WWW.FULLADVANCED.COM

FUNDAMENTOS DE LAS PRUEBAS

DE SOFTWARE

BASADO EN EL PROGRAMA PROBADOR
CERTIFICADO ISTQB V4 - NIVEL BÁSICO.

DISEÑO Y CONTENIDO:
KELLY AGUILAR ZAMBRANO
REVISIÓN: JULIO CESAR OROPEZA

AGOSTO 2023

¡Hola! Esperamos que este contenido sea de mucha ayuda para comprender de manera más ligera los fundamentos que todo probador a nivel básico debe poseer.

Este ebook es una muestra de 50 páginas de las 122 del nuevo ebook ajustado al Syllabus V4. En el material completo podrás obtener muchos gráficos, ejemplos y tips que te ayudarán si tu plan es presentar el examen internacional para el CTFL ISTQB 2023, incluso si te estás preparando para un entrevista de trabajo sin importar tu nivel de experiencia.

Para obtener el ebook completo ajustado el Syllabus V4, puedes acceder a *www.fulladvanced.com/ebooks*, y además obtendrás un resumen para comprender mejor el proceso de pruebas explicado paso a paso con ejemplos de productos de trabajo, y una guía de Scrum explicada al estilo Full Advanced 😉.

Éxito en tus planes profesionales, nos vemos en el futuro 😀.

Julio César y Kelly, Equipo Full Advanced.

CAP. 1 • FUNDAMENTOS DE LA PRUEBA

Existen **2 errores comunes sobre las pruebas** de software:

1. Sólo consisten en ejecutar el software y comprobar resultados.
2. Se enfocan en revisar que el software cumpla con la verificación de requerimientos.

Probar no es sólo correr pruebas.

Verificación:

¿Construímos bien el software según los requerimientos?

Validación:

¿Construímos el software adecuado basado en las necesidades reales del cliente?

PROCESO DE PRUEBAS

La definición de Pruebas de Software la dividimos en 2 partes: Proceso y Objetivos.

El Proceso

- Probar es un proceso no una actividad, se realiza durante todo el *ciclo de vida de desarrollo*, abarcando las pruebas estáticas y dinámicas.
- Las pruebas se deben preparar. Se planifica antes, durante y después de la ejecución de las pruebas.
- Los resultados de las pruebas del código y los productos de trabajo relacionados deben ser verificados.

Objetivos

- Se **verifica** y se **valida** el cumplimiento de los requisitos.
- Al detectar defectos comprendemos riesgos y mejoramos la calidad de los productos.
- Y cuando analizamos la **causa raíz** mejoramos los Procesos de Desarrollo.

OBJETIVOS DE LAS PRUEBAS

- Validar que todos los requerimientos especificados son correctos, incluidos los de ámbito legal
- Verificar que todos los requerimientos especificados han sido satisfechos, incluidos los de ámbito legal
- Evaluar que todos los productos de trabajo estén correctos
- Encontrar y prevenir fallas y defectos
- Garantizar la cobertura de pruebas requerida
- Reducir el nivel de riesgo de software de baja calidad
- Proporcionar información para la toma de decisiones
- Generar confianza en la calidad el software

Los objetivos de la prueba varían dependiendo del sistema que se esté probando, en qué fase del ciclo de desarrollo se está, o del nivel de prueba.

Producto de trabajo

Un producto de trabajo es cualquier documento, informe, reporte, diagrama e inclusive el código fuente o el sistema terminado.

Probar es distinto de depurar

Debugging o Depuración es el proceso de desarrollo que encuentra, analiza y repara el defecto.

Probar es verificar que el software funciona como se espera.

El ciclo de depuración implica que el desarrollador depura el código, corrige el código y realiza sus pruebas.

Luego del ciclo de depuración, un probador independiente ejecuta **las pruebas de confirmación.**

Las pruebas de verificación se realizan para garantizar que se cumpla lo establecido en los requerimientos.

¿POR QUÉ ES NECESARIO PROBAR EL SOFTWARE?

1. Todo el mundo comete errores, los errores introducen defectos que pueden generar fallas.

2. Reduce el riesgo y aumenta la confianza.

3. Algunos defectos son difíciles de identificar debido a que se originan en suposiciones erradas o puntos ciegos.

4. Algunas pruebas son obligatorias debido a estándares legales o comerciales.

¿ Cómo contribuye la prueba al éxito de un proyecto?

1. Cuando los probadores y diseñadores trabajan en equipo se puede aumentar la comprensión del diseño del sistema y cómo probarlo.

2. Cuando los probadores y desarrolladores trabajan en equipo mientras el código se está desarrollando, le permite al probador ubicar segmentos de la aplicación susceptibles a error.

3. Tener probadores verificando y validando el software antes de su liberación aumenta la probabilidad de que el software cumpla con las expectativas del usuario.

4. Que el software cumpla con las expectativas del usuario aumenta la confianza en el producto, y la confianza en el producto genera fidelidad al proveedor.

Aseguramiento de la Calidad y *Pruebas* son conceptos distintos

El Aseguramiento de la Calidad se centra en el cumplimiento de los procesos adecuados para generar productos de trabajo de mayor calidad.

El Control de la Calidad implica varias actividades, incluidas las actividades de prueba, lo que apoya el logro de niveles apropiados de calidad.

La Gestión de Calidad une a ambos conceptos: Aseguramiento y el Control de la calidad.

Full Advanced

ERROR, DEFECTO Y FALLA

www.FullAdvanced.com por Julio César Oropeza

¡Recuerda no confundir conceptos!

Error: Error, Mistake

Defecto: Falta,Defect, Bug, fault, fail.

Falla: Failure.

¿Por qué ocurren errores?

- Trabajo bajo presión
- Falibilidad humana
- Falta de experiencia del personal
- Complejidad del proyecto
- Falta de comunicación

Causa Raíz de un defecto

Cuando se detecta una falla, debemos rastrearla para determinar la razón por la cual ocurrió. Los defectos deben ser analizados para identificar la raíz de su origen, y así reducir la posibilidad de que se repita en el futuro.

Un humano comete un **ERROR** que puede introducir un **DEFECTO** que podría generar una **FALLA**

- No todos los defectos causarán fallas.
- No todas las fallas son de origen humano, pueden ser de origen ambiental.
- No todos los resultados inesperados de las pruebas son fallas.

Existen **falsos positivos** y **falsos negativos** en los resultados de las pruebas.

Si la prueba dice que hay un error cuando en realidad no lo hay, es un **Falso positivo**.

Si la prueba no detecta defectos que debió haber identificado, es un **Falso negativo**.

Oráculo de prueba es la fuente para determinar resultados esperados para compararlos con los resultados reales del software en pruebas.

SIETE PRINCIPIOS DE LAS PRUEBAS

1. Las pruebas demuestran que hay errores, no su ausencia
2. Es imposible probarlo todo
3. Probar desde el inicio del ciclo ahorra tiempo, esfuerzo y dinero
4. Los defectos se agrupan
5. Los casos de prueba deben actualizarse para encontrar nuevos defectos
6. Las pruebas dependen del sistema que se esté probando
7. Es imposible que no hayan errores

La creencia de que el software debe ser liberado sin errores y con absoluta correctitud es algo imposible de lograr. Sin embargo, se puede reducir el riesgo de falla en ambientes operativos haciendo pruebas con estándares altos de calidad.

PROCESO DE PRUEBAS: Actividades

Planificación y Control: ¿Cuales son los objetivos y cómo se alcanzan?

En la **planificación** se debe asegurar la comprensión de las expectativas del usuario, cuáles son sus metas, objetivos y los riesgos que se deberán considerar. Se establecen las bases de prueba y el plan de pruebas a seguir. El **control o monitoreo** verifica que las actividades se están llevando a cabo según el plan y se realiza durante todo el proceso de pruebas.

Análisis: Se analizan los requerimientos levantados a fin de verificar entre otras cosas, factibilidad de la prueba, tiempo y recursos necesarios. Se responde a la pregunta: **¿Qué probar?**

Diseño de los casos de prueba de alto nivel. Un caso de prueba es un grupo de valores, precondiciones y resultados esperados. Se debe responder **¿Cómo probar?**

Implementación: Aquí se crean o se completan los productos de prueba necesarios para la ejecución de la prueba.

A menudo implementación y ejecución son etapas combinadas, en esta etapa se debe preguntar **¿Está todo listo para ejecutar las pruebas?**

Ejecución: Se ejecutan los casos de prueba, se registran y reportan los resultados de la prueba.

Cierre o compleción de la prueba: Finalmente, se libera el software cuando los criterios de aprobación han sido satisfechos y se ejecutan las actividades de cierre del proyecto.

PROCESO DE PRUEBAS: Actividades

Planificación → Análisis → Diseño → Construcción → Pruebas → Cierre

Aunque parezca que el proceso de pruebas sigue una serie secuencial de actividades, algunas de ellas pueden llevarse a cabo iterativamente, como por ejemplo la implementación y la ejecución de las pruebas.

PROCESO DE PRUEBAS: Productos de Trabajo

Insumo	Actividad	Salida
Bases de Prueba	Actividades de Planificación	Plan de Pruebas
Plan de Prueba	Control y Monitoreo	• Reporte de progreso de pruebas • Reporte de resumen de pruebas
Bases de Prueba	Análisis de Pruebas	Condiciones de Prueba
Condiciones de Prueba	Diseño de Pruebas	• Escenarios • Conjuntos de Prueba • Data de Prueba

PROCESO DE PRUEBAS: Productos de Trabajo

Insumo	Actividad	Salida
Escenarios Conjuntos de Prueba Data de Prueba	Implementación de Pruebas	• Casos de prueba • Procedimientos • Juegos de prueba • Calendarios de ejecución de pruebas
• Casos de prueba • Juegos de prueba o Suites de prueba • Calendarios de ejecución de pruebas	Ejecución de Pruebas	• Reporte de resultado de pruebas • Reporte de defectos encontrados
• Reporte de progreso de pruebas • Reporte de resumen de pruebas	Cierre de Pruebas	Reportes a los interesados

Trazabilidad entre Bases de Prueba y Productos de Trabajo

Etapa	Entradas	Objetivos	Actividades	Salidas	Trazabilidad
Planificación de pruebas	Bases de Prueba	Definir los objetivos de las pruebas Establecer la estrategia de pruebas: técnicas a usar, tareas, fechas de entrega	Definir alcance, riesgos, enfoque y calendario de pruebas Establecer cómo medir el avance, quién hace qué y cómo lo hace Integrar las pruebas al ciclo de vida	Estrategia Plan de pruebas	A partir del plan de pruebas se genera trazabilidad hacia la totalidad de los productos de trabajo, ya que esta es la piedra angular del proyecto de pruebas

