

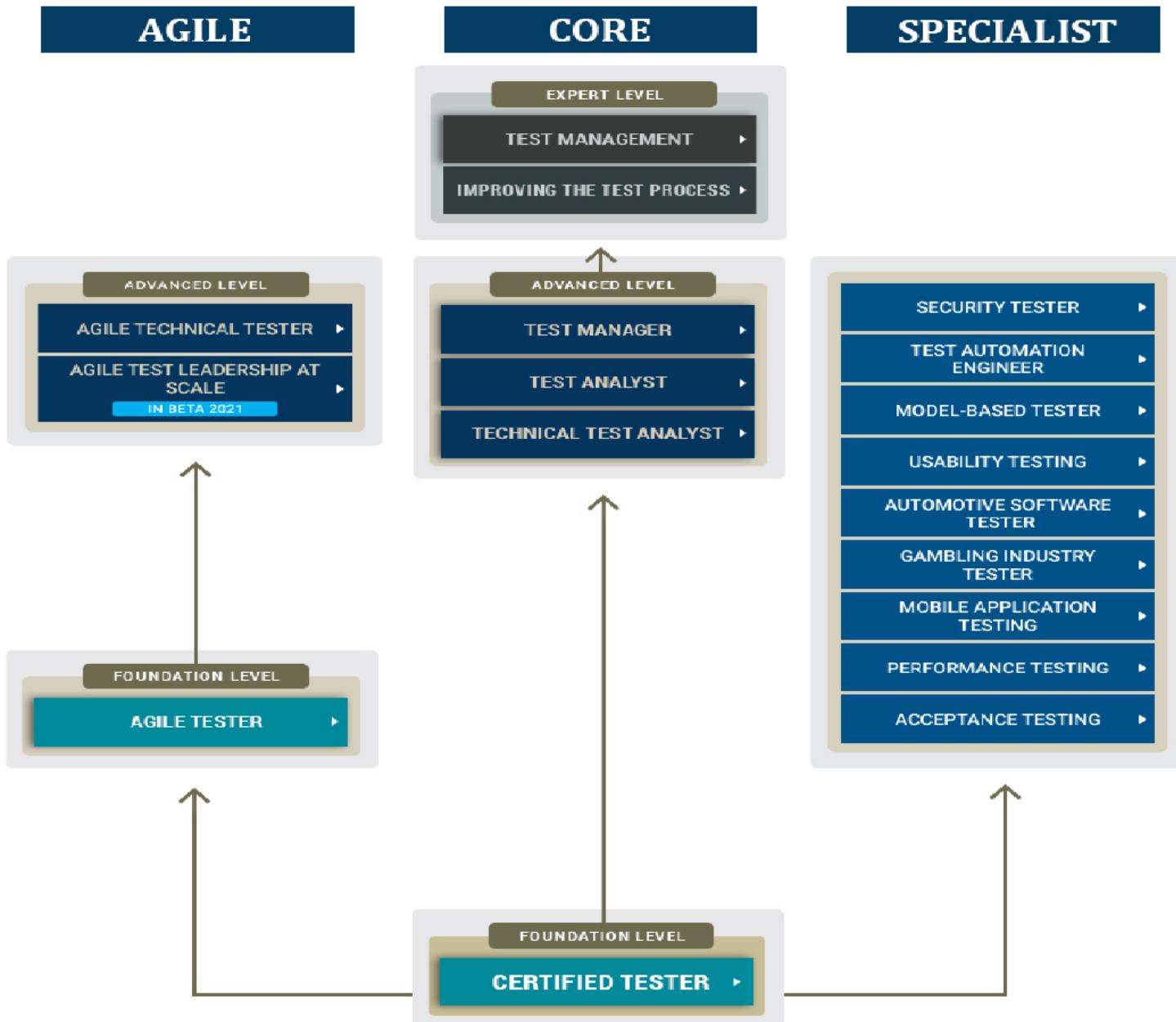


PROGRAMA DE CERTIFICACIÓN ISTQB

Foundation Introducción

INSTRUCTOR: JORGE PAZ





Programa

- 
- 0 Introducción al Programa de Certificación ISTQB
 - 1 Fundamentos de Testing (K2) 06/01/2021
 - 2 Pruebas durante todo el ciclo de vida del software (K2)
 - 3 Técnicas estáticas (K2)
 - 4 Técnicas de Prueba (K4)
 - 5 Gestión de pruebas (K3)
 - 6 Herramientas de soporte de pruebas (K2)

Examen de Certificación **ISTQB Foundation**

Preguntas de selección múltiple

Duración 60 minutos (o 75 min en inglés).

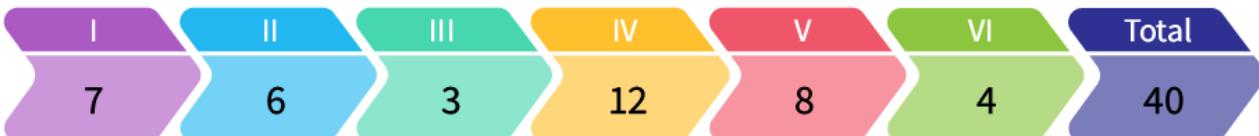
1 punto por cada respuesta correcta

Aprobación con $\geq 65\%$ (26 puntos)

Número de preguntas por K levels



Número de preguntas por capítulo



K1: Recordar

K2: Entender

K3: Aplicar

K4: Analizar

PROGRAMA DE CERTIFICACIÓN ISTQB **Foundation**

Sesión 1

Introducción

Este curso proporciona los procesos, las herramientas y habilidades esenciales que se necesitan para posicionarse en camino hacia el profesionalismo en pruebas.

International Software Testing Qualifications Board

Comité Internacional de Certificaciones de Pruebas de Software

ISTQB Foundation Level Syllabus 2011 vs 2018

2011

- Versión 2011 disponible hasta 03.12.2019.
- Más centrada en la Gestión de Pruebas.
- 27 Objetivos de Aprendizaje.
- Aborda Análisis estático con el uso de herramientas.
- Aborda temas éticos.

2018

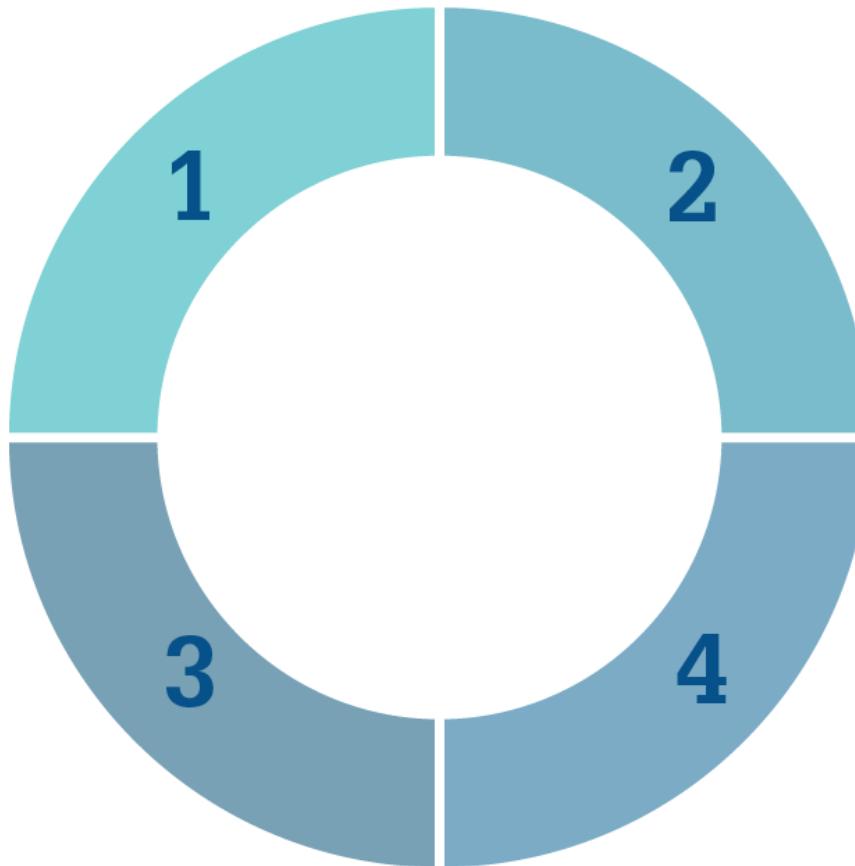
- Más centrada en las revisiones.
- 15 Objetivos de Aprendizaje
- Prioridad de pruebas WhriteBox se redujo de K4 a K3
- No Aborda temas éticos.
- Glosario con nuevas terminologías:
 - Agile methods
 - Continuous Integration/Continuous Delivery
 - Delivery pipelines
 - IoT



ISTQB® - FOUNDATION LEVEL

Fundamentals of Testing	Testing Throughout the Software Development Lifecycle	Static Testing	Test Techniques	Test Management	Tool Support for Testing
What is Testing?	Software Development Lifecycle Models	Static Testing Basics	Categories of Test Techniques	Test Organisation	Test Tool Considerations
Why is Testing Necessary?	Test Levels	Review Process	Black-box Test Techniques	Test Planning and Estimation	Effective Use of Tools
Seven Testing Principles	Test Types		White-box Test Techniques	Test Monitoring and Control	
Test Process	Maintenance Testing		Experience-based Test Techniques	Configuration Management	
The Psychology of Testing				Risk and Testing	
				Defect Management	

Objetivos



- Introducción y Objetivos

- I. Fundamentos de Pruebas
- II. Pruebas a través del Ciclo de Vida de Software
- III. Técnicas Estáticas
- IV. Técnicas de Prueba
- V. Gestión de Pruebas
- VI. Soporte de Herramientas para Pruebas

CAPITULO I

Fundamentos de Pruebas



Trabajo

Analizando data,
haciendo presentaciones,
transaccionando negocios,
Comunicando.



Casa

Comprando online,
pagando servicios,
Mirando televisión,
Jugando video juegos.



Cuando salimos

Celulares,
GPS,
tablets,
relojes.



¿Qué son las pruebas?

“Son una forma de evaluar la calidad de software y reducir el riesgo de falla en su funcionamiento”.

¿Porqué es necesario realizar pruebas?

¿Cómo pueden causar daño los defectos?

- Reputación dañada sobre la calidad.
- Costos de mantenimiento altos o impredecibles
- Retrasos inesperados en ciclos de versiones
- Falta de confianza en el sistema
- Demandas



A la Empresa

- Contaminación
- Desperdicio de recursos



Al entorno

- Pérdida de trabajos
- Pérdida de vidas
- Pérdida de derechos
- Pérdida de misiones
- Pérdida de guerras



A las personas, sociedades
y estados

¿De dónde vienen los defectos y qué ocasionan?

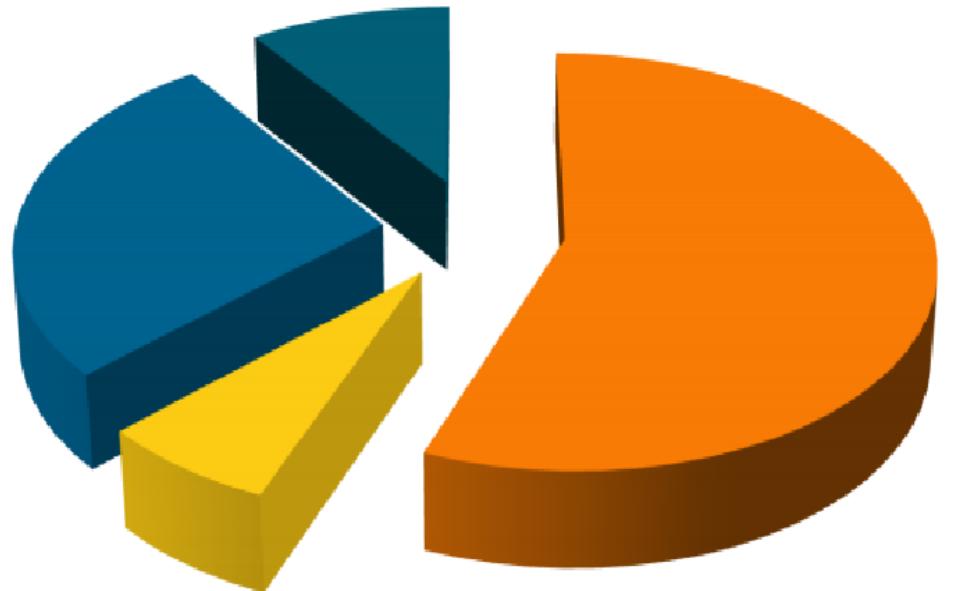
- La gente comete errores los cuales crean defectos en el Sistema.
 - Requisitos y especificaciones de diseño
 - Código (Lógica de negocios e interfaz de usuario)
 - Documentación (Electrónica e Impresa)
- Cuando un código defectuoso es ejecutado ocurren fallas.
- Si estas fallas son visibles a los clientes, usuarios u otros interesados, podrían resultar en insatisfacción con la calidad del sistema.

¿De dónde vienen los defectos y las fallas?

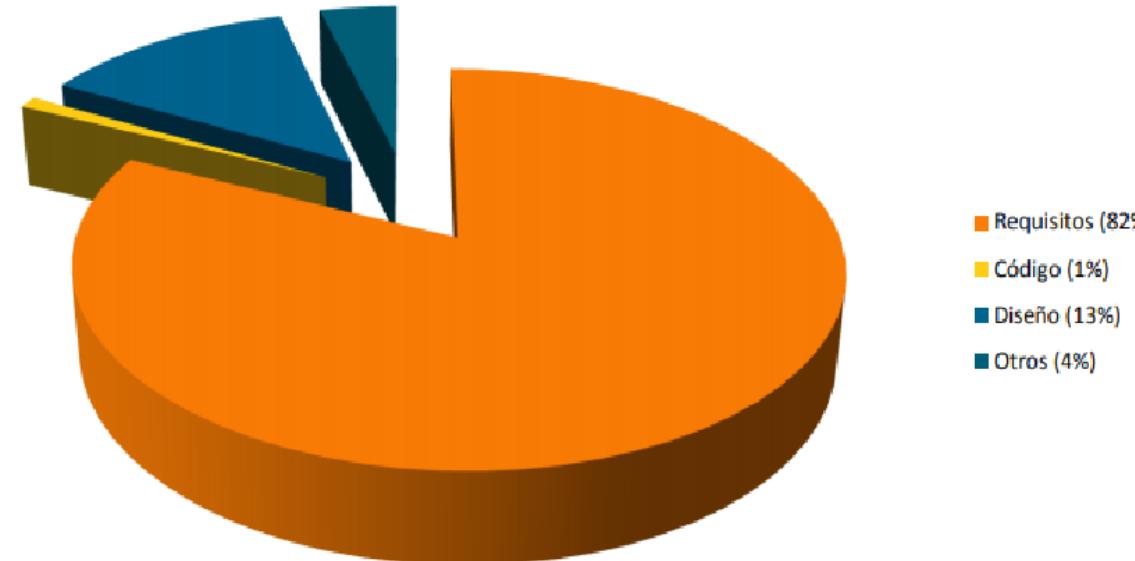
- Los defectos ocurren debido a...
 - Falibilidad de programadores, analistas y otros colaboradores individuales (incluyendo a los probadores)
 - Presión de tiempo
 - Complejidad del código, infraestructura o problemas a ser resueltos
 - Las tecnologías cambian continuamente y deben engranar
 - Muchas interacciones del Sistema
- Las fallas ocurren debido a defectos y...
 - Condiciones del entorno
 - Uso incorrecto (deliberado o accidental)



El coste de los defecto



Distribución típica del origen de los errores



Distribución típica del esfuerzo para resolver errores

¿En qué se diferencian?



