definición de objetivos de testing
Planificación de pruebas
Diseño de casos de prueba
Tipo de pruebas
Funcionales
No funcionales
Estructurales
Regresión / Re-testing
Preparación de entorno y datos
Ejecución de pruebas
¿Se detectan defectos?
Sí