Trazabilidad entre Bases de Prueba y Productos de Trabajo

Etapa	Entradas	Objetivos	Actividades	Salidas	Trazabilidad
Monitoreo y Control de pruebas	Cronograma de planificación Resultado de pruebas Reportes de análisis de riesgo	Verificar que el plan de pruebas cumpla con las fechas, estándares de calidad y demás especificaciones establecidas Tomar acciones correctivas en caso de desvíos en la planificación	Revisar plan de pruebas vs progreso real Revisar resultados de pruebas vs. Criterios de salida (definición de hecho) Determinar si más pruebas son necesarias Comunicar resultados a las partes interesadas	Reporte de progreso de pruebas Reporte resumen de pruebas	Casos de pruebas Vs Calendario de pruebas Casos de pruebas Vs Riesgos identificados Casos de prueba vs Cobertura establecida
Análisis de pruebas	Bases de Prueba	Evaluar las bases de prueba Determinar qué probar	Definir y priorizar las condiciones de prueba Validan requisitos Verifican requisitos	Condiciones de prueba Niveles de riesgo	Bases de prueba vs. Condiciones de prueba
Diseño de pruebas	Condiciones de Prueba Estrategia de prueba Plan de prueba	Transformar las condiciones de prueba en Escenarios. Determinar cómo probar	Definir y priorizar casos de prueba | Escenarios, y conjuntos de casos de prueba Identificar data, herramientas e infraestructura Diseñar el ambiente de prueba	Casos de prueba de alto nivel (Lógicos) | Escenarios Conjuntos de casos de Prueba y Procedimientos Identificación de datos de prueba	Condiciones de prueba vs. Casos de prueba de alto nivel | Escenarios
Implementación de pruebas	Casos de prueba de alto nivel | Escenarios	Crear o completar los productos de prueba necesarios para la ejecución de la prueba Determinar si está todo listo para ejecutar las pruebas	Desarrollar los casos de prueba de bajo nivel Desarrollar el ambiente de pruebas Desarrollar y priorizar procedimientos de prueba |Test procedures Desarrollar las Suites | Juegos de Pruebas	Casos de prueba de bajo nivel (concretos) Suites | Juegos de prueba Test Procedures | Procedimientos de Prueba Cronograma de ejecución de pruebas	Escenarios Vs. Casos de prueba de bajo nivel
Ejecución de pruebas	Suites/Juegos de prueba	Ejecutar las suites de pruebas de acuerdo al calendario de ejecución de pruebas	Registrar los identificadores y las versiones de los objetos de prueba Analizar causa raiz de defectos Comparar resultados obtenidos con los esperados: Oráculo de prueba	Reporte de resultado de pruebas Reporte de defectos encontrados	Defectos encontrados Vs. Cobertura establecida Defectos encontrados Vs. Riesgos identificados
Cierre de pruebas	Reporte de Resultados	Recopilar datos de actividades realizadas	Revisar que todos los defectos reportados hayan sido cerrados Almacenar los productos de trabajo que puedan ser reutilizables Analizar oportunidades de mejora	Reporte a los interesados |Stakeholders	Pruebas realizadas Vs. Definición de hecho

BUENAS PRÁCTICAS EN LAS PRUEBAS

Los desarrolladores y probadores tienen esquemas de pensamiento diferente

Desarrollador | Developer: Se enfoca en diseñar y construir un producto.

Probador | Tester: Se enfoca en verificar y validar el producto, encontrando tantos defectos como sea posible antes de la liberación a producción.

Maneras de comunicarse asertivamente entre equipos de desarrollo y Pruebas

- Recordar que es una colaboración, no una batalla entre bandos
- Enfatizar los beneficios de las pruebas
- Comunicar los defectos encontrados de una manera neutral, sin criticar al creador del producto de trabajo defectuoso
- Tratar de entender cómo se siente la otra persona
- Confirmar que el interlocutor ha entendido lo que se ha dicho, y viceversa

Roles de las pruebas

Se distinguen dos roles: uno encargado de la **gestión de prueba**, y el encargado de los aspectos técnicos de las pruebas.

Dependiendo del contexto, el líder de prueba responsable de la gestión de pruebas, se encarga de la planificación, monitoreo, control y cierre de las pruebas.

Mientras que el rol de prueba, responsabilidad del tester, se centra en el análisis, diseño, implementación y ejecución de pruebas.

Es común que algunas personas puedan percibir el proceso de prueba como una actividad destructiva, por esta razón probadores y líderes/ jefes de prueba deben saber comunicar de forma asertiva los resultados de las pruebas y construir relaciones positivas con sus colegas.

BUENAS PRÁCTICAS EN LAS PRUEBAS

Pruebas Independientes

Los probadores independientes son personas distintas al autor del producto de trabajo. Son necesarias para reducir el sesgo de confirmación y porque probadores independientes pueden encontrar diferentes tipos de defectos y fallas.

Hay niveles de independencia, de menor a mayor:

- Pruebas realizadas por la persona que creó el producto bajo prueba
- Pruebas realizadas por otra persona del mismo equipo
- Pruebas realizadas por una persona de un equipo diferente
- Pruebas realizadas por una persona de una empresa diferente

Ventajas de las pruebas independientes

Con probadores independientes aumenta la efectividad de las pruebas, lo cual es particularmente importante en grandes proyectos, o sistemas de seguridad crítica.

Desventajas de las pruebas independientes

El programador podría dejar toda la carga de las pruebas en el probador.

El probador puede aislarse del resto del equipo y tener problemas para alinearse a los objetivos del negocio.

Puedes ver el curso completo gratis en Youtube

HABILIDADES NECESARIAS PARA SER TESTER

Los probadores de software necesitan habilidades como el pensamiento crítico, la resolución de problemas, la comunicación efectiva y la capacidad de trabajar en equipo. Además, el entorno tecnológico en constante cambio significa que los probadores de software también deben mantenerse actualizados con las últimas herramientas y tendencias en la industria, con lo cual su capacidad autodidacta debe estar bien desarrollada.

Listemos algunas habilidades necesarias para ser un tester:

Habilidades Técnicas

- Conocimiento de técnicas de prueba para que puedas garantizar la mayor cobertura de pruebas posible con los recursos disponibles.
- Conocimientos de herramientas de prueba para que puedas ejecutar pruebas de forma apropiada según el contexto.
- Conocimiento del negocio para poder comprender qué debes probar y comunicarte con los usuarios finales.
- Como un plus, conocimientos básicos de programación para crear casos de prueba más eficientes.

Habilidades Blandas

- Excelente comunicación: debes poder comunicarte de forma clara y respetuosa con el equipo, ajustando el vocabulario al nivel técnico del interlocutor, tanto de forma verbal como escrita. Deberás escribir informes de prueba que deben ser suficientemente descriptivos y concisos a la vez, para lo cual debes tener dominio de la gramática y semántica del idioma en el que trabajas.

HABILIDADES NECESARIAS PARA SER TESTER

- La comunicación incluye la habilidad de escuchar activamente a los miembros del equipo y a los usuarios para comprender sus necesidades y preocupaciones.

  Recuerda que eres el portador de las "malas noticias", por esto tus habilidades de comunicación son determinantes para trabajar en equipo y manejar las diferencias de opinión de manera constructiva buscando soluciones en lugar de centrarse en los problemas.

- Pensamiento analítico: debes evaluar posibles escenarios de pruebas, analizar el sistema en búsqueda de defectos, priorizar pruebas, identificar patrones en los datos de prueba, comprender las relaciones causa-efecto de algunos defectos, optimizar recursos, evaluar el impacto de los defectos encontrados, proporcionar información precisa y coherente a los interesados, y tomar decisiones basadas en la evidencia y los datos disponibles. Además el pensamiento analítico es fundamental para la comunicación con los desarrolladores, saber que preguntas hacer, cuando hacerlas es esencial para comprender detalles técnicos y diseñar pruebas sólidas.

- Curiosidad para explorar el software bajo prueba, prestar especial atención a los detalles y ser metódico en la identificación y reporte de defectos en el software.

- Adaptabilidad y mejora continua: Debes estar dispuesto a ajustarse a los cambios en el entorno de desarrollo, las prioridades del proyecto y las metodologías de trabajo. Esto incluye aprender nuevas tecnologías, herramientas y enfoques de prueba a medida que evoluciona la industria de forma autodidacta.

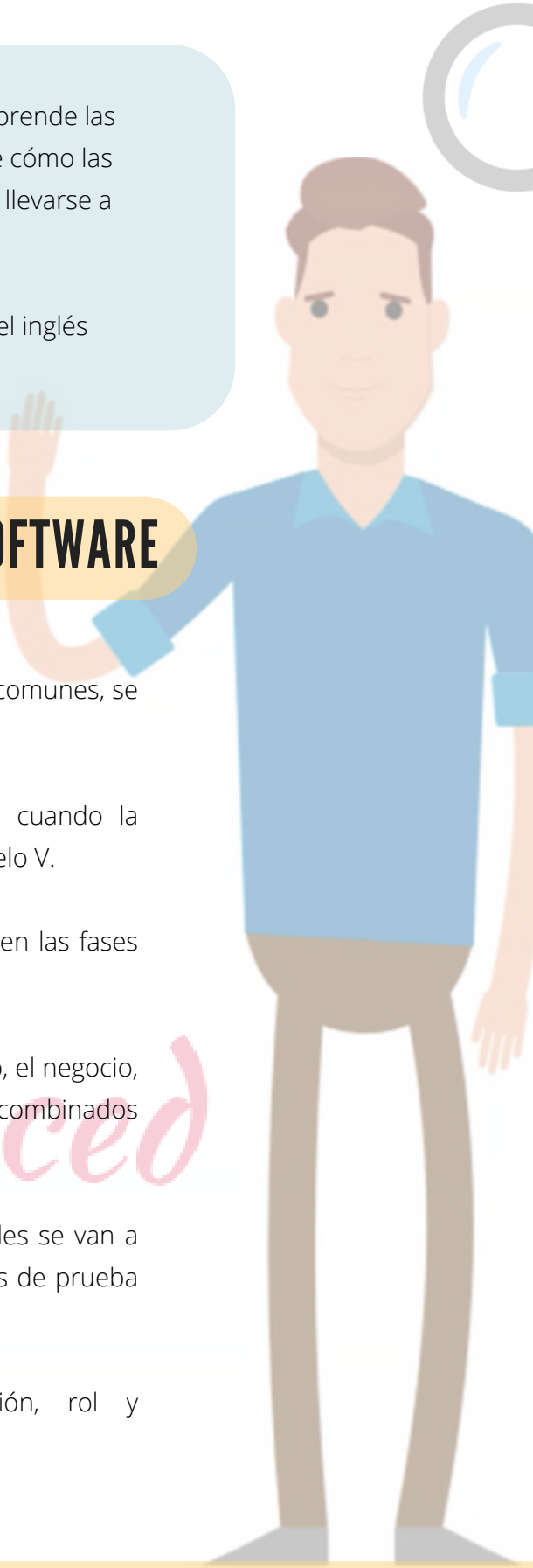
- Manejo eficazmente el tiempo, los recursos y la documentación de pruebas para mantener un proceso de prueba eficiente.