ERROR

del programador



DEFECTO

en el software



FALLO

depende del sistema

Cuál es el rol de las pruebas

- Debido a los defectos y fallas un sistema está sujeto a varios riesgos.
- Existen otros riesgos y restricciones que las pruebas no pueden gestionar mucho.
 - R. Implementación de todas las características: El Conjunto correcto.
 - R. Retraso de cronogramas: Suficientemente rápido.
 - R. del presupuesto: Aceptablemente económico
 - R. de Calidad: Listo para los clientes / versión / próximo paso
- Las pruebas gestionan los riesgos de calidad de varias maneras:
 - Proporcionando la información para guiar el proyecto, reducir y gestionar los riesgos, y separar los problemas importantes.
 - Las pruebas pueden también abordar las necesidades de conformidad, contractuales y reguladoras.

¿Que significa Calidad?

“Aptitud para el uso” vs Conformidad con los requisitos”

Relación de las pruebas y la Calidad:

- Las P. dan confianza donde ellas encuentran pocos defectos.
- Las P. Exitosas reducen el nivel de riesgo de calidad.
- Las P. Fallidas proporcionan una oportunidad para mejorar la calidad.
- Un conjunto de pruebas con cobertura adecuada proporcionan una evaluación de Calidad.

Pruebas, Aseguramiento de la Calidad y Mejoramiento de la Calidad

- Los grupos de pruebas son frecuentemente referidos erróneamente como grupos de “Aseguramiento de Calidad” o grupos de “QA”
- Idealmente, las pruebas son parte de una estrategia de Aseguramiento de la Calidad para un proyecto y en toda la organización.

Objetivos de las pruebas

Incluyen

La evaluación de productos de trabajo (requisitos, historias de usuario, diseño y código)

La verificación de cumplimiento de todos los requisitos especificados

La validación de completitud y funcionamiento del objeto de prueba

Esperan

Crear confianza en el nivel de calidad del objeto de prueba.

Prevenir defectos

Encontrar fallas y defectos

Proporcionar información suficiente a los interesados para que puedan tomar decisiones informadas.

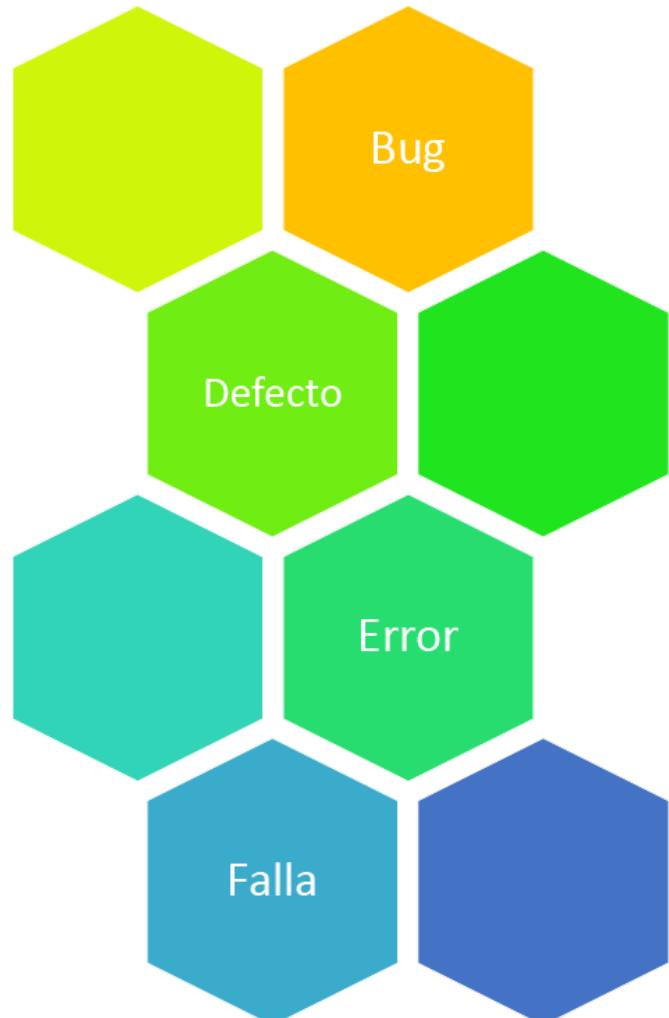
Respecto al Nivel de Calidad

Buscan reducir el nivel de riesgo de una calidad de software inadecuada (por ejemplo, fallas)

Fases y Objetivos de las Pruebas

- **Pruebas de Unidad / de Componente:** Encuentran defectos en partes individuales del sistema sometido a pruebas antes de que las partes sean integradas en el sistema.
- **Pruebas de Integración / Cadena:** Encuentran defectos en las relaciones e interfaces entre pares y grupos de componentes en el sistema sometido a pruebas a medida que las partes van juntándose.
- **Pruebas de Sistema:** Encuentran defectos en los comportamientos, funciones y respuestas totales y parciales del sistema sometido a pruebas como un todo.
- **Pruebas de Aceptación / Piloto:** Demuestran que el producto está listo para su despliegue/versión o para evaluar la calidad y dar información sobre el riesgo de despliegue/versión.
- **Pruebas de Mantenimiento:** Comprueban los defectos introducidos durante el desarrollo de los cambios.
- **Pruebas Operacionales:** Evalúan las características del sistema no funcionales tales como la fiabilidad o disponibilidad usualmente en el entorno operativo.

Términos:



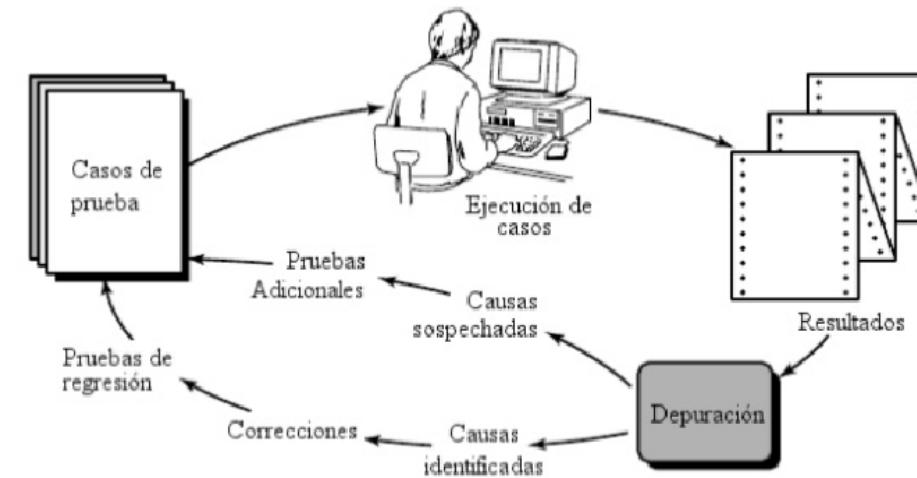
Pruebas y depuración

Las pruebas...

Encuentran fallas que son causadas por los Defectos.

La depuración...

- Analiza o Identifica la causa Raíz de un defecto.
- Repara el código
- Comprueba que se haya corregido



Las confirmación...

- Asegura que la corrección resuelve las fallas observadas.

Encontrar – Depurar - Confirmar

- 1) Identificar el problema
- 2) Diagnosticar el error
- 3) Corregir el Error
- 4) Probar la corrección del Error
- 5) Reiniciar el programa
(Por Errores previos o Generados)



Encontrar – Depurar - Confirmar

Marco Tradicional

- *Probadores son responsables de la prueba Inicial y Prueba de Confirmación.*
- *Desarrolladores realizan la depuración y las pruebas de componentes asociados.*

Marco Ágil

- *Los evaluadores pueden participar en la depuración y las pruebas de componentes.*

Probar es más que ejecutar Pruebas...

- El proceso de pruebas no es solo la ejecución de pruebas contra un sistema ejecutándose.
- Otras actividades de pruebas, antes y después de la ejecución incluyen:
 - Planificar y Controlar
 - Escoger condiciones de pruebas
 - Diseñar casos de pruebas
 - Comprobar resultados de pruebas
 - Evaluar criterios de salida
 - Informar resultados de pruebas
 - Tareas de cierre / Fin de prueba



Por qué es necesario ejecutar Pruebas...

Requisitos o Refinamiento	Diseño del Sistema	Codificación	Verificación antes del lanzamiento
<ul style="list-style-type: none"> • <u>Reduce riesgo</u> de desarrollar una funcionalidad incorrecta o no comprobable 	<ul style="list-style-type: none"> • Aumenta la comprensión para reducir el riesgo de defectos de diseño y permite identificar pruebas en una <u>etapa temprana</u>. 	<ul style="list-style-type: none"> • Aumenta comprensión de cada parte del código y cómo probarlo, reduciendo riesgo de defectos dentro del código. 	<ul style="list-style-type: none"> • Permite <u>respaldar</u> el proceso de eliminación de defectos que causaron fallas, aumentando la satisfacción de requisitos.

- El uso de técnicas de prueba apropiadas puede reducir la frecuencia de fallas o incumplimiento con las necesidades de los interesados.
- Las pruebas deben realizarse según la experiencia y los niveles de prueba apropiados y durante el ciclo de vida de desarrollo de software.

7 Principios Generales de las Pruebas

Conceptos Clave

- Las pruebas revelan la presencia de defectos.
- La imposibilidad de pruebas exhaustivas
- Los beneficios de las pruebas tempranas
- La agrupación de defectos
- La paradoja del pesticida
- Las pruebas deberían adaptarse a necesidades específicas
- La falacia de la ausencia de errores



Pruebas revelan la Presencia de Defectos: Una Metáfora

- Tiene una huerta hermosa, pero un día ve hojas comidas sobre los tomates.
- “Oh no”, piensa usted: “¡Gusanos Picudos!”
 - Sabe que tiene insectos en su jardín.
 - ¿Sino hubiera visto los síntomas, podría estar seguro de que no tuviera insectos?
 - ¿Tras ver los síntomas, podría estar seguro de que tiene Gusanos Picudos?
- Las pruebas son como caminar por su huerta: puede revelar la presencia de insectos, pero no puede probar la ausencia de ellos.



“Las pruebas pueden revelar la presencia de defectos, pero no pueden probar su ausencia”

Misión Imposible:

Las Pruebas Exhaustivas

- Se piensa que la misión del equipo de pruebas es "Solo asegurar que el software funcione antes de enviarlo", ese privilegio es imposible por las siguientes razones:

- Los caminos de ejecución en un software no-trivial son casi infinitos.
- Diferente complejidad y flujo de datos.
- Los cambios pueden causar regresiones significativas no obviamente relacionadas al cambio.
- Variedad de perfiles de uso y configuración, algunos desconocidos o imposibles de conocer.

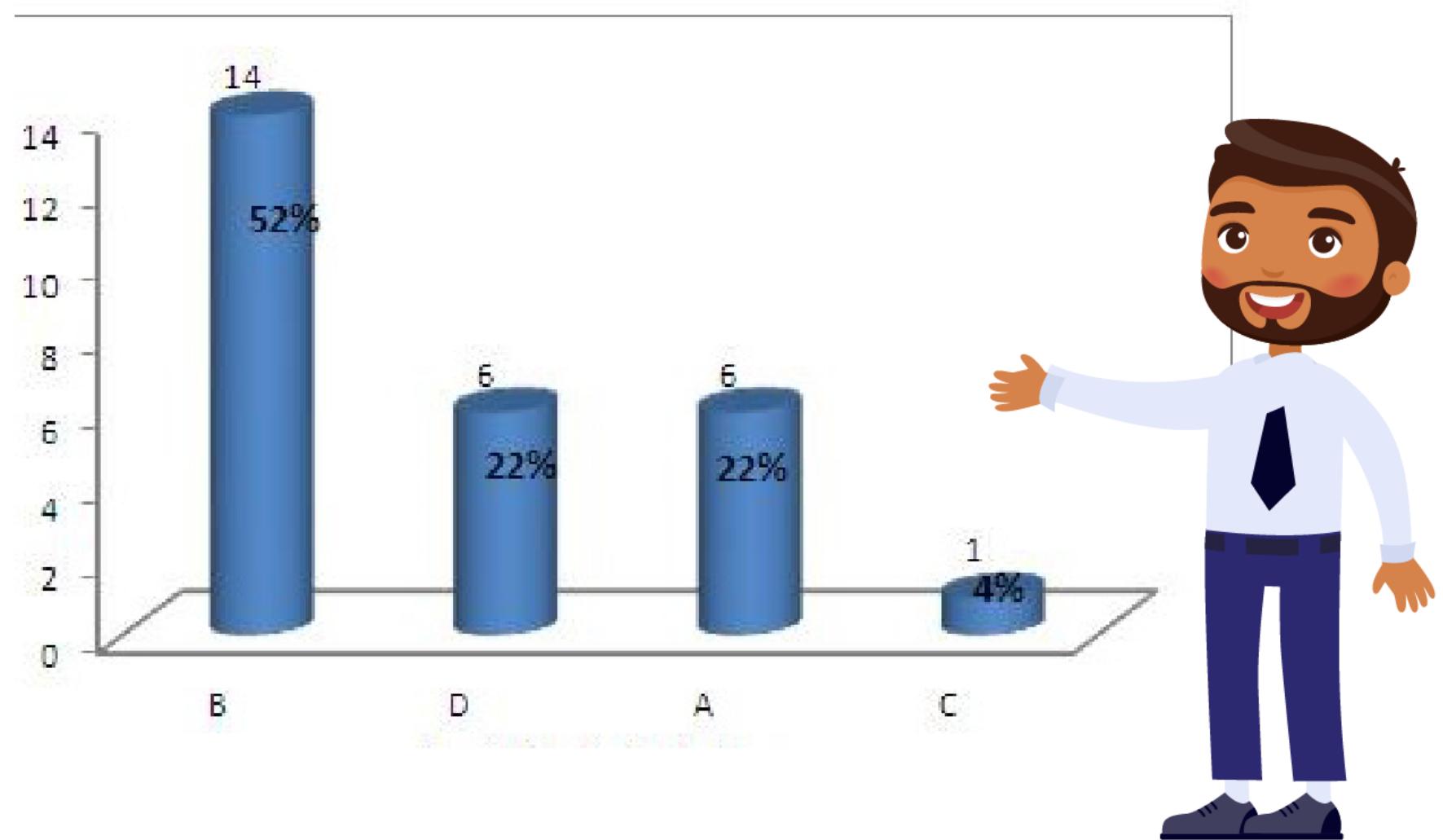
Las pruebas exhaustivas son aquellas que prueban todas las combinaciones de entradas y precondiciones.

- Las malas expectativas crean problemas para los equipos de prueba:
 - Demandas altas inalcanzables
 - Percepción de incompetencia cuando no se atienden las demandas.

Beneficios del Aseguramiento de la Calidad y las Pruebas tempranas

- El costo de un defecto tiene a aumentar a medida que le proyecto continua.
- La mayor parte de los costos asociados con defectos de una pre-versión tienen a ser asociados con el esfuerzo necesario para eliminarlos, por lo que costos más altos significan calendarios mas largos.
- Mientras más defectos entran en una actividad de aseguramiento de calidad o de pruebas, más defectos se escaparán de esa actividad.

Agrupamiento de Defectos



La paradoja del pesticida

- Regrese a su huerta.
- Ud. rocía un pesticida en su huerta, y los gusanos picudos mueren, pero el pesticida no es eficaz contra todos los insectos.
- Así como los pesticidas se vuelven menos eficaces, también lo hacen las pruebas.
- Las pruebas funcionales no pueden encontrar defectos de rendimiento.
- Intente nuevas técnicas de pruebas, si el objetivo es encontrar defectos.



Las pruebas deberían adaptarse a las necesidades

- Proyectos, organizaciones y productos diferentes tienen necesidades de pruebas diferentes.
- Las mejores prácticas existen, pero usted necesita adaptarlas a su proyecto.
- El fracaso de adaptar el equipo de pruebas y sus métodos a estas necesidades es un resultado común de disolución de equipos de pruebas.

La falacia de la Ausencia de Errores

- Encontrar y corregir muchos errores no garantiza la satisfacción del usuario, del cliente y/o de los interesados del negocio.
- Muchos productos con pocos defectos han fracasado en el mercado.
- Los proyectos exitosos equilibran las fuerzas que compiten en términos de características, calendario, presupuesto y calidad.

Proceso de pruebas básico

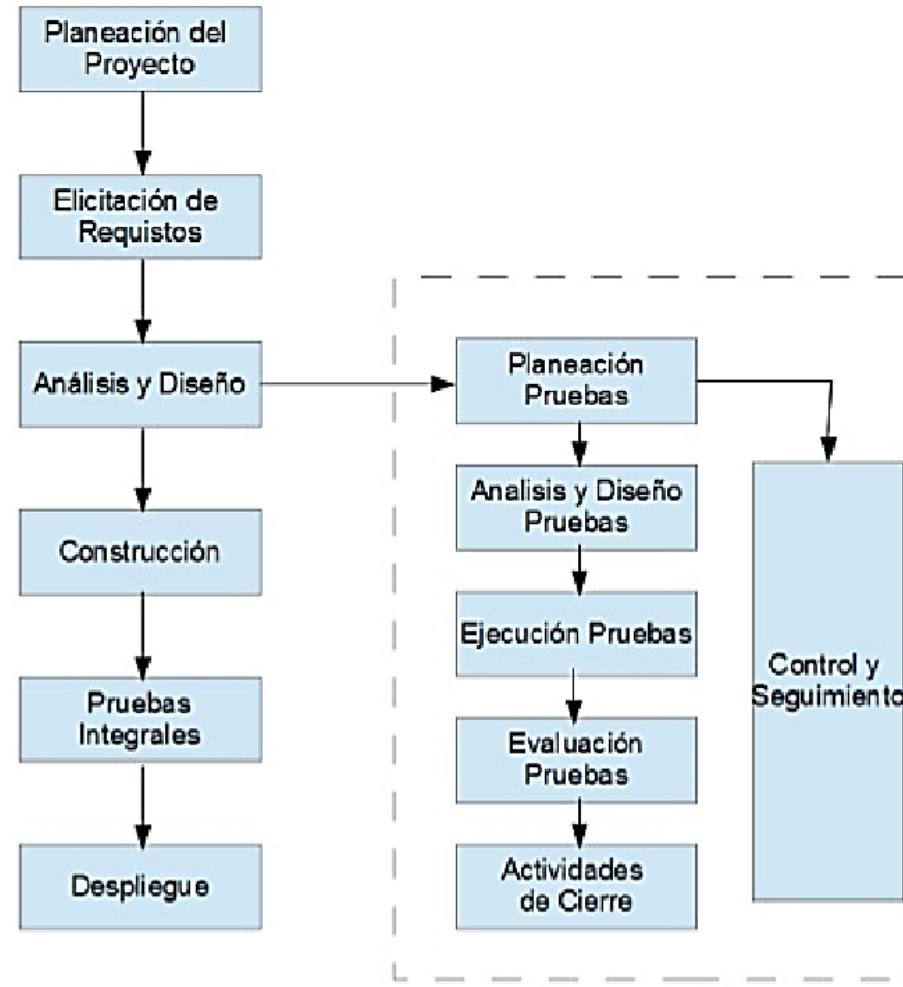
Proceso de pruebas básico



El rol de ingeniero de pruebas se centra en algunas de estas actividades, no en todas, dependiendo de como esten definidos los roles. Sin embargo, el ingeniero de pruebas eficaz debe comprender como funciona el proceso de pruebas y como se integra todo el proyecto desde una amplia perspectiva.

Proceso de pruebas básico del ISTQB

- Planificación y control
- Análisis y diseño
- Implementación y ejecución
- Evaluación de criterios de salida de pruebas y generación de informes
- Actividades de cierre de pruebas
- Nota: Estos pasos pueden superponerse, ser concurrentes y/o iterarse



Planificación y Control

La Planificación

- Determinar el alcance de las pruebas, los riesgos, los objetivos, las estrategias.
- Determinar los recursos de pruebas necesarios.
- Implementar las estrategias de pruebas.
- Calendarizar la implementación, ejecución y evaluación de las pruebas.
- Determinar los criterios de salida de pruebas.

El control

- Medir y analizar los resultados.
- Monitorear y documentar el progreso, la cobertura y los criterios de salida de pruebas.
- Indicar acciones correctivas.
- Tomar decisiones.



Muchas actividades de planificación y control implican la obtención del acuerdo, apoyo y consenso del equipo del proyecto y de la gerencia del proyecto.

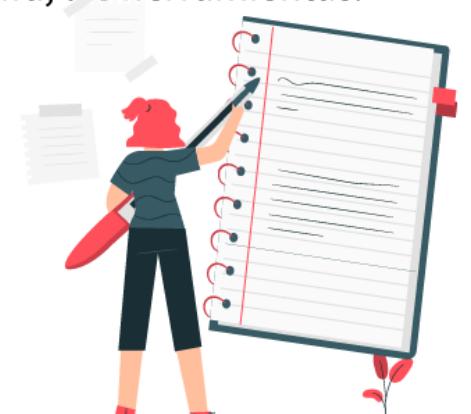
Análisis y diseño

Análisis

- Revisar las bases de las pruebas (p.ej. los requisitos o las especificaciones de diseño, la arquitectura de la red / del sistema, los riesgos de calidad).
- Identificar y priorizar las condiciones de las pruebas, los requisitos de las pruebas o los objetivos de pruebas y los datos de pruebas necesarios basados en el análisis de los ítems de pruebas (p.ej. su comportamiento, especificación y estructura).
- Evaluar la comprobabilidad de los requisitos y del sistema.

Diseño

- Diseñar y priorizar las combinaciones de los datos de pruebas, de las acciones y de los resultados esperados.
- Identificar los datos de pruebas necesarios para las condiciones y casos de pruebas.
- Diseñar el entorno de pruebas.
- Identificar la infraestructura, las herramientas.



Implementación y Ejecución

Implementación

- Desarrollar, implementar y priorizar los casos de prueba, crear datos, escribir procedimientos.
- Crear armaduras de pruebas, guiones.
- Organizar juegos de pruebas y secuencia de procedimientos.
- Verificar el entorno de pruebas.

Ejecución

- Ejecutar los casos de prueba
- Registrar resultados de prueba, versiones, herramientas y testware.
- Comparar resultados reales y esperados.
- Reportar y analizar las incidencias.
- Repetir las pruebas corregidas y/o Actualizadas.
- Ejecutar pruebas de confirmación y/o regresión

Criterios de Salida, Generación de informes y Cierre

Salida e Informes

Revisar los registros de pruebas contra los criterios de salida del Plan de Pruebas.

Evaluar si se necesitan más pruebas o si el criterio de salida especificado debería ser modificado.

Escribir un informe del resumen de pruebas para los interesados del negocio.

Cierre

Confirmar los entregables de las pruebas, la resolución final o la postergación de los informes de defectos, y la aceptación del sistema.

Finalizar y archivar el testware, el entorno de pruebas y la infraestructura de pruebas.

Entregar el testware a la organización de mantenimiento.

Realizar una retrospectiva para capturar mejoras para futuras versiones de proyectos y procesos de pruebas.

Ejercicio: Pasos y tareas de pruebas realizadas.

- Haga memoria de un proyecto reciente.
- Note cuales de los pasos y tareas del proceso prueba del ISTQB fueron llevador a cabo.
- Note cuales de los pasos y tareas del proceso de pruebas ISTQB no fueron llevados a cabo.
- Note si algunos pasos se superpusieron o fueron completamente paralelos.
- Discuta.



La psicología de las pruebas

Atributos del Buen Probador

- **Curiosidad**
- **Pesimismo Profesional**
- **Ojo crítico**
- **Atención al detalle**
- **Buenas habilidades de comunicación**



Habilidades del Tester

- Lectura
 - Especificaciones, correos electrónicos, casos de prueba, etc.
- Escritura
 - Casos de pruebas, informes de defectos, documentación de pruebas, etc.
- Habilidades en tecnologías, proyectos y pruebas
 - Tecnología: Lenguajes de programación, sistemas operativos, redes, etc.
 - Dominio de aplicación: banca, factores humanos, aplicaciones de oficina, etc.
 - Pruebas: Construcción de Scripts, exploración, automatización, etc.

Los diferentes modos de pensar

- La separación de funciones (a través de pruebas independientes) ayuda a enfocar las pruebas.
- Los probadores profesionales son más eficaces en las actividades de pruebas, encontrando fallas al ser más objetivos y no tener parcialidad del autor.

Verificación/Revisión

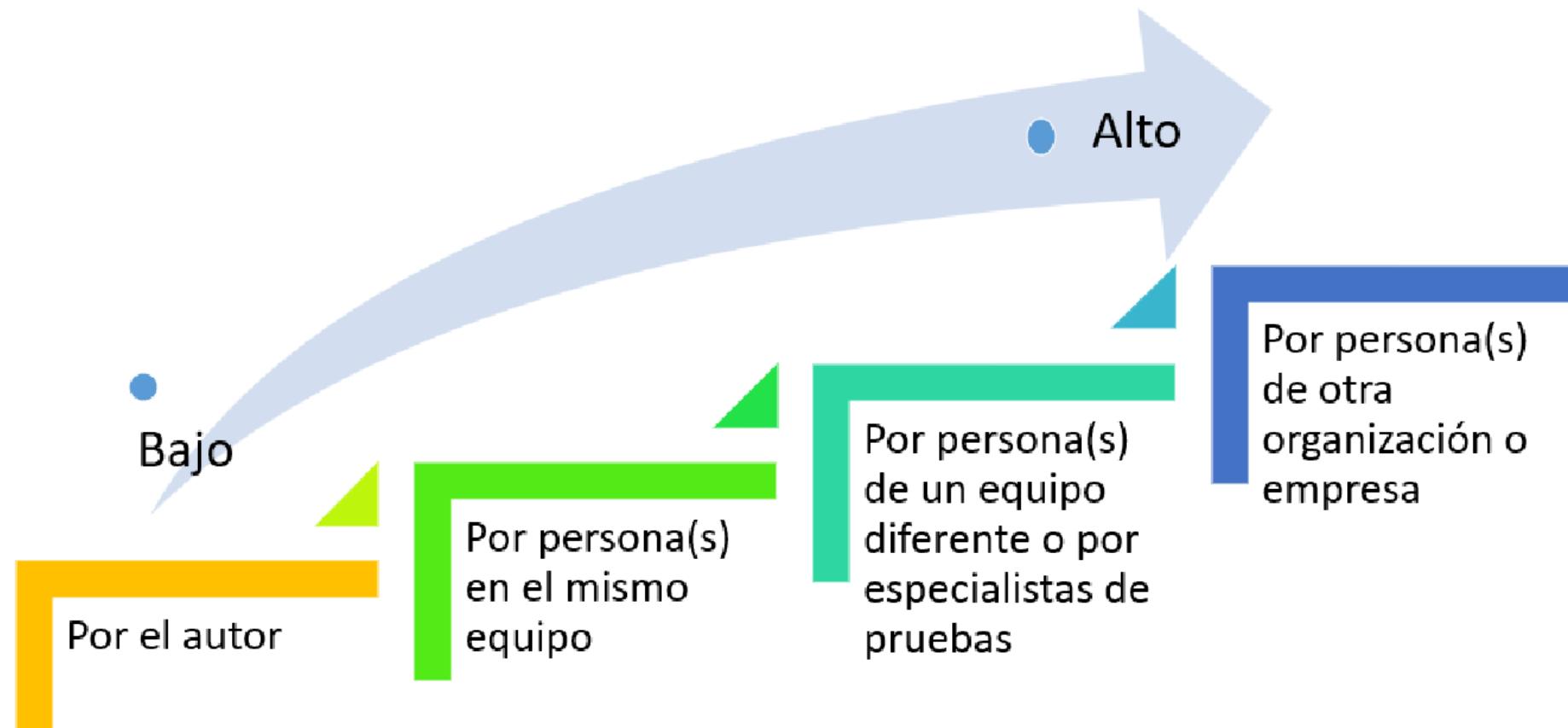
- Proceso de mejora continua.
- Busca asegurar que se satisface los requisitos durante el ciclo de vida de software
- Presta atención a la trazabilidad entre componentes y especificaciones.

Validación

- Comprobación al final del ciclo de vida del desarrollo.
- Satisfacción de requisitos y expectativas.
- Puede realizarse en ambientes de operación.
- Participa un especialista y/o el cliente.

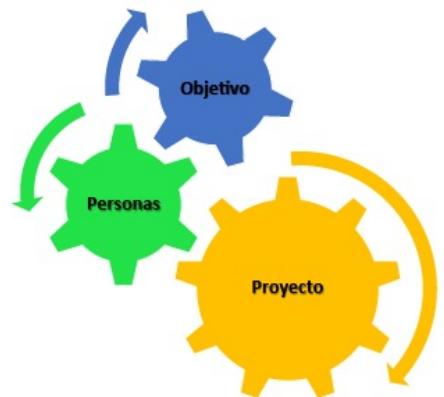
Grados de independencia

Los probadores independientes suelen ser más eficaces en encontrar fallas.