- Aunque no tengas un rol de liderazgo formal, un buen tester debe tener la capacidad de asumir la responsabilidad, mostrar iniciativa y motivar a otros en el equipo.

CAP. 2 • LAS PRUEBAS A TRAVÉS DEL CICLO DE DESARROLLO DE SOFTWARE

El modelo de ciclo de vida de desarrollo de software comprende las etapas por las cuales avanza un proyecto de software, describe cómo las actividades se relacionan entre sí, y en qué momento deberían llevarse a cabo.

A menudo lo encontrarás representado bajo las siglas **SDLC**, del inglés *Software Development Life Cicle*.



MODELOS DE CICLO DE DESARROLLO DE SOFTWARE

Los modelos de ciclo de vida de desarrollo de software (SDLC) más comunes, se clasifican en:

Modelos de desarrollo secuenciales: cada fase comienza cuando la anterior ha finalizado, por ejemplo el modelo en cascada, o modelo V.

Modelos de desarrollo iterativos e incrementales: se repiten las fases tantas veces como sea necesario, por ejemplo, Scrum.

Los modelos se seleccionan en base a las metas, el tipo de producto, el negocio, y los riesgos asociados entre otras consideraciones, y pueden ser combinados entre sí.

El SDLC escogido determina el alcance de la prueba, qué actividades se van a desarrollar, en qué niveles se va a probar, qué tipos qué y técnicas de prueba se van a aplicar.

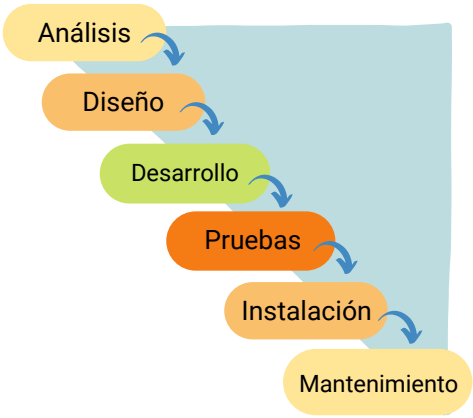
Además, impacta el nivel de detalle de la documentación, rol y responsabilidades del probador, y cuánto se va a automatizar.

MODELO DE DESARROLLO: SECUENCIAL

MODELO CASCADA:

Los modelos secuenciales son aquellos donde cualquier fase del proceso de desarrollo comienza una vez que la anterior ya finalizó. Por ejemplo, el modelo en cascada.

En teoría no hay solapamiento de fases y es beneficioso tener retroalimentación temprana de la fase siguiente. Es decir, si hablamos de las fases de Análisis y Diseño, en cuanto llegamos a la etapa de Diseño deberíamos obtener retroalimentación sobre cómo fue el desenvolvimiento de la etapa de Análisis.

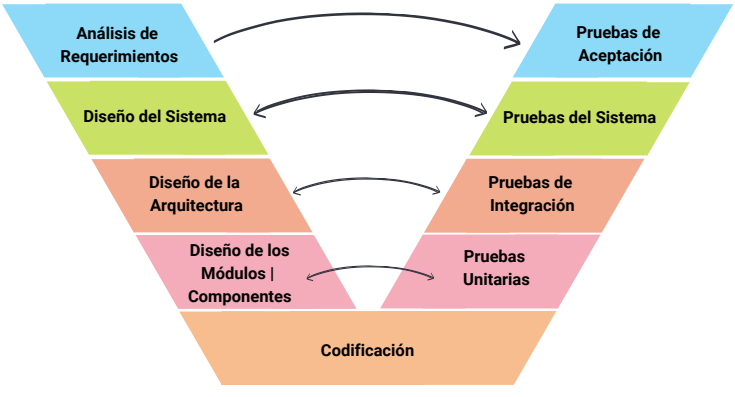


MODELO DE DESARROLLO EN CASCADA

MODELO V:

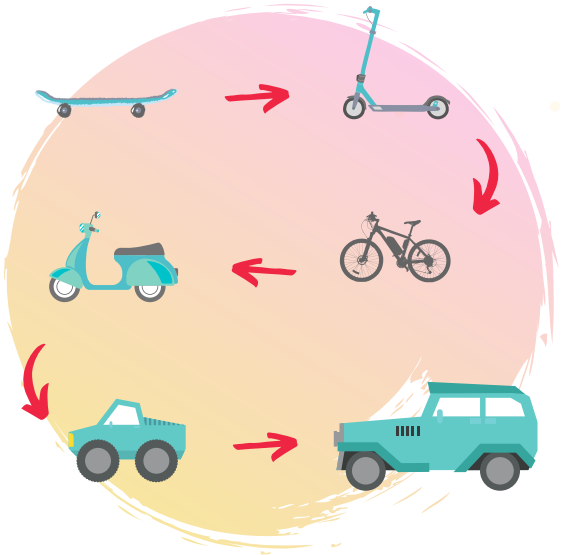
El **Modelo V**, es una evolución del modelo en cascada que integra el proceso de pruebas al proceso de desarrollo, para cada etapa de desarrollo tenemos una etapa de pruebas asociada.

En el siguiente gráfico se puede ver el detalle:



MODELO V

MODELO DE DESARROLLO: ITERATIVO INCREMENTAL

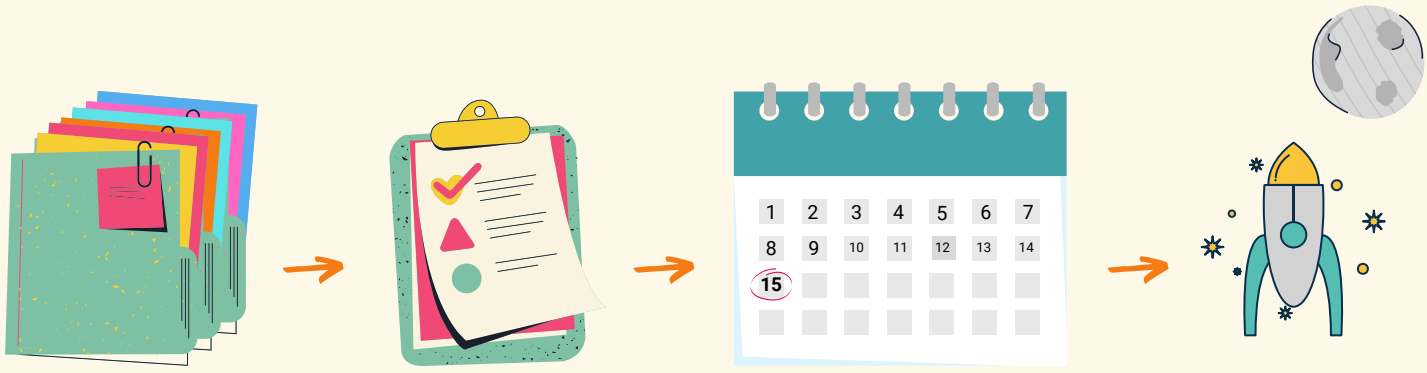


Se comienza desarrollando una especie de versión *light* de los requisitos del sistema, y en cada iteración se agregan nuevas funcionalidades. Se presume que al final de la iteración obtuvimos un producto funcional, no solamente un requerimiento desarrollado que por sí mismo no es operativo.

SCRUM:

Un ejemplo de Modelo Iterativo Incremental es SCRUM. Tiene como objetivo hacer entregas de funcionalidades terminadas en periodos cortos de tiempo. Cada iteración tiende a ser relativamente corta: horas, días o un par de semanas. Las entregas suelen abarcar dos o tres nuevas funcionalidades.

En resumen, se tiene una lista de requerimientos, de la cual se escogen los que se van a desarrollar durante el periodo de tiempo establecido, y que resultarán en una funcionalidad lista.



Product Backlog

Lista de requerimientos planificados para todo el proyecto

Sprint Backlog

Lista de requerimientos planificados para el *Sprint*

Sprint

Ventana de tiempo de 15 días, incluso 24 horas

Incremento

Funcionalidad lista

NIVELES DE PRUEBAS

Las pruebas de software son un conjunto de **actividades** organizadas y gestionadas en distintos niveles y relacionadas con su respectiva actividad de desarrollo.

Los Niveles de Prueba describen el objetivo de la prueba, y en qué momento se va a realizar.

Los tipos de prueba determinan qué se va a hacer para cumplir dichos objetivos. Este punto se verá mas adelante.

Desarrollador

- **Pruebas Unitarias o de Componentes**
- **Pruebas de Integración de Componentes**

Probador

- **Pruebas de Sistema**
- **Pruebas de Integración de Sistemas**
- **Pruebas de Aceptación**

NIVELES DE PRUEBAS

1. **Pruebas de Componentes**
2. **Pruebas de Integración**:
   - de Componentes
   - de Integración de Sistema
3. **Pruebas de Sistema**
4. **Pruebas de Aceptación:**
   - Pruebas de Aceptación de Usuarios,
   - Pruebas de Aceptación Operativa,
   - Pruebas de Aceptación Contractual y de Regulación,
   - Pruebas Alfa y Beta.

De la tabla anterior preste especial atención a qué pruebas realiza el desarrollador, y cuáles son responsabilidad del tester.

BUENAS PRÁCTICAS PARA LA EJECUCIÓN DE PRUEBAS

- Cada ciclo de desarrollo debe contar con pruebas de validación y verificación
- Cada actividad de desarrollo tiene una actividad correspondiente de pruebas
- Cada nivel de pruebas tiene objetivos específicos
- El análisis y diseño de cada nivel de pruebas debe comenzar durante la actividad de desarrollo que corresponda
- Los probadores deben participar en las discusiones de definición y refinamiento de requerimientos y diseño

COTS: son las siglas de Commercial off-the-shelf o software comercial de distribución masiva, por ejemplo Microsoft Office o Adobe Photoshop.