Objetivo claros

- ¿Las pruebas deberían encontrar defectos o confirmar que el software funciona?
¿Qué %? ¿En qué Nivel?
- Una política de pruebas puede ayudar a establecer claramente los objetivos.



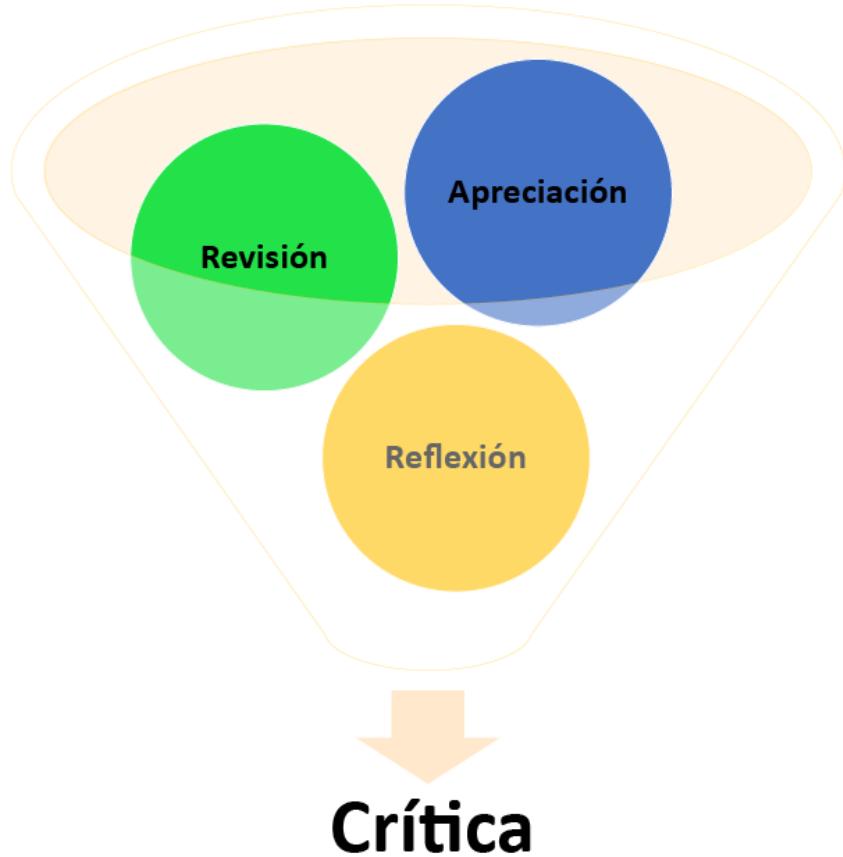
Enfoque

- Se debe mantener enfoque en las metas y objetivos del proyecto de pruebas.
- Las mentes estrechas pierden de vista prioridades más importantes.
- Se debe evitar distracción de las tareas clave.



¿Constructor o Destructor?

- A veces las pruebas son vistas como una actividad destructiva.
- Las pruebas son esenciales para la gestión de riesgos.
- Los probadores algunas veces reciben el final de las emociones.
- Se debe reportar adecuadamente los defectos o problemas.
 - Colaborar
 - Comunicar neutralmente, sin juzgar
 - Comprender a los colegas y su reacción a los hallazgos
 - Confirmar que los entiende.
 - Confirmar que te entiendan.



Psicología del tester en Acción

- Establece los hechos.
- Describe sin juzgar.
- Pregunta.
- Hablar cuidadosamente.
- Propone que se considere la situación.



- Introducción y Objetivos
- I. Fundamentos de Pruebas
- II. **Pruebas a través del Ciclo de Vida de Software**
- III. Técnicas Estáticas
- IV. Técnicas de Prueba
- V. Gestión de Pruebas
- VI. Soporte de Herramientas para Pruebas

CAPITULO II

Pruebas a través del Ciclo de Vida de Software

Modelos de desarrollo de software

Describen los tipos de actividad que se realizan en cada etapa de un proyecto de desarrollo de software, y cómo las actividades se relacionan entre sí de forma lógica y cronológica.

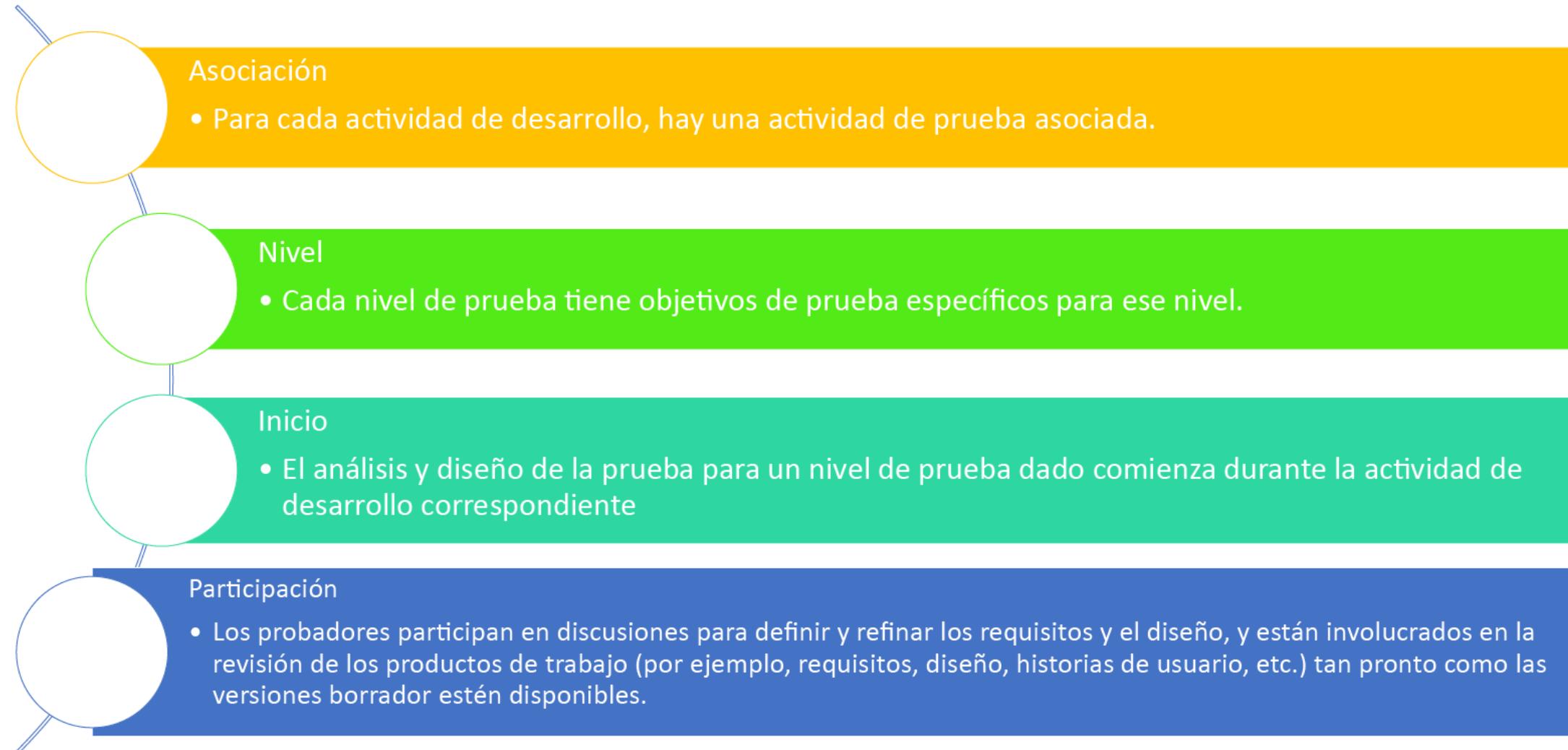
Hay diferentes modelos de ciclo de vida de desarrollo de software, cada uno de los cuales requiere diferentes enfoques de prueba.

ISTQB clasifica los modelos de ciclo de vida de desarrollo de software comunes de la siguiente manera:

- Modelos de desarrollo secuencial.
- Modelos de desarrollo iterativos e incrementales.

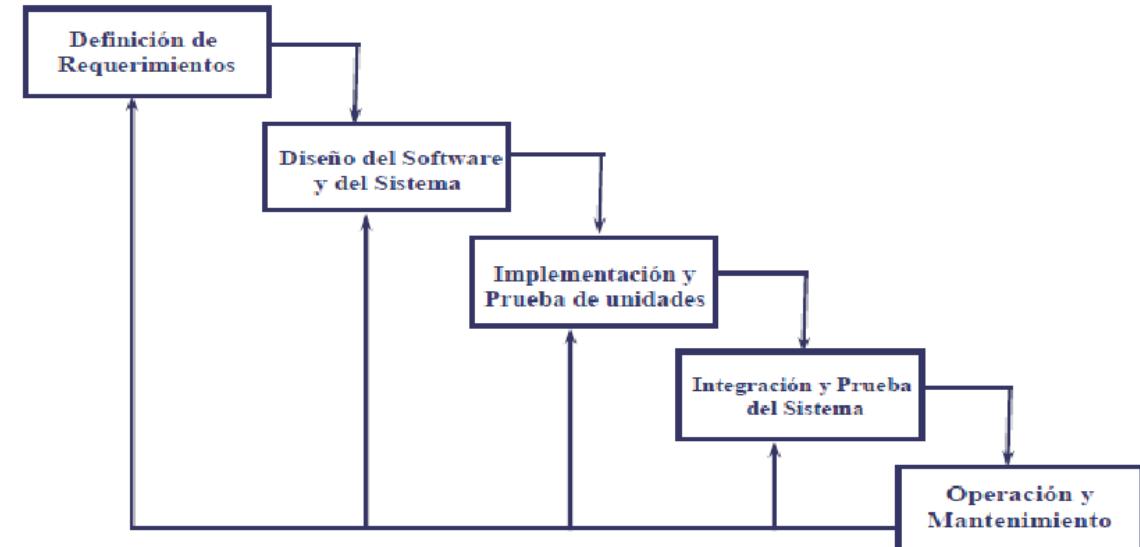
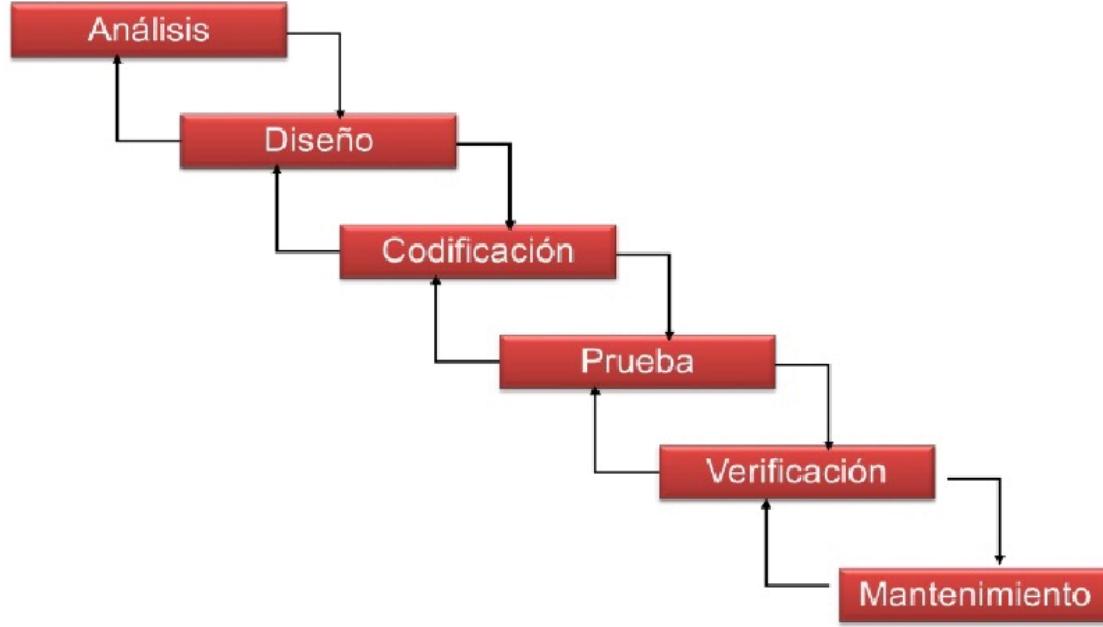


Características que hacen que las pruebas sean adecuadas:



Modelo Cascada

- En el modelo en Cascada, las actividades de desarrollo (por ejemplo, análisis de requisitos, diseño, codificación, prueba) se completan una tras otra.
- Las actividades de prueba sólo ocurren después de que todas las demás actividades de desarrollo hayan sido completadas.

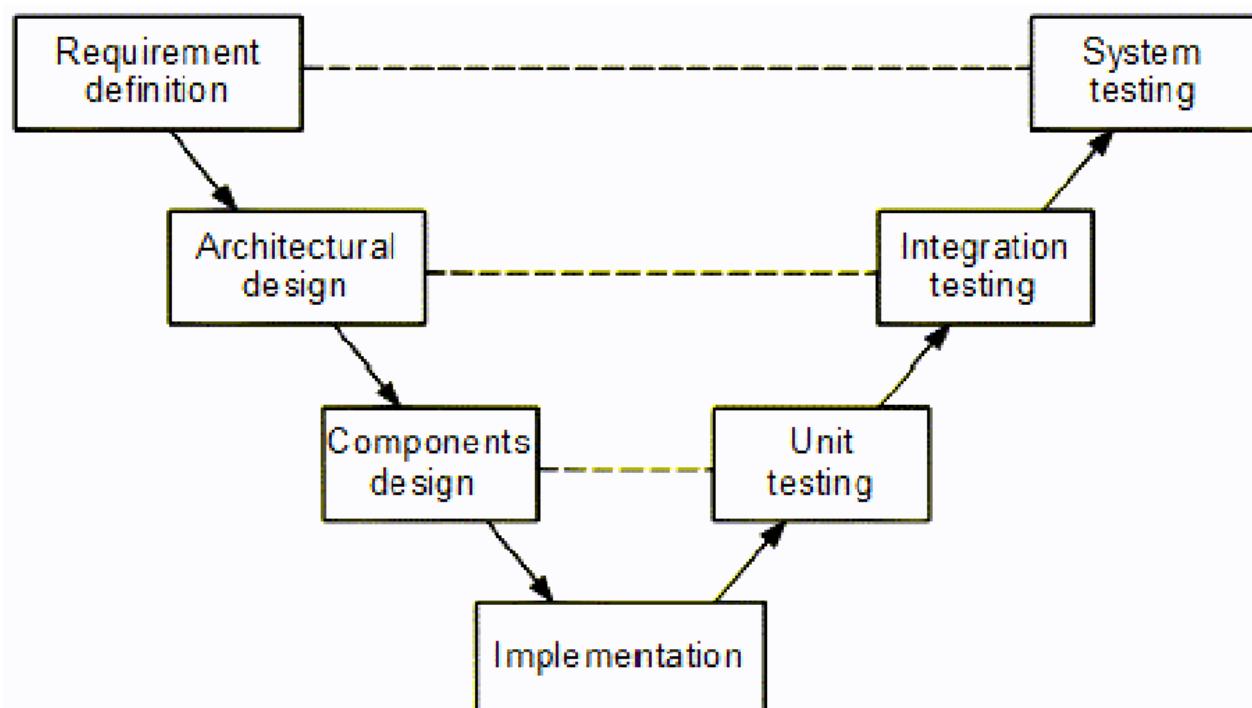


Modelo en “V” o Secuencial

Describe el proceso de desarrollo de software como un flujo lineal y secuencial de actividades.