Puedes ver el capítulo completo gratis en Youtube

Diseño y contenido: Kelly Aguilar Zambrano

NIVELES DE PRUEBAS

Nivel	Objetivos	Objetos de Prueba	Bases de Prueba	Defectos típicos	Responsables
Componentes	Conseguir defectos y fallas en objetos de prueba Reducir el riesgo Construir confianza en la calidad del componente Prevenir defectos que escalen a niveles de prueba más altos	Código / Estructuras de datos Clases Módulos de bases de datos	Especificaciones de Componentes Código Modelo de datos	Funcionamiento no acorde a la especificación Problemas de flujos de datos Código y lógica incorrecta	Realizado por el desarrollador
Integración	Conseguir defectos y fallas entre las interfaces de módulos y sistemas, así como en los módulos en sí mismos Reducir el riesgo Construir confianza en la calidad de las interfaces Prevenir defectos que escalen a niveles de prueba más altos	Subsistemas Bases de Datos Infraestructura Interfaces Interfaces de comunicación de aplicaciones API's Microservicios	Diseño de software y sistemas Diagrama de secuencias Especificaciones de interfaz y protocolos de comunicación Casos de uso Arquitectura a nivel de componentes o sistemas Flujos de trabajo Definiciones de interfaces externas	Datos incorrectos, faltantes o mal codificados Secuenciación o sincronización incorrecta a las llamadas de interfaz Incompatibilidad de la interfaz Fallos en la comunicación de componentes Fallos en la comunicación entre componentes manejados inadecuadamente o no manejados en absoluto Suposiciones incorrectas acerca del significado, unidades o límites de datos transmitidos entre componentes	Las pruebas de integración de *componentes* son a menudo responsabilidad de los desarrolladores Las pruebas de integración de *sistemas* son responsabilidad de los probadores
Sistemas	Reducir el riesgo Verificar que los requerimientos del sistema son los diseñados y a su vez validarlos con lo especificado por el usuario Validar que el sistema está completo y funciona como se esperaba Generar confianza en la calidad del sistema en su conjunto Encontrar defectos y fallas Evitar que los defectos escapen a niveles de prueba más altos, como las pruebas de aceptación o producción.	Aplicaciones Sistemas de hardware y software Sistemas operativos Sistema bajo prueba Configuración del sistema y datos de configuración	Especificaciones de Requisitos del sistema Informes de análisis de riesgo Casos de Uso Épicas e historias de usuarios Modelos de comportamiento del sistema Diagramas de estado Manuales de Sistema y de Usuario	Cálculos incorrectos Comportamiento incorrecto de tareas del sistema Control de datos o flujos de datos incorrectos dentro del sistema Funcionamiento incorrecto del sistema	Realizadas generalmente por probadores independientes

NIVELES DE PRUEBAS

Nivel	Objetivos	Objetos de Prueba	Bases de Prueba	Defectos típicos	Responsabilidades
Aceptación	Establecer confianza en la calidad del sistema como un todo Validar que el sistema está completo y funciona como se espera Verificar que los requerimientos funcionales y no funcionales del sistema cumplen con las especificaciones Satisfacer requisitos legales o regulatorios.	Sistema bajo prueba Configuración del sistema y datos de configuración Procesos empresariales para un sistema totalmente integrado Sistemas de recuperación y sitios activos Procesos operativos y de mantenimiento Formularios Reportes Datos de producción	**Generales:** Procesos de negocio Requisitos del usuario o del negocio Regulaciones, contratos legales y estándares Casos de uso Requisitos del sistema Documentación del sistema o del usuario Procedimientos de instalación Informes de análisis de riesgos **Operativas:** Procedimientos de respaldo y restauración Procedimientos de recuperación ante desastres Requisitos no funcionales Documentación de operaciones Instrucciones de despliegue e instalación Objetivos de rendimiento Paquetes de bases de datos Normas o regulaciones de seguridad	Las reglas de negocio no se implementan correctamente El sistema no cumple los requisitos contractuales o reglamentarios Fallas no funcionales	Clientes, usuarios de negocio, propietarios de productos o los operadores de un sistema y otras partes interesadas

NIVELES DE PRUEBAS: Tipos de Pruebas de Aceptación

Pruebas de Aceptación de Usuario: se centran en validar que el sistema cumple los requisitos de funcionamiento esperado en un entorno operativo real o simulado.

Pruebas de Aceptación Operativa: validan si el sistema cumple con los requisitos de operación y la realizan los usuarios y los administradores de la aplicación.

Pruebas de Aceptación Contractual: se realizan según los criterios de aceptación de un contrato para producir software desarrollado a medida.

Pruebas Alfa y Beta: se realizan para recibir retroalimentación de usuarios, clientes y/u operadores potenciales o existentes antes de la liberación del producto de software al mercado.

- **Pruebas Alpha:** se realizan en las instalaciones de la organización que desarrolla, pero no por el equipo de desarrollo.

- **Pruebas Beta:** se realizan en las propias ubicaciones del cliente, fuera de la organización que desarrolla.

TIPOS DE PRUEBAS:

Pruebas Funcionales: se encargan del "qué" debe hacer el sistema.

Pruebas No Funcionales: prueban los atributos de calidad del producto de software, qué tan bien se ejecutan elementos como: recuperación, mantenibilidad, seguridad, resistencia a fallos, volumen, estrés y carga.

Pruebas de Caja Blanca: se enfocan en la estructura interna del sistema, entiéndase el código, arquitectura o flujos de datos dentro del sistema. Suele ejecutarse en el nivel de Componente e Integración.

Pruebas de Caja negra: se enfocan en comprobar las características funcionales y no funcionales del sistema comparándolas contra las especificaciones dadas.

Pruebas Relacionadas a Cambios:

Pruebas de Confirmación: garantiza que el defecto reparado pase las pruebas que previamente falló.

Pruebas de Regresión: garantizan que no se hayan generado nuevos problemas secundarios como consecuencia de los ajustes realizados al software.

Pruebas de Mantenimiento: Suceden cuando se realizan cambios como parte del mantenimiento del software, evalúan el éxito del cambio en sí mismo y verifican posibles efectos secundarios. Es decir, constan de dos partes:

Probar los cambios y,
Pruebas de regresión.

Disparadores de Mantenimiento: Son modificaciones, migración y retiro del software.

El **Análisis de impacto** evalúa las consecuencias de la implementación de un cambio en el software.

CAP. 3 • PRUEBAS ESTÁTICAS

Las Pruebas Estáticas consisten en la revisión de documentos de requerimientos, diagramas o del código fuente para corregir y reparar defectos a bajo costo.

Las pruebas estáticas pueden ejecutarse de dos formas:

Análisis Estático:

Se usa para probar estáticamente el código fuente. Verifican los estándares de codificación, estructuras de datos y flujo de control. El compilador podría ser un ejemplo de una herramienta de este tipo.

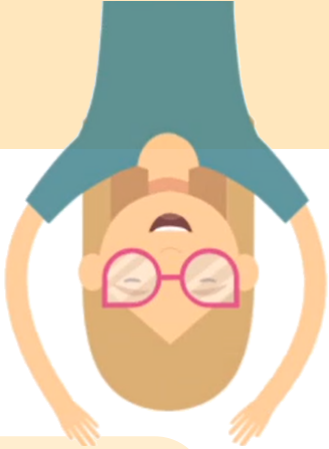
Revisiones:

Consisten en examinar cuidadosamente un producto de trabajo con el principal objetivo de encontrar y remover defectos.

DIFERENCIA ENTRE PRUEBAS
Estáticas y Dinámicas

- La prueba estática detecta defectos, la prueba dinámica identifica fallas
- La prueba estática detecta defectos difíciles de alcanzar cuando se ejecuta el software
- La prueba estática se enfoca en la calidad interna de los productos de trabajo, la dinámica en el comportamiento del software
- Los defectos encontrados en las pruebas estáticas son más fáciles de conseguir y reparar.

REVISIÓN DE LOS PRODUCTOS DE TRABAJO



Las revisiones pueden ser formales e informales

Las revisiones informales no siguen un proceso definido y no son documentadas formalmente.

Las revisiones formales tienen roles definidos, siguen un proceso establecido y deben ser documentadas.

El grado de formalidad de un proceso de revisión depende del contexto, así como de diversos aspectos técnicos y humanos, y determina qué roles son necesarios.

¡RECUERDA!

La prueba estática no implica la ejecución de código fuente, por lo tanto detecta defectos; la prueba dinámica, posterior a la ejecución del sistema, identifica fallas.

Los roles de una revisión pueden ser:

Autor del producto bajo revisión

Escriba, quien toma nota de lo discutido

Revisores que ejecutan las revisiones en si mismas

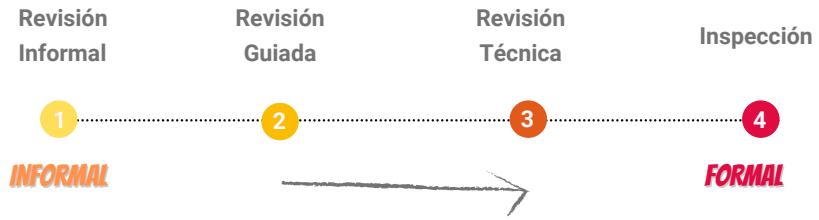
Lider, que es el responsable de la revisión

Facilitador, quien modera la reunión y

Director, quien planifica y controla.

Hay varios tipos de revisión:

De menor a mayor formalidad los **tipos de revisión** son: Informal, guiada, técnica e Inspecciones.



TIPOS DE REVISIÓN

Informal: No tiene un proceso formal documentado, no es necesaria una reunión de revisión, puede ser realizado entre colegas. El uso de listas de comprobación o Checklist es opcional.

Guiada | Walkthrough: Dirigida por el autor del producto de trabajo, el escriba es obligatorio. El uso de listas de comprobación, y la preparación individual previa es opcional.

Técnica: Se realiza entre pares técnicos del autor, si hay una reunión debe haber una preparación individual previa. Debe haber un facilitador y escriba obligatoriamente, quien idealmente no será el autor.

Inspecciones: Los resultados se documentan formalmente. Requiere preparación individual, tiene roles definidos: autor, director, facilitador, escriba, revisores y líder de revisión, puede incluir también un lector dedicado. El autor, no puede actuar como facilitador, líder de revisión, lector o escriba. Se hace uso de listas de comprobación.

ACTIVIDADES DEL PROCESO DE REVISIÓN

1. **Planificación:** definir el alcance, establecer objetivos, roles, tiempo y plazos

2. **Inicio de la revisión:** distribuir el material e instruir a los participantes

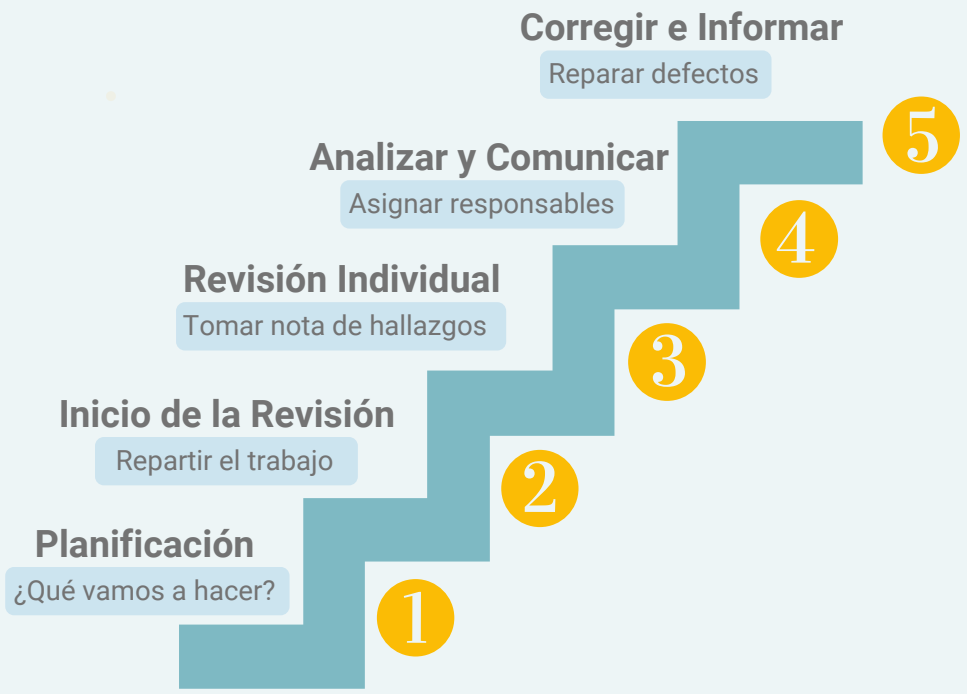
3. **Revisión Individual** del material y tomar notas de los defectos encontrados

4. **Analizar y comunicar defectos** a los responsables

5. **Corregir e informar,** elaborar informes de hallazgos, recopilar metricas, comprobar criterios de salida.

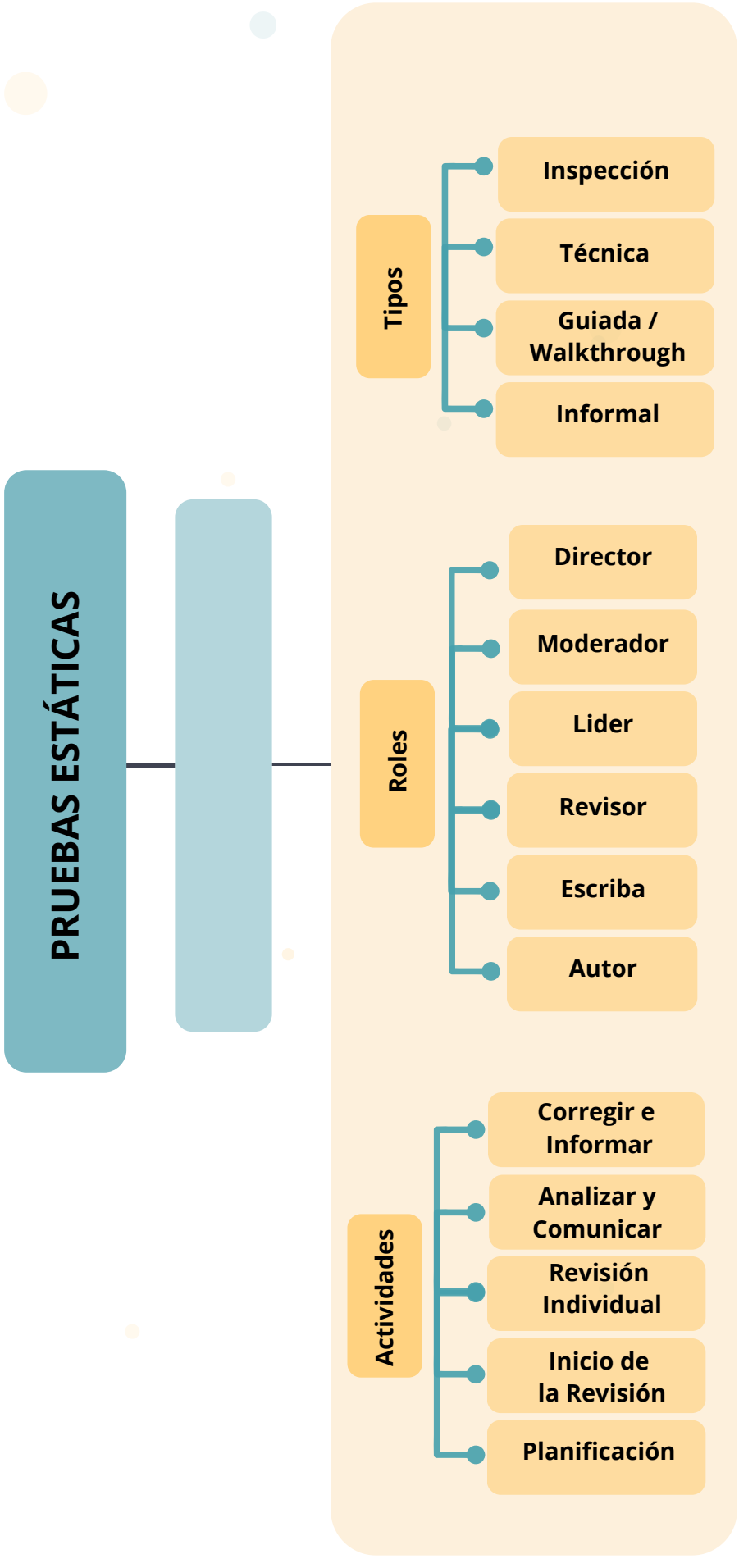
¡RECUERDA!

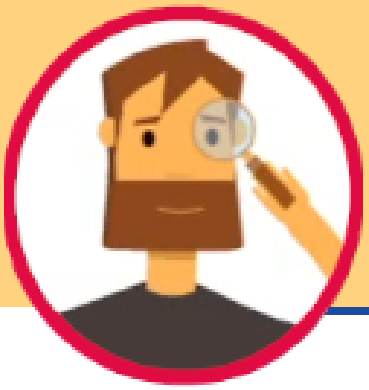
Si tu plan es presentar el examen de certificación internacional CTFL ISTQB, te recomendamos realizar los exámenes de prueba que tenemos para ti, y así asegurarte que estás listo para aprobar.



Actividades del proceso de Revisión

RESUMEN DEL CAPITULO





CAP. 4 • ANÁLISIS Y DISEÑO DE PRUEBAS

Las **pruebas estáticas** y **dinámicas** se complementan entre sí para hacer pruebas efectivas. Las revisiones y análisis estático encuentran defectos, mientras que las pruebas dinámicas encuentran fallas. Las fallas solo ocurren cuando el código es ejecutado.

ELECCIÓN DE TÉCNICAS DE PRUEBAS

Algunas técnicas formales o informales son más adecuadas para ciertas situaciones y niveles de prueba; otras se pueden aplicar en todos los niveles de prueba. Pretender que una sola técnica de prueba te proporcionará los resultados que esperas para la totalidad de un proyecto, suele ser una postura optimista, en la mayoría de los casos será necesario implementar una combinación de técnicas para lograr abarcar los casos que deseamos.

Las pruebas dinámicas constan de tres técnicas principales: Caja blanca, Caja negra y Basadas en la experiencia

TÉCNICAS DE PRUEBAS

Las técnicas de caja negra o basadas en especificaciones, son apropiadas en todos los niveles de prueba donde exista una especificación, desde la prueba de componentes hasta la prueba de aceptación.

Las técnicas de caja blanca o basadas en la estructura, también se pueden utilizar en todos los niveles de prueba. Los desarrolladores utilizan técnicas basadas en estructuras en las pruebas de componentes y las pruebas de integración de componentes, especialmente cuando hay un buen soporte de herramientas para la cobertura de código.

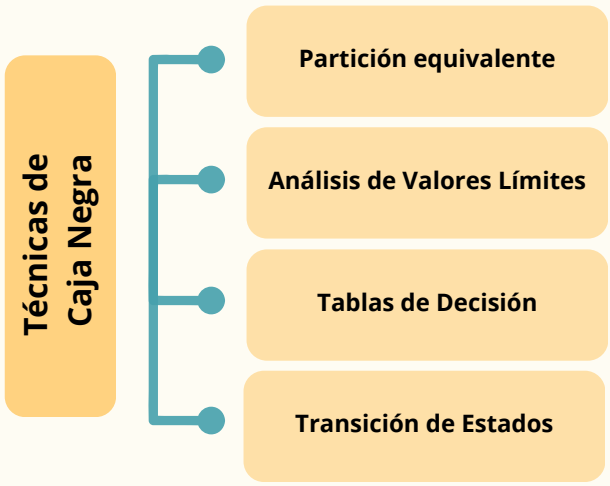
Las técnicas basadas en la experiencia se utilizan para complementar las técnicas de caja negra y blanca, y cuando la especificación es inadecuada, no existe, o no está actualizada. Este puede ser el único tipo de técnica utilizada para sistemas de bajo riesgo, y ser particularmente útil bajo una presión de tiempo extrema.

TÉCNICAS DE PRUEBAS DE CAJA NEGRA

Las técnicas de caja negra ven el software como un recuadro negro con entradas y salidas, pero no tienen conocimiento de cómo funciona internamente el sistema. El probador se concentra en lo que hace el software, no en cómo lo hace.

En las técnicas de caja negra se pueden listar: Partición Equivalente, Análisis de valores límites, Pruebas de tablas de decisión y Pruebas de transición de estados.

Partición Equivalente



También se le conoce como particiones de equivalencia o **clases de equivalencia** y se aplica en cualquier nivel de pruebas, consisten en identificar el conjunto de valores que van a producir una reacción equivalente en el software, y tomar un valor de ese conjunto como representante de la prueba.

Sirva el siguiente escenario como ejemplo:

En un sistema electoral tenemos la variable "Edad para votar", hay personas mayores de edad y menores de edad, todas las personas dentro de cada grupo sin importar los años que tengan, serán tratados de la misma forma en un proceso electoral.

Menores de Edad	Mayores de Edad
0..17	18+
Valor a probar: 15	Valor a probar: 21

(Valores tomados aleatoriamente dentro de cada partición de equivalencia)

TÉCNICAS DE PRUEBAS DE CAJA NEGRA

Análisis de valores frontera

Consiste en tomar los valores límites de los conjuntos que nombramos anteriormente debido a que es común que en casos de prueba relacionados con rangos, los defectos se concentren justamente en los límites de ambos conjuntos. Existen 2 técnicas al momento de implementar un "Análisis de valores Frontera", uno está basado en el análisis de 2 puntos y otro en el análisis de 3 puntos.

Al usar la técnica de 2 puntos la Frontera hace referencia a una línea divisoria entre 2 particiones con valores a ambos lados de esa línea, pero la Frontera o el Límite en sí mismo, no es un valor.

En la técnica de 3 puntos, el valor Frontera o Límite está "EN" la Frontera, y los otros 2 valores se encuentran a los lados de dicho valor, bien sean estos provenientes de una partición válida o inválida.

Para más ejemplos explicados paso a paso, dirígete al canal de ***Youtube/FullAdvanced.***

Suponga un campo que acepta un valor entero único positivo entre 1 y 5. El rango válido es del 1 al 5, ambos extremos incluidos. Sin embargo, sabemos que un campo de valor entero de un dígito puede recibir valores del 0 al 9.