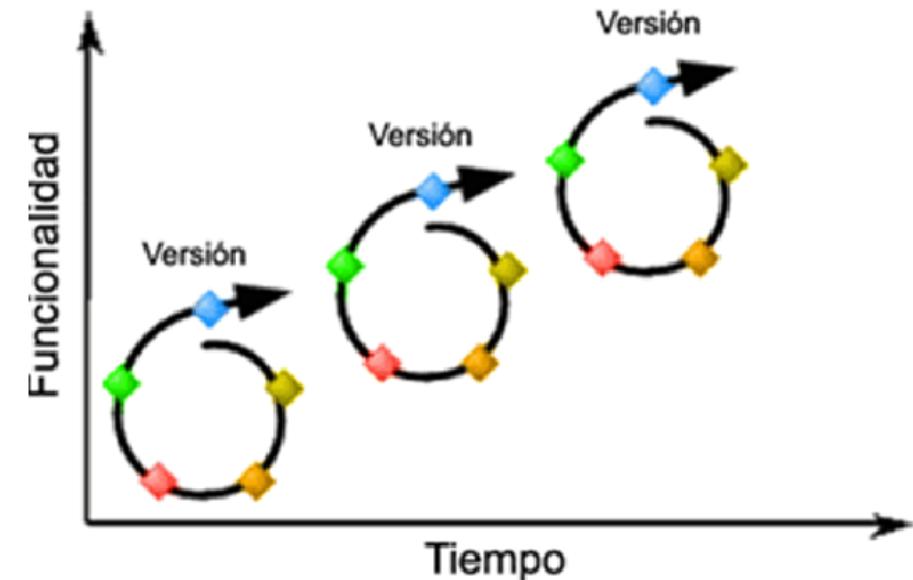
Esto significa que cualquier fase del proceso de desarrollo debe comenzar cuando se haya completado la fase anterior.

- Usualmente es dirigido por riesgos de presupuestos y de calendarios.
- El desarrollo y el Testing son dos ramas que apuntan a los mismos niveles.
- Los niveles dependen del proyecto y el producto que se esta desarrollando.
- ¡Es difícil planificar con tanta antelación!
- ¡Cuando los planes fracasan, las pruebas sufren (al final)!



Modelo Iterativo, Evolutivo o Incremental

- Dirigido por los riesgos de calendarios para alcanzar un mercado o una fecha
- El conjunto de características crece en torno a la funcionalidad Básica
- Puede liberar algo y en cualquier momento una vez que la funcionalidad básica se complete



A diferencia de los modelos secuenciales pueden ofrecer software utilizable en semanas o incluso días, pero sólo pueden ofrecer el conjunto completo de requisitos durante un periodo de meses o incluso años.

Modelo Iterativo, Evolutivo o Incremental

- **Rational Unified Process:** Cada iteración tiende a ser relativamente larga (Ej, dos a tres meses), y los incrementos de las prestaciones son proporcionalmente grandes.
- **Scrum:** Cada iteración tiende a ser relativamente corta (Ej, horas, días o pocas semanas), y los incrementos son proporcionalmente pequeños, como unas pocas mejoras y/o dos o tres prestaciones nuevas.
- **Kanban:** Implementado con o sin iteraciones de longitud fija, que puede ofrecer una sola mejora o prestación una vez finalizada, o puede agrupar prestaciones para liberarlas de una sola vez.
- **Espiral (o prototipado):** Implica la creación de incrementos experimentales, algunos de los cuales pueden ser reelaborados en profundidad o incluso abandonados en trabajos de desarrollo posteriores.

Integración de Sistemas

Muchos proyectos involucran la integración de componentes.

Opciones de mitigación:

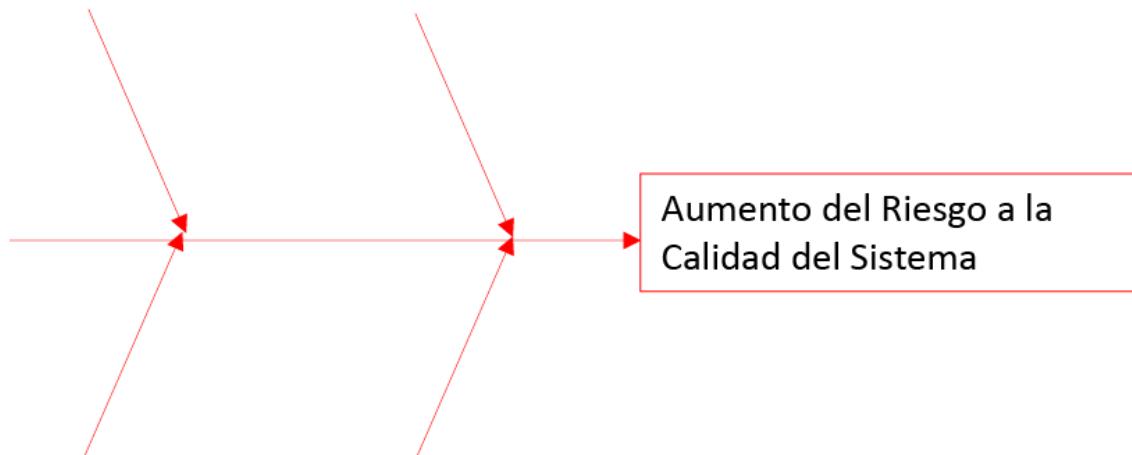
- Integrar y gestionar las pruebas de proveedores.
- Confiar en las pruebas de componente del vendedor
- Corregir las pruebas o calidad del vendedor
- Ignorar y reemplazar sus pruebas

Acoplamiento de componentes

Problemas de Calidad del Vendedor

Irreemplazabilidad del código

Escencialidad de componentes



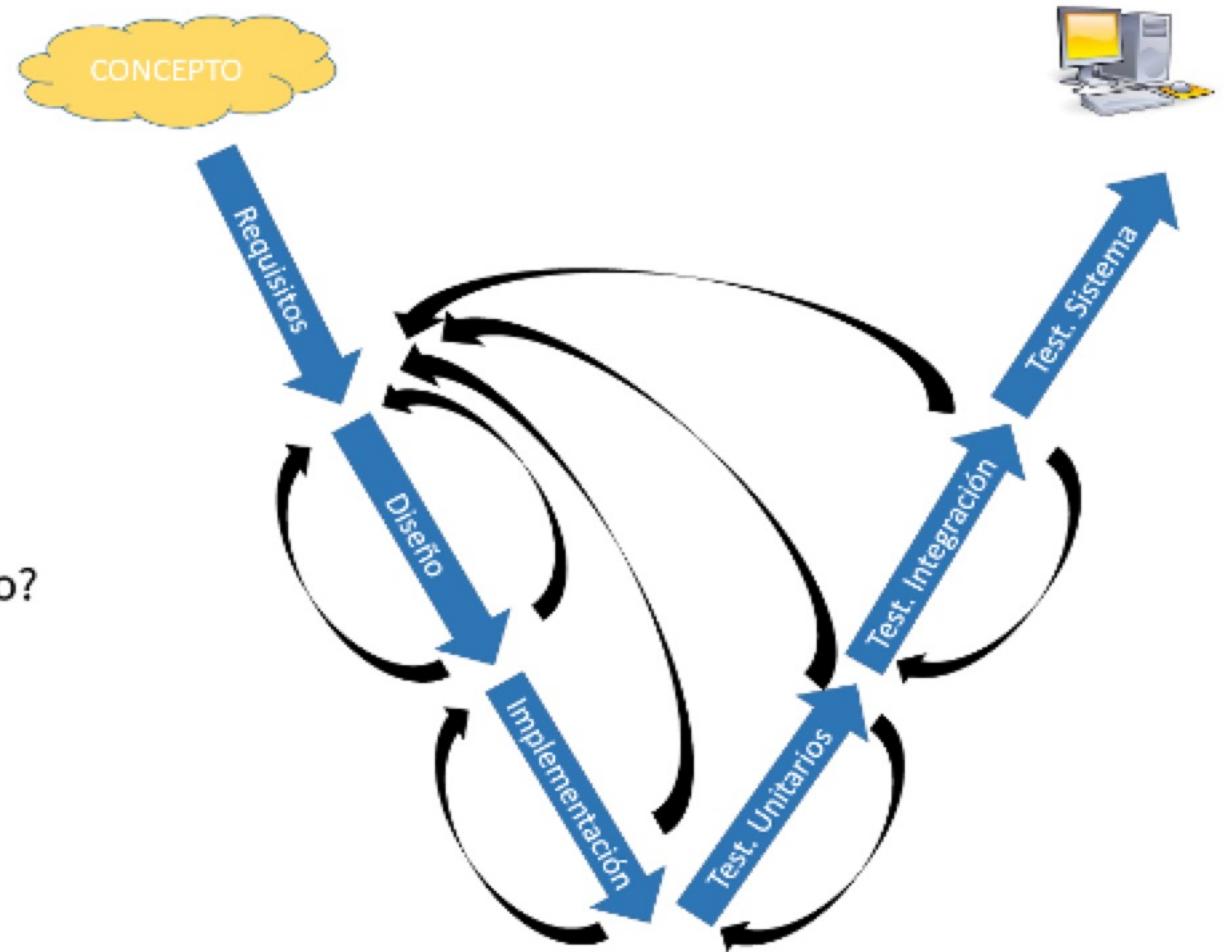
Verificación y validación

- Verificación

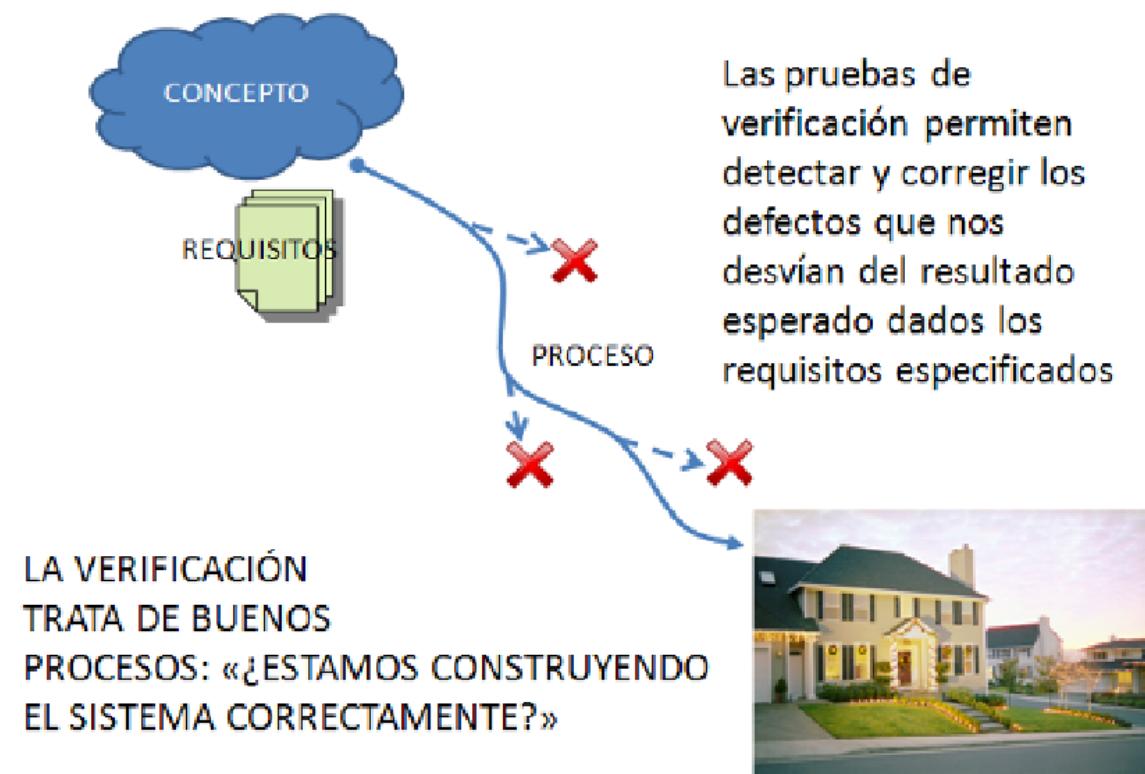
- ¿Estamos construyendo correctamente el producto?
- Buscar defectos en las fases

- Validación

- ¿Estamos construyendo el producto correcto?
- Buscar defectos en el sistema, basado en la fase de entregas.