Particiones de Equivalencia:

Inválida (demasiado baja)	Válida	Inválida (demasiado alta)
0	1, 2, 3, 4, 5	6, 7, 8, 9

Análisis de dos valores por frontera:

Inválida - Válida (demasiado baja)	Válida - Inválida (demasiado alta)
0 y 1	5, 6

Análisis de tres valores por frontera:

Inválida - Válida (demasiado baja)	Válida - Inválida (demasiado alta)
0, 1 y 2	4, 5 y 6

TÉCNICAS DE PRUEBAS DE CAJA NEGRA

Tabla de Decisión

Probar todas las combinaciones puede ser poco práctico o hasta imposible, por esta razón cuando hay múltiples condiciones y salidas se pueden usar las tablas de decisión.

Para crear una tabla de decisión puedes seguir los siguientes 3 pasos:

Paso 1: Identificar las condiciones y agregarlas a la tabla.

Paso 2: Identificar todas las posibles combinaciones de verdadero y falso.

Paso 3: Identificar las salidas correspondientes a cada combinación.

Veamos un ejemplo:

Supongamos que tenemos una solicitud de compra de un vehículo, donde debe ingresar el modelo del vehículo y el color. Si ingresa ambos, el sistema procesa ambos datos y por consiguiente la solicitud, y en caso de que uno de los valores falte, se envía un mensaje de error.

Condiciones	Combinación 1	Combinación 2	Combinación 3	Combinación 4
¿Se ingresó el modelo del vehículo?	Verdadero	Verdadero	Falso	Falso
¿Se ingresó el color del vehículo?	Verdadero	Falso	Verdadero	Falso
Acciones/Salidas				
Resultado	Se procesa la solicitud del vehículo	Mensaje de Error	Mensaje de Error	Mensaje de Error

Este método puede aplicarse a cualquier cantidad de condiciones, y garantiza que todas las posibles combinaciones han sido cubiertas, e incluso identificar redundancias o escenarios inalcanzables.

TÉCNICAS DE PRUEBAS DE CAJA NEGRA

Pruebas de Transición de Estado

Consisten en verificar el flujo de eventos de un software. Cada paso es un estado, y la transición es el salto de un estado a otro. Para hacer estas pruebas se pueden usar diagramas y tablas de transición, y se suele utilizar para probar menús o para probar la navegación en pantalla.

Para comprender un Gráfico de Transición de Estado es importante identificar el significado de los símbolos:

- El **Círculo** representa un **Estado.**

- Las **Transiciones** se representan por **Líneas** con punta para garantizar la direccionalidad.

- El Evento puede estar condicionado por una **Condición de Guarda,** está representado por Corchetes "[ ]", lo que significa que si no se cumple, no se ejecutará la transición.

Encender [Si hay electricidad]

Los **Eventos** y las **Acciones** son textos cercanos a las transiciones, se suelen separar por el símbolo de barra oblicua o slash. Primero la Acción y luego el Evento.

Veamos un ejemplo sencillo sobre un sistema de encendido de un bombillo.

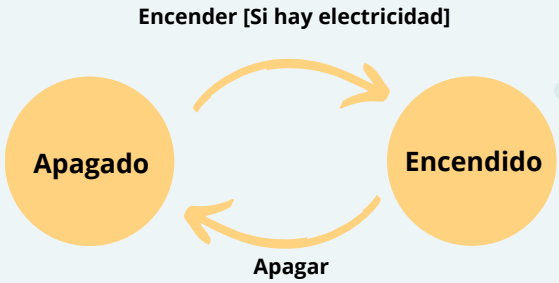


Diagrama de Transición

Para más ejemplos explicados paso a paso, dirígete al episodio 14 siguiendo este link ***Youtube/FullAdvanced.***

TÉCNICAS DE PRUEBAS DE CAJA BLANCA

Las técnicas de caja blanca

o caja de cristal son llamadas así porque se puede ver que hay dentro de la caja, conocemos la estructura del código y por esa razón también se llaman pruebas basadas en la estructura.

En este grupo se estudian dos técnicas en particular: **las pruebas de cobertura de sentencia y las pruebas de cobertura de rama o decisión.**

Pruebas de sentencias

Una sentencia es una expresión de código que puede ejecutarse como un todo, por ejemplo un cálculo matemático, una declaración if, while o case, y la idea es cubrir cada sentencia del código fuente para validar que no existen defectos en ellas.

1. Leer A
2. Leer B

3. Si A+B > 100 entonces
4. Imprimir “Suma mayor a 100”
5. Fin

6. Si A > 50 entonces
7. Imprimir “A es mayor”
8. Fin

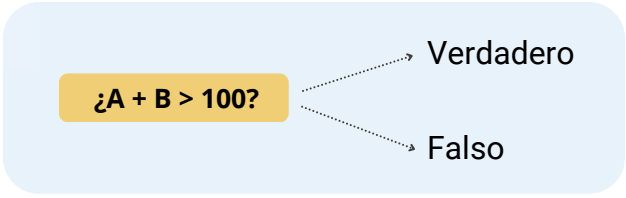
Cada línea es una sentencia. En este código hay 8 sentencias.

Pruebas de ramas o decisión

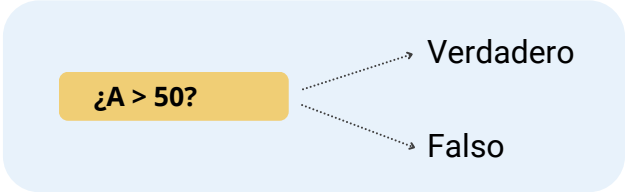
Para probar las estructuras que pueden generar más de un resultado, como los condicionales o ramas, es necesario usar la técnica de cobertura de ramas para garantizar un correcto funcionamiento del código. Esto consiste en probar cada condición validando cada posible resultado verdadero o falso.

Una **condición** se lee como una pregunta, o un conjunto de preguntas, y tiene dos posibles respuestas.

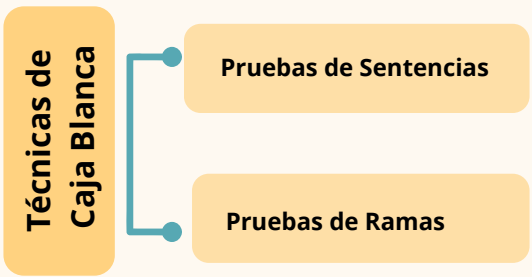
La **decisión** o rama es la respuesta a la pregunta de la condición.



Condición en la sentencia 3 del código visto anteriormente.



Condición en la sentencia 6 del código visto anteriormente.



TÉCNICAS BASADAS EN LA EXPERIENCIA

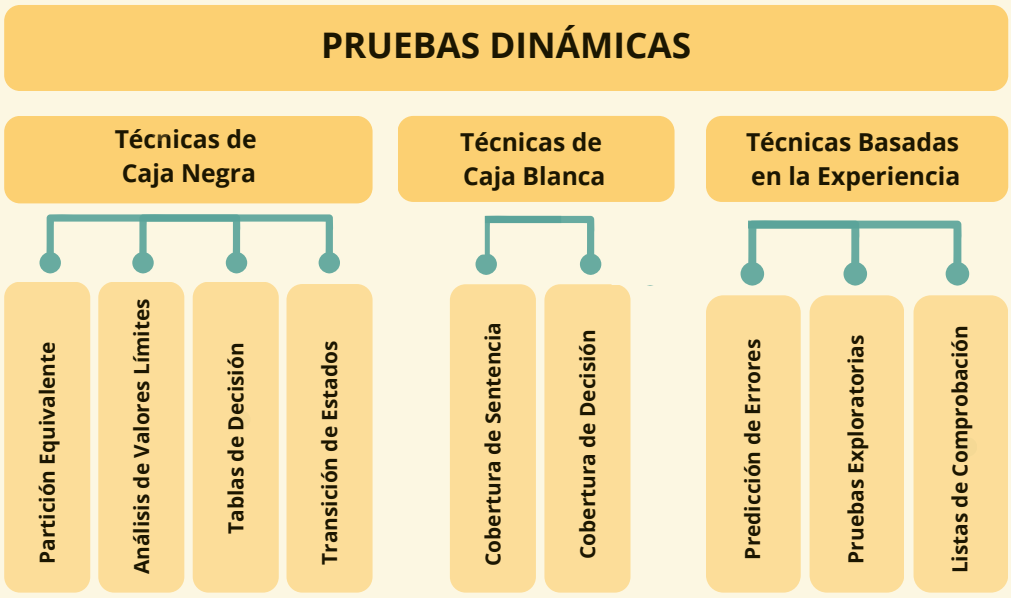
Algunas veces no tenemos tiempo disponible para ejecutar las pruebas antes descritas, o no hay especificaciones bien documentadas, y es allí donde entran en juego las técnicas de pruebas basadas en la experticia del probador: **predicción de errores, pruebas exploratorias y las basadas en listas de comprobación.**

Estas técnicas aprovechan la experiencia de los desarrolladores, probadores y usuarios, para diseñar, implementar y ejecutar las pruebas. Usualmente estas técnicas se combinan con las de caja negra y caja blanca.

La predicción de errores se basa en la información que se tiene de cómo ha funcionado la aplicación en el pasado, tipo de errores que tienden a cometer los desarrolladores y fallos que se han producido en otras aplicaciones.

Las pruebas exploratorias consisten en explorar el software para saber que hace, que no hace, qué funciona y qué no, las decisiones sobre que probar se van tomando sobre la marcha.

Las pruebas basadas en listas de comprobación consisten en listar los posibles defectos/fallas que pueda presentar el software, basándose como las anteriores, en la experiencia del probador sobre por qué y cómo falla el software.



CAP. 5 • GESTIÓN DE LAS ACTIVIDADES DE PRUEBA

Ejecutar pruebas es una actividad compleja, a menudo es un miniproyecto dentro de un proyecto de software. Por eso es importante saber cómo organizar al equipo, estimar, planificar y aplicar estrategias para distribuir eficientemente el esfuerzo de prueba, así como monitorear el progreso y crear reportes de hallazgos acordes a la audiencia que los leerá.

Full Advanced

¿QUÉ ES UN PLAN DE PRUEBAS?

Un plan de pruebas es un resumen general del proyecto, contiene una breve descripción del sistema sujeto a pruebas que da contexto acerca de lo que se va a detallar en el resto del documento.

Esta información es generalmente proporcionada por el Dueño del Producto, y puede ser extraída del documento de especificación de requerimientos del sistema.

El plan de prueba va evolucionando a medida que lo hace el proyecto. Pero, **¿Qué contiene un plan de prueba?**

La plantilla del plan de prueba IEEE 829 contiene: Un identificador, una introducción, objetos a probar, qué probar y qué no, las actividades a ejecutar, los entregables, las necesidades de ambiente, responsabilidades, el recurso humano disponible, que entrenamiento necesita el equipo, el calendario de actividades, y los criterios de entrada/salida para dar inicio a las pruebas, y para dar las pruebas por concluidas respectivamente; riesgos, contingencias y aprobaciones requeridas.