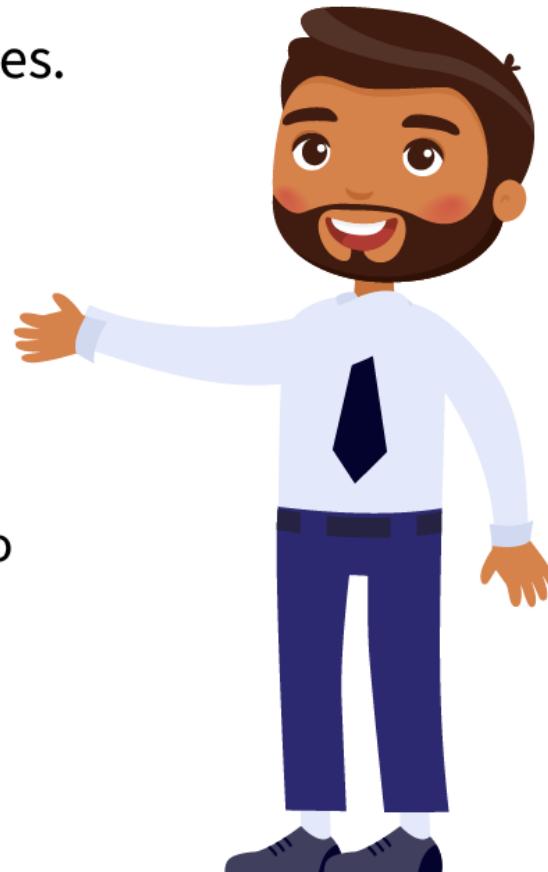
Verificación y validación



Estándar de Procesos de Software IEEE 12207

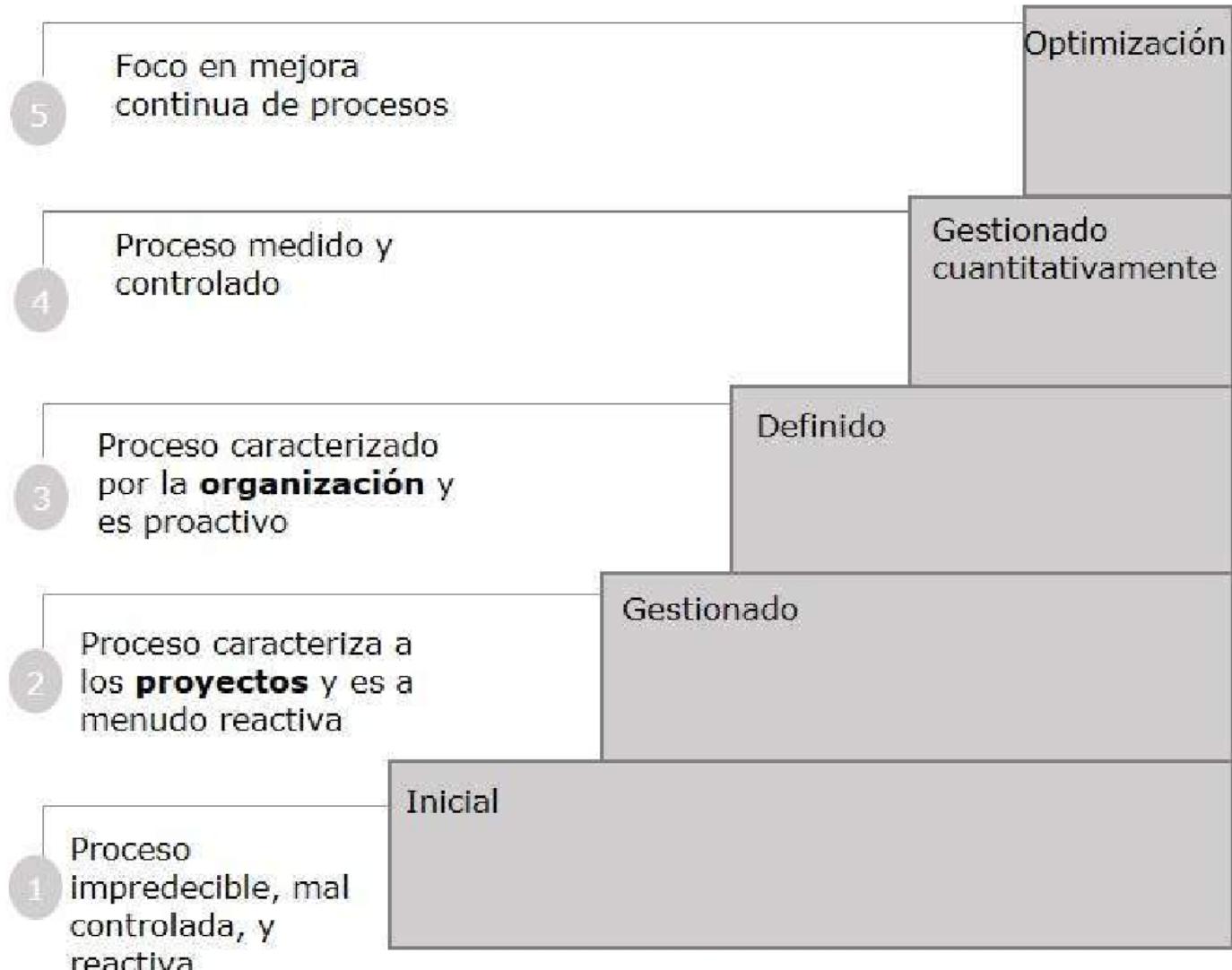
Alcance

- Propósito, aplicación adaptación, conformidad, limitaciones.
- Referencias normativas
- Definiciones
- Aplicación
 - Procesos de ciclo de vida, adaptación
- Procesos de ciclo de vida
 - Adquisición, suministro, desarrollo, operación, mantenimiento
- Procesos de Apoyo
 - Publicaciones técnicas, CM, QA, etc,
- Procesos de ciclo de vida organizaciones.



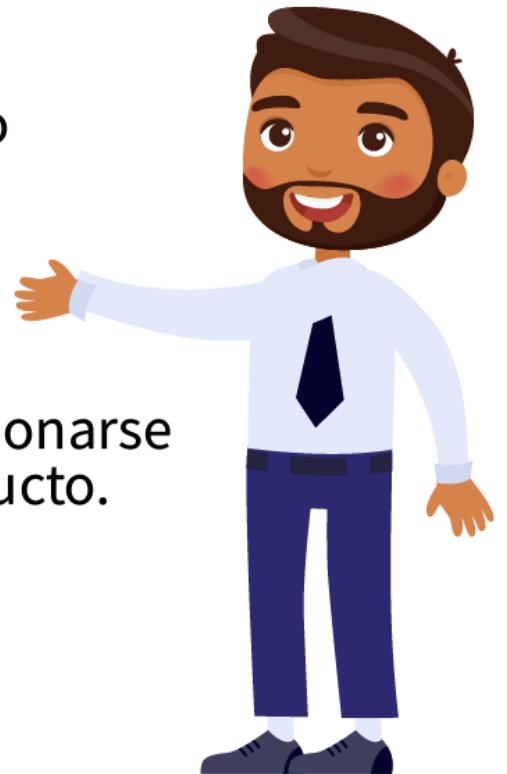
Madurez del proceso CMMI

- El **CMMI** es un enfoque de mejora de procesos que provee a las organizaciones de los elementos esenciales para un **proceso** efectivo.
- Mide la madurez del desarrollo del software en una escala del 1 al 5.



Independencia de los Modelos

- Características generales de buenas pruebas:
 - Tienen actividades de pruebas para cada actividad de desarrollo
 - Los niveles de pruebas tienen objetivos enfocados
 - El análisis y diseño comienzan temprano
- Los modelos de ciclo de vida de desarrollo de software deben seleccionarse y adaptarse al contexto de las características del proyecto y del producto.



Niveles o Fases de Pruebas

“Son grupos de actividades de prueba que se organizan y gestionan conjuntamente”

Pruebas de componentes (o Unidad)

- **Objetivo:** Encontrar defectos, crear confianza, reducir riesgo en las piezas individuales del sistema sometido a pruebas.
- **Tipos de prueba:** de funcionalidad, sobre el uso de recursos de rendimiento y estructurales.
- **El ítem sometido a pruebas puede ser:**
Componentes, programas, programas de conversión.
- **Las armaduras y herramientas:** A nivel de API (drivers y stubs)
- **Responsable:** usualmente programadores, pero puede variar.

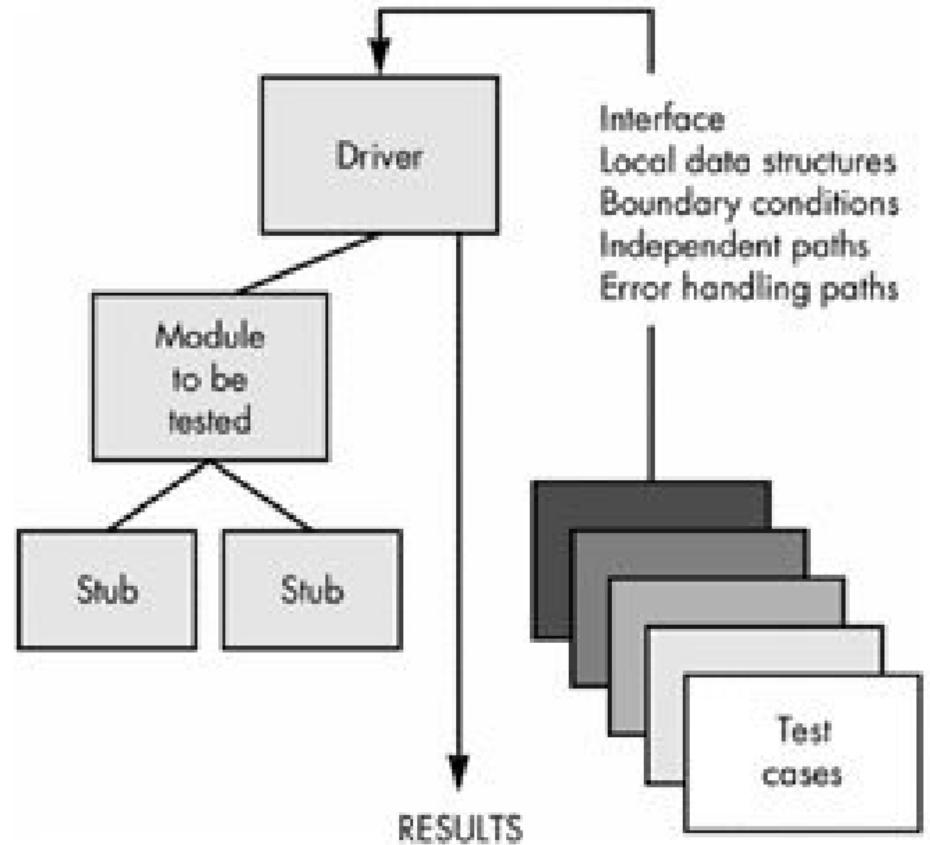
Proceso de pruebas de unidad / componente

- Pruebas que...
 - Implican acceso al código
 - Se ejecutan en un entorno de desarrollo
 - Necesitan drivers, stubs..
 - Se realizan por el programador que esquivó el código
- A menudo los defectos se corrigen sin haber sido informados, ello reduce la transparencia del proceso.



Drivers y Stubs

- Simulan partes del flujo de llamadas
- A veces requiere previa configuración de datos
- **Driver:** Función que llama al módulo sometido a pruebas.
- **Stub:** Función que llama directa o indirectamente los módulos sometidos a pruebas.



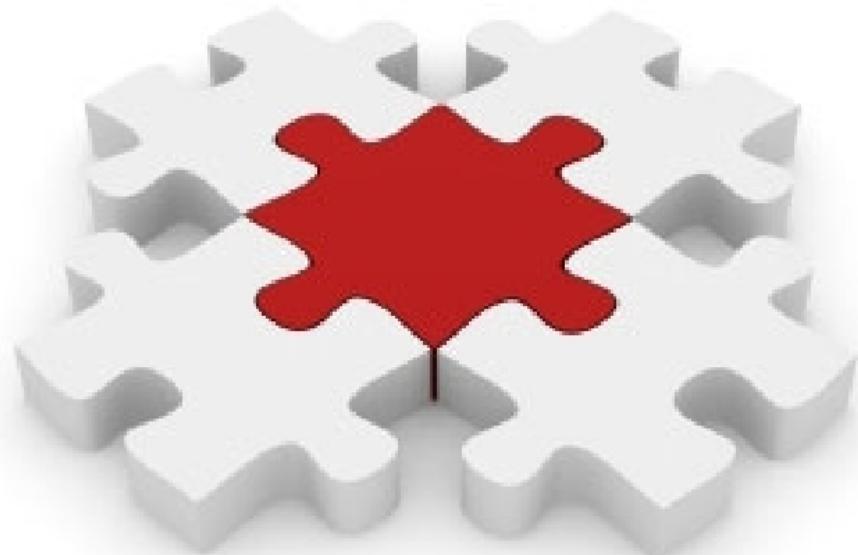
Pruebas de Integración

- **Objetivo:** Encontrar defectos, crear confianza, reducir riesgo en las relaciones e interfaces entre pares y grupos de componentes a medida que las piezas se van juntando.
- **Base:** Diseño, arquitectura, esquemas, flujos de datos, etc.
- **Tipos de prueba:** de funcionalidad, sobre el uso de recursos de rendimiento y estructurales.
- **El ítem sometido a pruebas puede ser:**

Compilaciones, redes centrales. Infraestructura, interfaces, configuraciones, datos de configuración.
- **Responsable:** usualmente probadores y programadores

Niveles de pruebas de integración

- **Pruebas de integración de componentes:** pruebas de relaciones entre unidades o componentes.
- **Pruebas de integración de sistemas:** Pruebas de interacciones entre sistemas enteros, luego de pruebas de sistema.



Pruebas de Sistema

- **Objetivo:** Encontrar defectos, crear confianza, reducir riesgo en todos los comportamientos, funciones y respuestas generales.
- **Base:** Requisitos, diseño de alto nivel, casos de uso, listas de comprobación.
- **Tipos de prueba:** de funcionalidad, seguridad, rendimiento, fiabilidad, etc.
- **El ítem sometido a pruebas puede ser:** Todo el sistema en un ambiente lo más realista posible.
- **Responsable:** Típicamente probadores independientes

Pruebas de aceptación

- **Objetivo:** Demostrar que el producto está listo para su despliegue y liberación.
- **Base:** Requisitos, casos de uso, procesos de negocio, contratos, experiencia y riesgos de calidad.
- **Tipos de prueba:** de funcionalidad, portabilidad, rendimiento.
- **El ítem sometido a pruebas puede ser:** Sistema integrado totalmente.
- **Responsable:** A menudo son los clientes, pero también probadores independientes.

Clasificación por objetivos específicos

- **Pruebas estáticas**, las cuales *no involucran la ejecución del ítem* sometido a pruebas.

Consisten en:

- Análisis estático (implica utilización de herramientas para evaluar el ítem sometido a pruebas)
- Revisiones, las cuales involucran la evaluación manual del ítem sometido a pruebas.

- **Pruebas dinámicas**, que *involucran la ejecución del ítem* sometido a pruebas.

Subtipos de pruebas dinámicas

- **Prueba estructural o de caja blanca.** El diseño de estas pruebas está basado principalmente en la estructura del ítem sometido a prueba.
- **Prueba de comportamiento o de caja negra.** El diseño de estas pruebas se basa principalmente en el comportamiento del ítem sometido a prueba.
- **Pruebas basadas en la experiencia.** El diseño de los casos de prueba se base en el conocimiento y experiencia del probador, los cuales pueden abarcar tanto los aspectos de comportamiento como estructurales.



Tipos de Prueba de comportamiento

Se dividen en **Pruebas Funcionales y No Funcionales**.

- **Las Pruebas Funcionales:** Prueban el “Qué” del comportamiento del ítem, mientras las No Funcionales prueban el “Cómo” del comportamiento del ítem.
- **El programa de estudios básico del ISTQB utiliza el estándar ISO 9126 para clasificar las pruebas de comportamiento** (funcionales o no funcionales)

Un tipo de prueba relacionado con el cambio, es la prueba de confirmación, llamada también repetición de pruebas.

Pruebas no funcionales

- Evalúan las características de sistemas y software, como la usabilidad, la eficiencia del desempeño o la seguridad.
- Se pueden realizar pruebas no funcionales en todos los niveles de prueba, y se deben realizar tan pronto como sea posible.
- El descubrimiento tardío de defectos no funcionales puede ser extremadamente **caro**.

Pruebas de caja Blanca

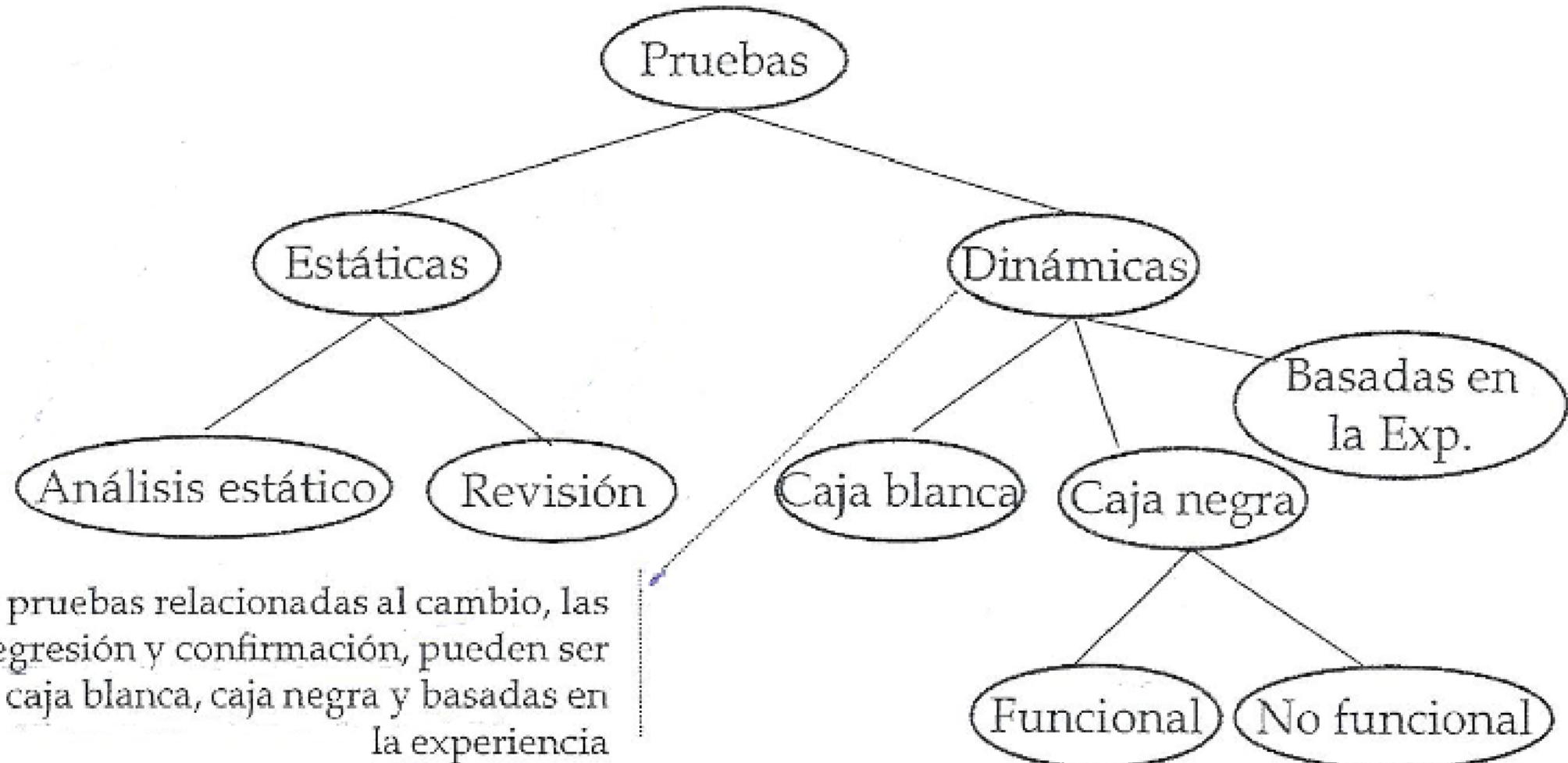
- Son aquellas basadas en la estructura interna del sistema o en su implementación.
- La estructura interna puede incluir código, arquitectura, flujos de trabajo y/o flujos de datos dentro del sistema.
- Se puede medir la intensidad de la prueba de caja blanca a través de la cobertura estructural. La cobertura estructural es la medida en que algún tipo de elemento estructural ha sido practicado mediante pruebas, y se expresa como un porcentaje del tipo de elemento cubierto.

Pruebas Asociadas al Cambio

Se realizan para confirmar que los cambios han corregido el defecto o implementado la funcionalidad correctamente, y no han causado ninguna consecuencia adversa imprevista.

- **Prueba de confirmación:** Tiene como objetivo confirmar que el defecto original se ha solucionado de forma satisfactoria.
- **Prueba de regresión:** implica la realización de pruebas para detectar estos efectos secundarios no deseados.

Tipos de Prueba - Resumen



Pruebas de Mantenimiento

- Son aquellas pruebas que se realizan como parte del mantenimiento, en estas se evalúa el éxito con el que se realizaron los cambios y se comprueba los posibles efectos secundarios
- Los cambios de diversa índole son casi inevitables en el software y en los sistemas entregados, ya sea para corregir defectos descubiertos en el uso operativo, para añadir nuevas funcionalidades o para eliminar o alterar funcionalidades ya entregadas.

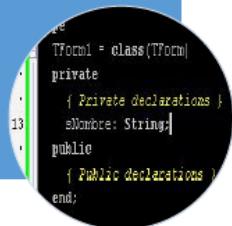
Pruebas de Mantenimiento

Abordan el cambio en sí mismo.

“Lo que fue modificado y lo que no se debería modificar.

- Mejoras, creación de defectos, cambios de entornos operativos, parches.

Modificación



- Soporte del nuevo entorno

Migración



- Final de vida útil de un sistema o subsistema.

Retiro



Pruebas de Seguridad

- **Amenazas de seguridad**

Virus

Infiltración de servidores

Denegación de servicios(Acciones que saturan una aplicación, servicio o sistema)

Ransomware

- **Supuestos equivocados:**

La Encriptación HTTPS resuelve problemas de seguridad.

La compra de un Firewall resuelve los problemas.

Los administradores de sistemas o redes calificados pueden resolver todos los problemas

Interoperatividad

La mayoría de las aplicaciones tienen que correr con otras aplicaciones en el mismo entorno, así que tenemos que comprobar que el objeto de pruebas puede funcionar correctamente con estos programas, con el sistema operativo o los sistemas, que lo hospedarán, con las bases de datos que éste interactúa o cohabita y así sucesivamente.

- **Incluye:**
 - Interfaces
 - Intercambio de datos.
 - Puede ser uno a uno o como colección de componentes



Rendimiento y fiabilidad

- **Rendimiento:**

- “Demasiado lento”: Degradación de rendimiento inaceptable en el tiempo.

- **Fiabilidad**

- En caso de fallas debe poder completar las funciones normales.
 - “Sistema se cae o cuelga aleatoriamente.

Estrés, capacidad y volumen

Estrés: Condiciones externas causan fallas.

Capacidad: Problemas de funcionalidad, rendimiento o fiabilidad.

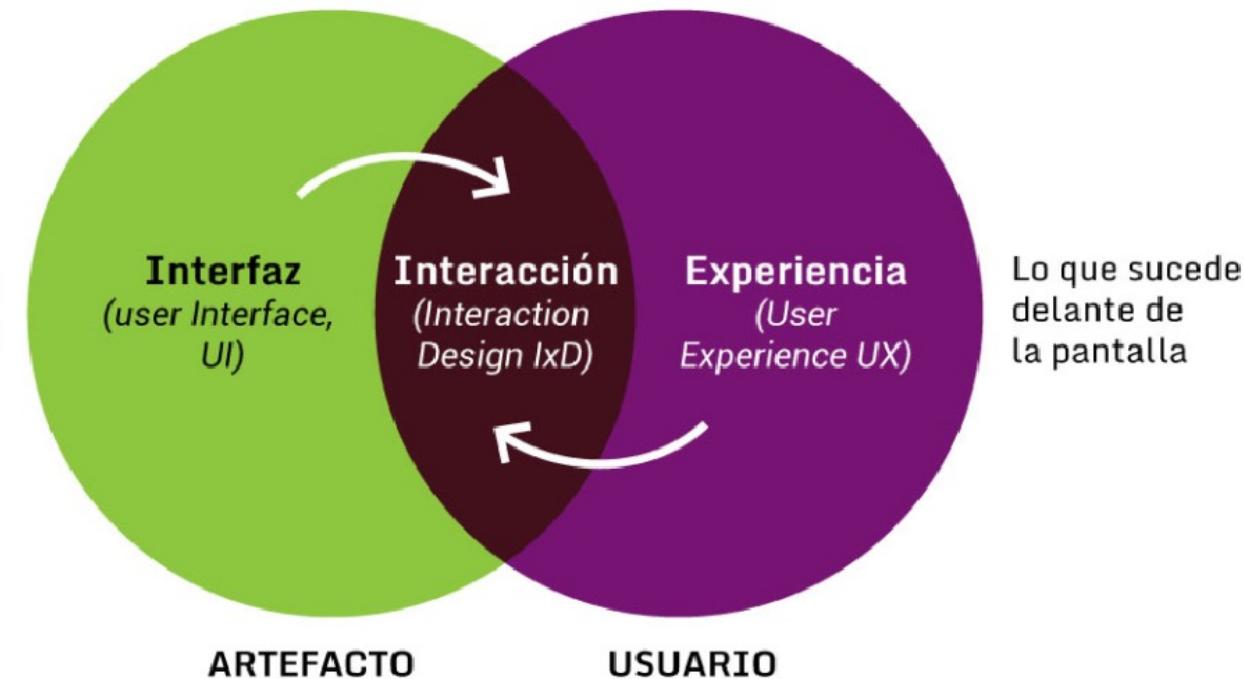
Volumen: Problemas debido a la tasa de flujo de datos.



Usabilidad e interfaz de usuarios

- UX
- Las interfaces deben ser coherentes con:
 - Flujo de trabajo.
 - Especificaciones.
 - Usuario, público objetivo.

Lo que sucede en la pantalla



Lo que sucede delante de la pantalla

Configuración y portabilidad

- La plataforma debe considerarse en diferentes maneras
- Las familias de plataformas pueden soportar diversas configuraciones de hardware
- **¿Cómo son manejados?**
 - Cambios de configuraciones
 - Añadir espacio al disco
 - Agregar memoria
 - Actualizar y agregar CPU



Otras pruebas Funcionales / No funcionales

- Localización (interfaz de usuario)
- Localización (operacional)
- Estándares y cumplimiento regulatorio
- Manejo de errores y recuperación
- Recuperación de desastres
- En red/internet o distribuido
- Sincronización y la coordinación
- Calidad de datos
- Conversión de datos
- Operaciones
- Instalación
- Desinstalación
- Manejo de fecha y hora
- Documentación
- Y muchos otros ...

Regresión

- Son pruebas que intentan descubrir bugs, carencias de funcional o divergencias funcionales causados por la realización de un cambio en el aplicativo.
- **Los hallazgos pueden ser de ámbito:**
 - Local (La corrección crea nuevo defecto)
 - Expuesto (La corrección revela un defecto existente)
 - A distancia (La corrección en un área afecta a otra)

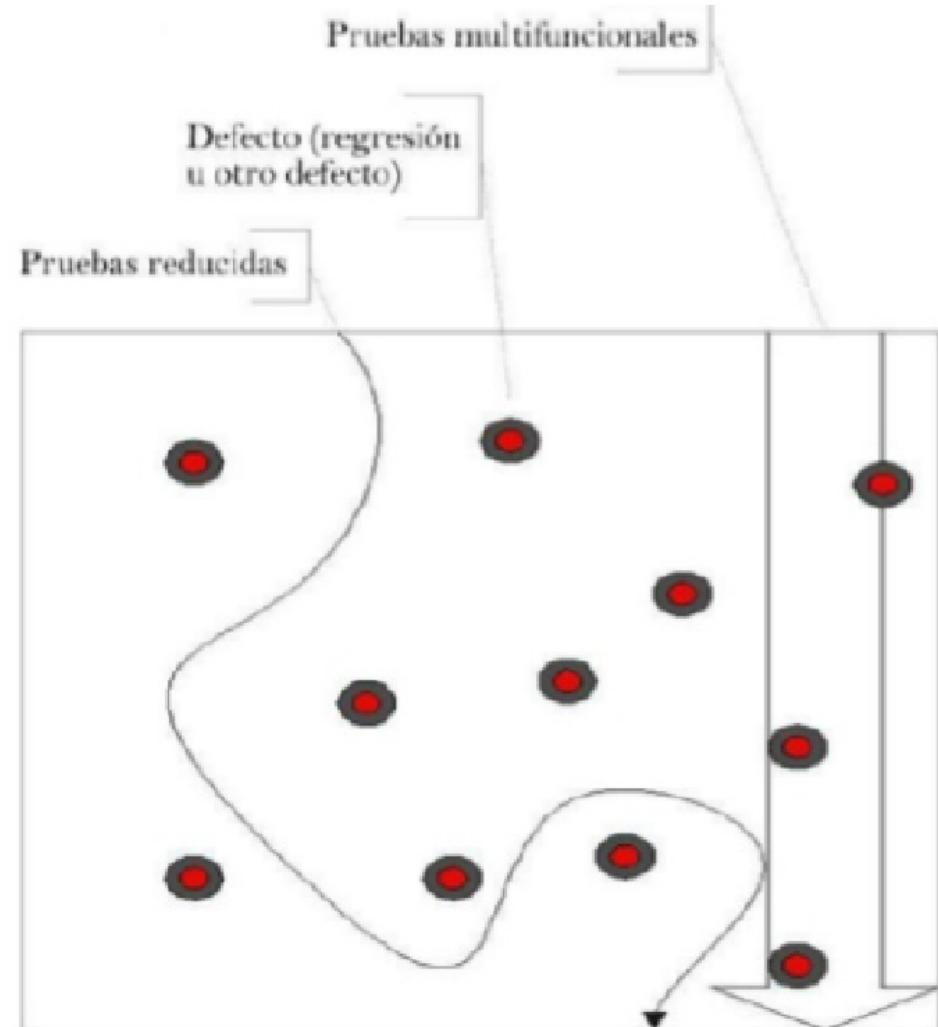


Estrategia de regresión 1: Repetir todas las pruebas

- Si las pruebas abarcan los riesgos de calidad importantes, entonces la repetición de todas las pruebas deberían encontrar las observaciones más importantes.
- La automatización es el único medio práctico para sistemas complejos grandes.

Estrategia 2: Repetir algunas pruebas

- A menudo, la automatización completa es imposible.
- Se debe seleccionar algunas pruebas en función a:
 - Trazabilidad
 - Análisis de cambio / Impacto
 - Análisis de riesgos
- Utilice pruebas multifuncionales para “pruebas de regresión accidental”
- Se puede usar la cobertura de código para evaluar el nivel de riesgo.



Otras tres estrategias de regresión

- Generación de versión más lentamente
- Combinación de parches de emergencia para permitir flexibilidad manteniendo el riesgo de regresión bajo.
- Utilización de pruebas de clientes o usuarios
 - Beta de software para mercados masivos
 - Versiones pilotos, de etapa o fase.