Criterios de Entrada

Los criterios de entrada son condiciones o requisitos que deben cumplirse antes de que comience una fase o etapa específica del proceso de pruebas. Estos criterios se establecen para garantizar que el entorno de pruebas sea adecuado y que el trabajo de prueba pueda llevarse a cabo de manera efectiva.

Criterios de Salida

Los criterios de salida son condiciones que determinan cuándo una fase de pruebas se considera completa y puede concluir. Estos criterios se establecen para evaluar si los objetivos de la fase de pruebas se han cumplido y si el software está en un estado aceptable para pasar a la siguiente etapa del proceso de desarrollo o pruebas.

CRITERIOS DE ENTRADA Y CRITERIOS DE SALIDA

Los criterios de entrada definen las condiciones que deben cumplirse para dar inicio a una actividad.

Los criterios de salida definen las condiciones deben cumplirse para poder cerrar una actividad, también se les puede llamar definición de hecho.

Los criterios de entrada más comunes tienen que ver con la disponibilidad de requisitos, historias de usuarios, entornos de prueba, datos de pruebas y demás herramientas necesarias. Es decir, que lo que necesitamos para arrancar las pruebas estén completos.

Un ejemplo de criterios de salida puede ser que las pruebas planificadas hayan sido ejecutadas a totalidad, o que cierto nivel de cobertura se haya alcanzado, por ejemplo se diseñaron 100 casos de prueba y se estableció que con un 80% de casos aprobados se puede dar como satisfecha la prueba.

Algunas veces las pruebas deben terminarse aunque no se hayan cumplido los criterios de salida, esto puede ser válido en casos donde se haya hecho un análisis de riesgo y se acepte el costo de poner el producto en marcha sin haber culminado lo planeado.

Calendario de Ejecución de la prueba

Una vez que los casos de prueba han sido desarrollados como corresponde, se debe organizar en el calendario el orden en el que deben ejecutarse y definir cuál es la secuencia más eficiente para ejecutar las pruebas.

Lo ideal es ejecutar los casos de pruebas en orden de prioridad tomando en cuenta las dependencias entre ellos. Si tenemos un caso de prueba de prioridad alta, que depende de otro de prioridad más baja, este último se debe ejecutar primero.

TÉCNICAS DE ESTIMACIÓN: RATIOS, EXTRAPOLACIÓN, DELPHI Y 3 PUNTOS

Las técnicas de estimación permiten determinar el esfuerzo de prueba relacionado con un proyecto, con esto se puede planificar la cantidad de recursos necesarios, tiempo requerido y presupuesto para un proyecto ágil, de forma que se puedan tomar decisiones informadas y realizar una planificación efectiva, aunque es importante tener en cuenta que las estimaciones son aproximaciones, y que algunas veces será necesario ajustarlas a medida que se avanza en el proyecto.

Técnicas de Estimación de esfuerzo de la Prueba

La técnica basada en métricas estima el esfuerzo de prueba basada en métricas de proyectos similares anteriores y de datos de la industria.

Basada en expertos: estima el esfuerzo de la prueba basándose en la experiencia de los encargados de las tareas de prueba, o por expertos. Es decir, implica consultar a las personas que harán el trabajo y a otras personas con experiencia en las tareas a realizar.

TÉCNICAS DE ESTIMACIÓN BASADA EN MÉTRICAS

Algunos de los datos que son analizados para la estimación son:

- La velocidad del equipo. Es decir, la cantidad de trabajo que pueden completar en un sprint.
- El progreso. Es decir, cómo avanzan en la compleción de tareas con respecto a la planificación.
- Dónde se registran los cuellos de botella.
- La cantidad de defectos encontrados, con el fin de identificar patrones y establecer estimaciones más realistas sobre la calidad y los desafíos que pueden surgir en un proyecto futuro.

Dentro de la estimación basada en métricas, podemos encontrar la técnica basada en ratios o proporciones, y la extrapolación.

AUTOMATIZACIÓN DE PRUEBAS

¿Qué es la automatización?

La automatización de pruebas podría describirse como el desarrollo de un sistema para probar otro sistema. En términos prácticos, consiste en el desarrollo de un programa que permite ejecutar casos de prueba y comparar automáticamente los resultados obtenidos vs. los resultados esperados.

Entonces, podríamos decir que la automatización de pruebas es la utilización de un *software A* para realizar y apoyar en las actividades de prueba que se realizan sobre otro *software B*, que fue construido para resolver una necesidad en particular.

El *software A* prueba, mientras que el *software B* es probado.

El *software B* satisface la necesidad del cliente. El *software A* garantiza que se cumpla lo solicitado.

A es el software para probar, y *B* el software bajo prueba.

Se pueden hacer pruebas automatizadas mediante: Interfaces dentro del programa, API 's, la interfaz gráfica de usuario, una consola, o servicios web.

¿Para qué sirve la automatización?

La automatización de pruebas sirve para ejecutar tareas repetitivas mejorando la integridad de las mismas y comparando automáticamente resultados obtenidos vs. esperados, con esto se disminuye al mínimo el riesgo de errores humanos y se ahorra tiempo de ejecución. Es decir, la automatización es ideal para pruebas de regresión.

La Automatización es imprescindible para ejecutar pruebas que no es posible realizar de forma manual, como por ejemplo las pruebas de estrés, o pruebas de carga. Sin embargo, hay que tomar en cuenta que no pueden usarse para todo, y que se debe invertir un tiempo considerable en el análisis, diseño, implementación, desarrollo, mantenimiento y documentación del software de prueba.

AUTOMATIZACIÓN DE PRUEBAS

Cuando hablamos de automatización es posible que se piense en software, sin embargo, es importante recordar que una solución de automatización de pruebas incluye: el software, la documentación, los casos de prueba, los datos y el entorno de prueba. Es por eso que decimos que una solución de automatización debe ser gestionada como un proyecto en paralelo al software que se está probando.

Beneficios y riesgos de la automatización

Tiempo ahorrado al reducir el trabajo manual repetitivo, por ejemplo: ejecutar pruebas de regresión.

Prevención de errores humanos, y mayor consistencia y repetibilidad en las pruebas.

Reducción de los tiempos de ejecución de pruebas, con detección más temprana de defectos, retroalimentación más rápida y un tiempo de comercialización más rápido.

Aumentar la frecuencia de las pruebas.

Expectativas poco realistas sobre la herramienta: Facilidad de uso, funcionalidad, y costos.

Subestimar el esfuerzo requerido para introducir una herramienta, mantener los scripts de prueba y cambiar el proceso de prueba manual existente.

Uso inapropiado de la herramienta cuando la prueba manual es lo recomendado.

Ignorar la necesidad de las pruebas manuales.

Incompatibilidad con la plataforma de desarrollo.

PIRAMIDE DE PRUEBAS

La pirámide de pruebas también se conoce como la pirámide de automatización de pruebas, ya que la idea básica es que todos los niveles de prueba en la pirámide se pueden automatizar.

La forma de la pirámide indica que hay más pruebas en los niveles inferiores, lo que garantiza pruebas más tempranas y una eliminación de defectos más económica. Las pruebas unitarias son la base de la pirámide, seguidas de las pruebas de integración y finalmente las pruebas de interfaz de usuario, de aceptación, o las pruebas de extremo a extremo; depende de la bibliografía que consultes.

Habrá muchas pruebas unitarias, comparativamente menos pruebas de integración, y así sucesivamente cuanto más alto estemos en los niveles de prueba. A medida que el software evoluciona, los flujos y requisitos también pueden cambiar, lo que resulta en un mayor esfuerzo de mantenimiento para las pruebas automatizadas. Esto puede hacer que algunas organizaciones prefieran pruebas manuales para la capa de aceptación, especialmente si los ciclos de lanzamiento son rápidos y las pruebas cambian frecuentemente.

Fuente: Agile Testing Foundations: An ISTQB Foundation Level Agile Tester guide by Rex Black

En el contexto de la pirámide de automatización de pruebas, las pruebas manuales pueden representarse como una nube en la parte superior.

Por lo general, las pruebas unitarias y de integración se automatizan utilizando diferentes herramientas de prueba de API, como frameworks de pruebas unitarias, o arquitecturas orientada a servicios (SOA), etc. Las pruebas del sistema y de aceptación generalmente se automatizan con algunas herramientas de prueba basadas en GUI o frameworks de prueba.

Incluso si todos los niveles de la pirámide de prueba pudieran automatizarse, aún quedarían algunas pruebas manuales. Esto podría ser exploratorio y las pruebas podrían centrarse más en la validación que en la verificación. Estas pruebas son muy beneficiosas para el proyecto, porque encuentran defectos que otras pruebas podrían pasar por alto.

GESTIÓN DE RIESGOS

El riesgo se utiliza para decidir dónde y cuándo empezar a realizar la prueba y para identificar las áreas que necesitan más atención.

¿Qué es un riesgo?

El riesgo implica la posibilidad de que ocurra un evento en el futuro que tenga consecuencias negativas.

Cuanto más probable es el resultado, peor es el riesgo. El riesgo se clasifica en: riesgos del proyecto y riesgos del producto o factores relacionados con lo que estamos probando.

La prueba basada en el riesgo se basa en el conocimiento del equipo implicado en el proyecto para llevar a cabo el análisis de riesgo de producto.

Riesgos de Producto vs. Riesgos de Proyecto

El riesgo de producto implica la posibilidad de que un producto de trabajo no satisfaga las expectativas razonables de los usuarios o partes interesadas.

El riesgo de proyecto implica situaciones que, en caso de que ocurrieran, podrían tener un efecto negativo en la capacidad de un proyecto para lograr sus objetivos. Hay riesgos de proyecto asociados al proyecto en sí mismo, a la organización, carácter político, técnico y asociados a proveedores.

Para descubrir los riesgos asociados a su proyecto, pregúntese a usted mismo y a otras partes interesadas del proyecto:

- ¿Qué podría salir mal en el proyecto para retrasar o invalidar el plan de prueba, la estrategia de prueba y la estimación de prueba?
- ¿Cuáles son los resultados inaceptables de las pruebas o durante las pruebas?
- ¿Cuáles son las probabilidades e impactos de cada uno de estos riesgos?

Diseño y contenido: Kelly Aguilar Zambrano

GESTIÓN DE RIESGOS

Probabilidad vs. Impacto

La probabilidad del riesgo es una medida cuantitativa de la posibilidad de que ocurra un evento específico. La probabilidad se expresa generalmente como un número entre 0 y 1, donde 0 significa que el evento es imposible de ocurrir y 1 significa que es seguro que ocurrirá.

El impacto es la consecuencia del de que el evento ocurra.

La prueba se utiliza para reducir la probabilidad de que ocurra un evento adverso o para reducir el impacto de dicho evento. Es decir, se utiliza como una actividad de mitigación del riesgo.

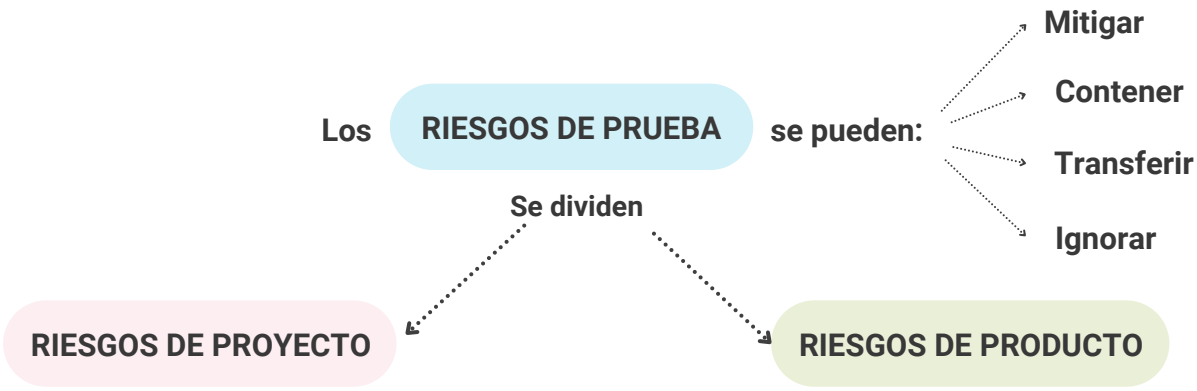
Para cualquier riesgo de producto o proyecto, hay cuatro opciones típicas:

Mitigar: tomar medidas por adelantado para reducir la probabilidad y posiblemente el impacto del riesgo.

Contener: tener un plan para reducir el impacto en caso de que el riesgo se convierta en un resultado.

Transferir: convencer a algún otro miembro del equipo o parte interesada del proyecto para reducir la probabilidad o aceptar el impacto del riesgo.

Ignorar: no hacer nada sobre el riesgo, que generalmente es una opción inteligente solo cuando hay poco que hacer o cuando la probabilidad y el impacto son bajos.



Por cuestiones asociadas a la ORGANIZACIÓN

Por problemas asociados a las PERSONAS

Por problemas de CARÁCTER TÉCNICO

Por problemas asociados con los PROVEEDORES

Insatisfacción del usuario

Daños a terceros

Pérdida de dinero, confianza y reputación

Altos costos de mantenimiento

Sanciones penales

Daños físicos, lesiones o la muerte

INFORMES DE PRUEBA

¿QUÉ ES UN INFORME DE PRUEBA?

Un **informe de prueba** resume y comunica los resultados de la actividad de prueba.

Hay dos tipos de informe: **El informe de progreso de prueba**, se emite periódicamente durante una actividad de prueba, mientras que **el informe de finalización de prueba** se prepara al final de una actividad de prueba.

El estándar ISO/IEC/IEEE 29119-3 contiene ejemplos para cada tipo.

Durante el monitoreo y control se generan informes de progreso de prueba.

Al finalizar cada hito de prueba, se generan reportes de finalización o resumen de prueba.

Los informes de progreso de prueba describen:

- La duración de la prueba.
- El estado de las actividades de prueba y el avance con respecto al plan de prueba.
- Los factores que impiden el avance.
- Pruebas previstas para el próximo período objeto de informe.
- La calidad del objeto de prueba.
- Riesgos encontrados durante el período de prueba.

Los informes de finalización de prueba describen:

- Resumen de la prueba realizada.
- Estado de la prueba y calidad del producto con respecto a los criterios de salida.
- Información sobre lo ocurrido durante un período de prueba.
- Desviaciones en el calendario, la duración o el esfuerzo de las actividades de prueba.
- Factores que han bloqueado o continúan bloqueando el avance.
- Métricas de defectos, TC, cobertura de la prueba, avance de la actividad y consumo de recursos.
- Riesgos no resueltos, defectos no corregidos.
- Lecciones aprendidas.

Diseño y contenido: Kelly Aguilar Zambrano

GESTIÓN DE DEFECTOS

Uno de los principales objetivos de las pruebas es encontrar problemas. Cada organización puede llamarlo de forma diferente: incidentes, errores, defectos, o problemas.

Un incidente es cualquier situación en la que el sistema muestra un comportamiento cuestionable. A menudo nos referimos a un incidente como un defecto solo cuando la causa raíz reside en el elemento que estamos probando.

Hablamos de incidentes para indicar la posibilidad de un comportamiento cuestionable aunque no sea necesariamente un defecto verdadero.

Es más común encontrar defectos reportados contra el código o el sistema mismo. Sin embargo, también hay casos en los que se informan defectos según los requisitos y las especificaciones de diseño, guías y pruebas para usuarios y operadores.

La forma en que se registran los defectos puede variar, dependiendo del contexto del componente o sistema que se está probando, el nivel de prueba y el modelo de ciclo de vida de desarrollo de software.

La organización debe establecer un proceso de gestión de defectos que incluya un flujo de trabajo y reglas de clasificación, y debe ser acordado con todos aquellos que participan en la gestión de defectos, incluyendo diseñadores, desarrolladores, probadores y propietarios de producto.

Al escribir un reporte de incidente es importante tener en mente a los lectores, un informe para programadores o gerentes comúnmente es diferente.

Un ejemplo del contenido de un informe de defecto puede encontrarse en la norma ISO (ISO/IEC/IEEE 29119-3).

Y qué sucede con los informes de incidentes después de que se presentan

Cada organización establece su sistema de seguimiento de defectos, pero podemos ver este ejemplo para tener una idea general:

Después de que se **reporta** un incidente, un par del probador que lo reportó o un gerente de pruebas **revisa** el informe. Si tiene éxito en la revisión, el reporte del incidente se **abre**, por lo que ahora el equipo del proyecto debe decidir si se **repara** o no el defecto.

Si se va a reparar el defecto, se asigna un programador para repararlo. También se puede tomar la decisión de **diferirlo**, es decir, no repararlo durante esa fase, o durante ese proyecto.

GESTIÓN DE DEFECTOS

Una vez que el programador ha reparado el defecto, el informe del incidente vuelve al probador para la prueba de confirmación. Si la prueba de confirmación falla, el informe del incidente se vuelve a abrir y luego se vuelve a asignar.

Una vez que el probador confirma una buena reparación, se cierra el informe del incidente. No queda más trabajo por hacer.

En cualquier estado que no sea rechazado, diferido o cerrado, el informe del incidente tiene un propietario claramente identificado. El propietario es responsable de la transición del incidente a un estado posterior permitido. Las flechas en el diagrama de estados muestran estas transiciones permitidas.

Un informe de defectos puede tener los siguientes campos:

ID	Titulo	
Fecha	Responsable	Breve Resumen
Versión	Ambiente	Fase
Alcance/ Impacto	Descripción	
Prioridad		
Estado		
Resultados Esperados	Resultados Obtenidos	
Conclusiones		
Recomendaciones	Aprobaciones	
Áreas Afectadas		
Historial		
Referencias		

Ciclo de vida de defectos

CAP. 6 • HERRAMIENTAS DE PRUEBAS

Las herramientas de prueba dan soporte a tareas que pueden resultar complicadas de realizar manualmente, o requieren muchas repeticiones que pueden incurrir en la introducción de más defectos, o consumir tiempo valioso que puede usarse en otras actividades que requieran mayor atención creativa.

Veamos algunos ejemplos de herramientas de pruebas:

- **Herramientas para la Gestión de Pruebas**
- **Herramientas para Pruebas Estáticas**
- **Herramientas para el Diseño e Implementación de Pruebas**
- **Herramientas para la escalabilidad y estandarización de la implementación**
- **Herramientas para la Ejecución y cobertura de Pruebas**
- **Herramientas para pruebas no funcionales**
- **Herramientas para DevOps**
- **Herramientas de Colaboración**

Pueden clasificarse de acuerdo al objetivo, el precio, el tipo de licencia, las tecnologías utilizadas o el soporte que brindan. Algunas herramientas pueden tener un poquito de todo, pero se van a clasificar según la actividad con la que tengan mayor relación.

PRINCIPIOS BÁSICOS PARA LA SELECCIÓN DE HERRAMIENTAS

Cuando se está considerando introducir una herramienta en la organización es importante preguntarnos si la organización tiene la madurez suficiente para afrontar el proceso.

Se identifica la necesidad y se busca una herramienta que la resuelva de forma eficiente, apoyándose en los puntos fuertes de la organización y fortaleciendo los débiles.

Es importante validar si la herramienta cuenta con un periodo de prueba gratuito que permita validar si se ajusta a los requerimientos de la organización y hacer una prueba de concepto con un proyecto, preferiblemente pequeño, que sirva como piloto de pruebas. Con esto se puede aprender a usarla, validar si la herramienta se ajusta a la organización y si cumple con los objetivos que se plantearon al inicio de la prueba.

Durante la prueba piloto se debería además, establecer cómo se ajustará la herramienta a los procesos, y protocolos de documentación que ya existen en la organización, evitando cambiar cosas de nuestra operativa actual para ajustarse a la herramienta, sino más bien cómo usar la herramienta para optimizar lo existente.

La elección de una herramienta depende de que tan bien se ajuste a la organización, y nunca al revés.

¿QUÉ ES EL EFECTO SONDA?

El efecto sonda es lo que produce el instrumento que mide sobre el objeto que está siendo medido. Supongamos que se quiere medir el tiempo de respuesta de un sistema con una herramienta, pero como dicha herramienta debe interactuar con el sistema a probar, el intercambio de mensajes entre ambos, puede afectar dicho tiempo de respuesta.

¡Hola! Hemos llegado al final del recorrido de este ebook, pero recuerda que aún queda contenido por aprender 😉. Para obtener el ebook completo ajustado el Syllabus V4, puedes acceder a ***www.fulladvanced.com/ebooks***.

Te dejamos los links al curso de Youtube, y otro contenido que puede serte de utilidad para consolidar conocimientos.

- Curso Probador Certificado Fundamentos de Pruebas
- Introducción a la Automatización de Pruebas
- ¿Cómo escribir casos de prueba?
- ¿Cómo reportar un defecto?
- ¿Qué es y cómo escribir un plan de prueba?
- ¿Cómo prepararme para una entrevista de trabajo?

Cualquier duda que te surja al estudiar puedes dejarla en los comentarios de nuestro canal de youtube, siempre respondemos 🤓.

¡Exito en tus planes profesionales, nos vemos en el futuro 😀.

Julio César y Kelly, Equipo Full Advanced.