- Introducción y Objetivos
- I. Fundamentos de Pruebas
- II. Pruebas a través del Ciclo de Vida de Software
- III. Técnicas Estáticas**
- IV. Técnicas de Prueba
- V. Gestión de Pruebas
- VI. Soporte de Herramientas para Pruebas

CAPITULO III

Consisten en evaluación del código o productos de trabajo de forma manual o basada en herramientas.

Diferenciando pruebas estáticas y dinámicas



Pruebas estáticas

- Detectan defectos en los productos de trabajo
- Usadas para mejorar la consistencia y la calidad interna de los productos de trabajo



Pruebas dinámicas

- Identifican los fallos causados por defectos cuando se ejecuta el software.
- Valida comportamientos visibles desde el exterior

Pruebas y Análisis para la búsqueda de defectos

Análisis Estático

- Análisis de artefactos de software, requisitos o código.
- Permite detectar errores en fase temprana de escritura
- Implica análisis por medio de una herramienta
- Puede referirse a la revisión y mantenimiento de documentación
- Encuentra defectos, no síntomas



Pruebas Estáticas

- Pruebas de un componente o sistema en un nivel de la especificación o implementación sin ejecución de ese software.
- Involucra la utilización de las revisiones y las herramientas.



```
def query_set(self, user):
    """Return a querySet of connections for user."""
    set1 = self.filter(from_user=user).select_related(depth=1)
    set2 = self.filter(to_user=user).select_related(depth=1)
    return set1 | set2

def are_connected(self, user1, user2):
    if self.filter(from_user=user1, to_user=user2).count() > 0:
        return True
    if self.filter(from_user=user2, to_user=user1).count() > 0:
        return True
    return False

def request(self, user1, user2):
    """Creates proper object regardless of the order of users in a connection"""
    connection = self.filter(from_user=user1, to_user=user2)
    if not connection:
        connection = self.filter(from_user=user2, to_user=user1)
    connection.delete()
    connection.save()

models.py
```



Pruebas Dinámicas

- Requiere ejecución de la aplicación
- Es más lento y necesita un proceso completo de testeo.
- Permite ver muchos errores que quedan ocultos en un análisis estático

Búsqueda de defectos

Análisis Estático

¿Qué podemos analizar?

Código del programa

- Flujo de control
- Flujo de datos

Modelos del programa

- Simulaciones
- Antipatrones

Salidas generadas

- Html/XML

Documentos de requisitos y diseño

- Esquemas de bases de datos
- Arquitectura

Beneficios del Análisis Estático

Detección temprana y más barata de defectos.

Advertencias sobre agrupaciones de defectos por programación peligrosa o alta complejidad.

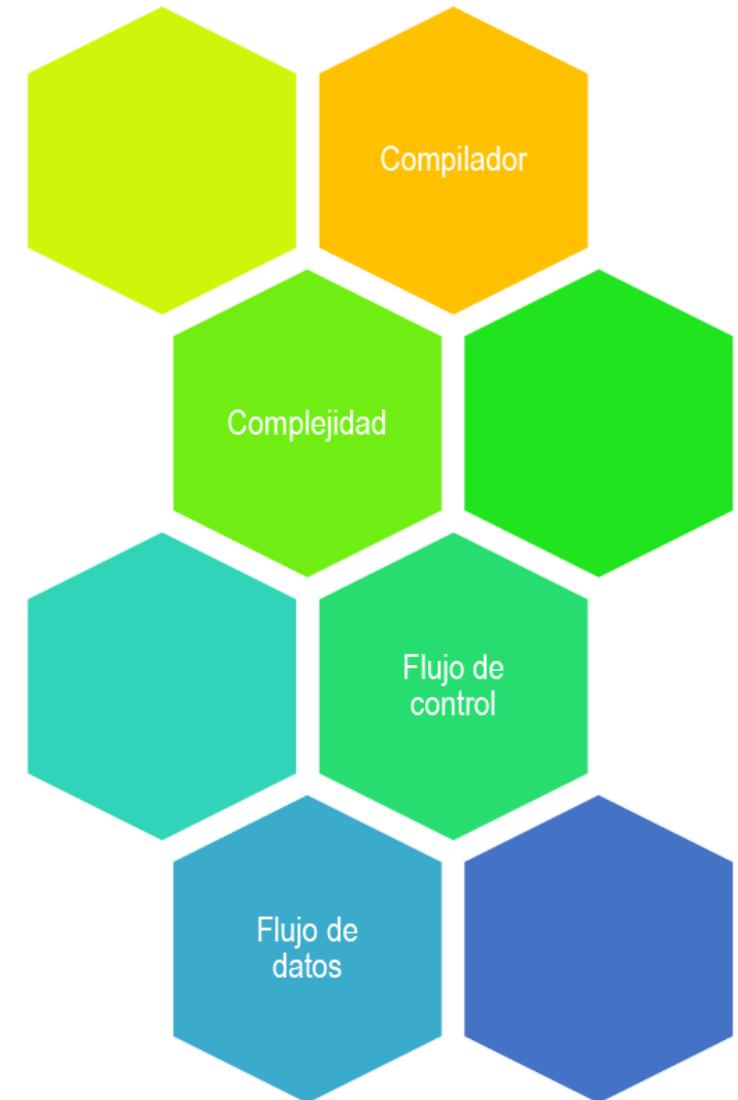
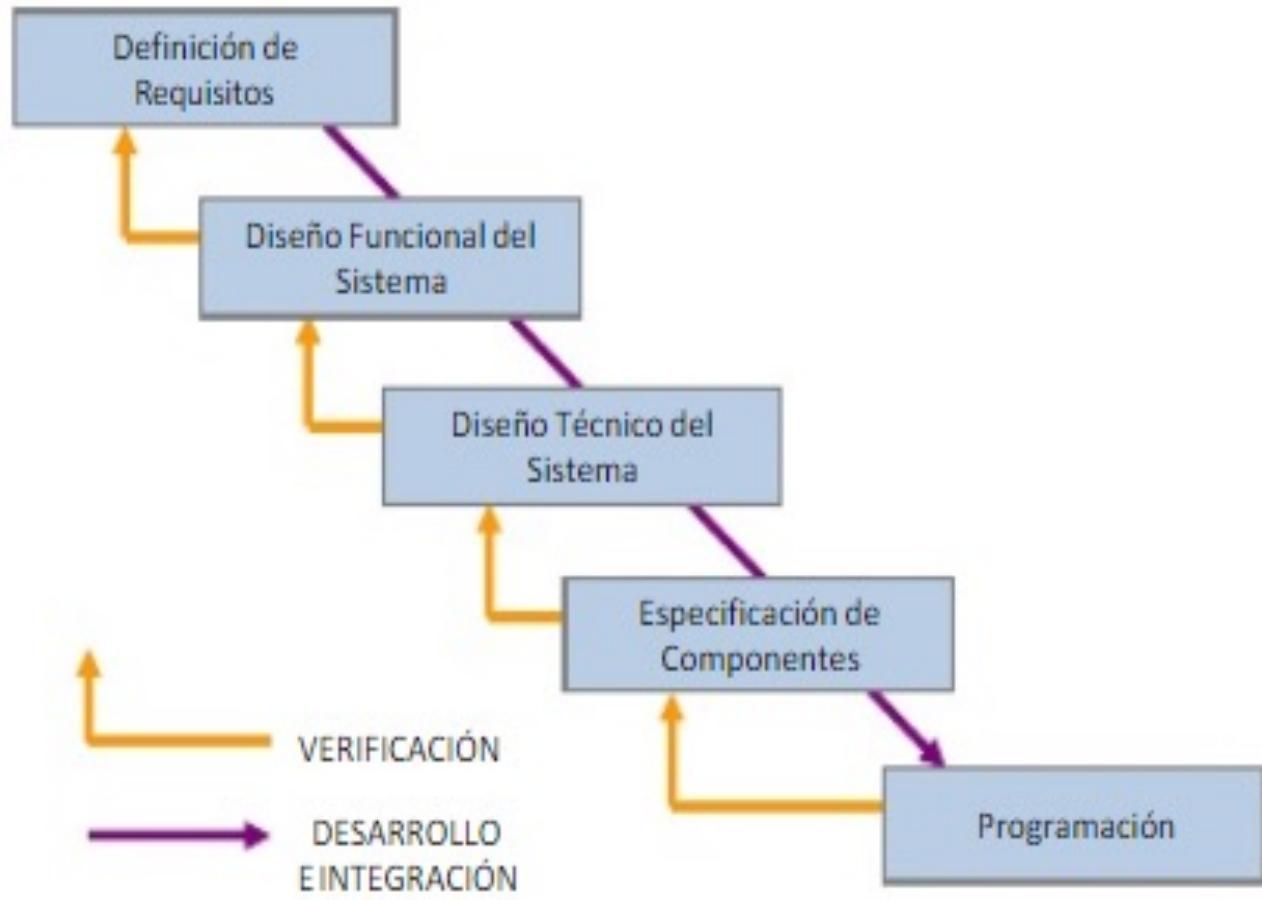
Detección de dependencias e inconsistencias

Prevención basada en métricas y lecciones aprendidas

Defectos Típicos

- Defectos en las especificaciones
- Defectos en el diseño y arquitectura del software
- Defectos en las especificaciones de interfaces
- Mantenibilidad insuficiente
- Desviaciones con respecto a estándares acordados
- Variables con valor no definido
- Inconsistencia de interfaz entre módulos y componentes
- Variables no utilizadas
- Lógica faltante o errónea
- Complejidad excesiva
- Código inaccesible (muerto)
- Vulnerabilidades de seguridad
- Insuficiencia de Cobertura de prueba

Proceso de análisis estático



Herramientas para las pruebas estáticas

- Ortografía, gramática y comprobación de dificultad de lectura.
- Herramientas para comprobar constructos peligrosos de programación. (J-test, Safer, C, lint, ...)
- Herramientas de análisis de complejidad
- Simuladores de sistemas



El proceso de revisión

Tipos de revisiones según IEEE 1028

Por el proceso básico de una revisión

Inspección (“inspection”)

Revisión guiada (“walkthrough”)

Revisión técnica (“technical review”)

Revisión informal “informal review”

Por naturaleza del objeto a revisar

De producto

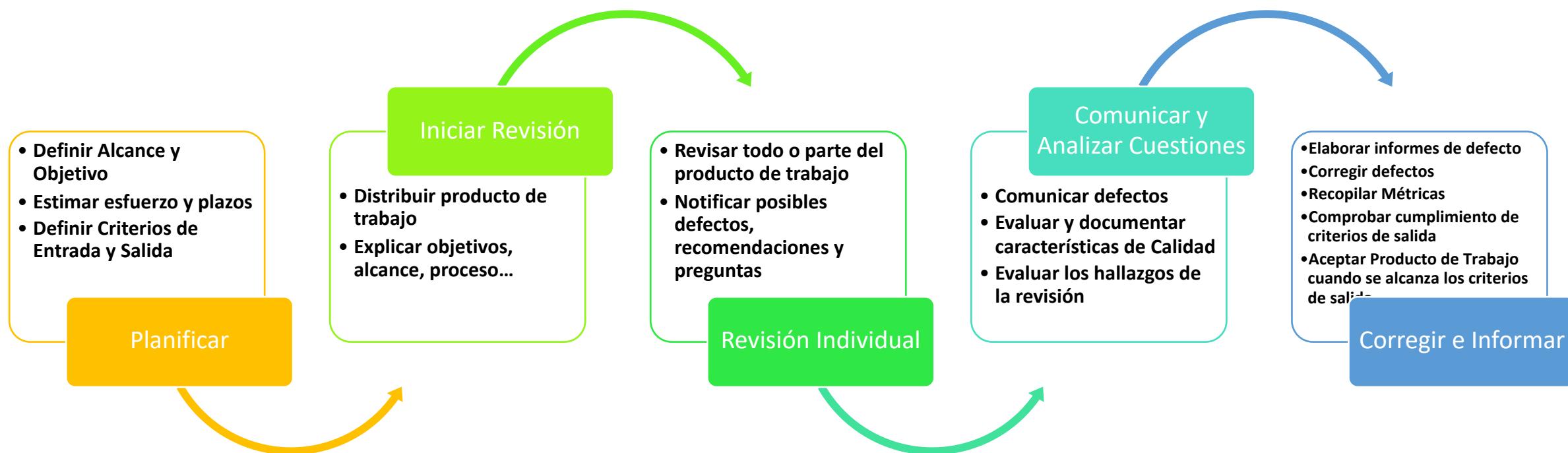
Roles y responsabilidades de una revisión Formal

Autor	Director	Facilitador (Moderador)	Líder de Revisión	Revisores
<ul style="list-style-type: none"> • Crea el producto de trabajo bajo revisión • Corrige los defectos en el producto de trabajo bajo revisión (si fuera necesario) 	<ul style="list-style-type: none"> • Responsable de la planificación de la revisión. • Decide acerca de la ejecución de las revisiones. • Asigna personal, presupuesto y tiempo. • Supervisa la rentabilidad en curso • Ejecuta las decisiones de control en caso de resultados inadecuados 	<ul style="list-style-type: none"> • Asegura el funcionamiento efectivo de las reuniones de revisión. • Si fuera necesario, realiza una mediación entre los distintos puntos de vista. • A menudo, es la persona de la que depende el éxito de la revisión 	<ul style="list-style-type: none"> • Asume la responsabilidad general de la revisión • Decide quiénes estarán involucrados y organiza cuándo y dónde se llevará a cabo 	<ul style="list-style-type: none"> • Pueden ser expertos en la materia, personas que trabajan en el proyecto, implicados, y/o personas con antecedentes técnicos o de negocio específicos • Identifican posibles defectos en el producto de trabajo bajo revisión • Pueden representar diferentes perspectivas

Tipos de revisión

REVISIÓN INFORMAL	REVISIÓN GUIADA	REVISIÓN TÉCNICA	INSPECCIÓN
<ul style="list-style-type: none"> • Objetivo principal: detectar defectos potenciales. • Posibles objetivos adicionales: generar nuevas ideas o soluciones, resolver de forma rápida problemas menores. • No se basa en un proceso formal (documentado). • Puede no implicar una reunión de revisión. • Puede ser realizado por un compañero de trabajo del autor (comprobación entre amigos) o por más personas. • Se pueden documentar los resultados. • Varía en utilidad dependiendo de los revisores. • El uso de listas de comprobación es opcional. • Utilizado con mucha frecuencia en el desarrollo Ágil. 	<ul style="list-style-type: none"> • Objetivos principales: detectar defectos y evaluar la conformidad con estándares y especificaciones. • Posibles objetivos adicionales: intercambio de ideas sobre técnicas o variaciones de estilo, formación de los participantes, alcanzar un consenso. • La preparación individual antes de la reunión de revisión es opcional. • Normalmente, la reunión de revisión está dirigida por el autor del producto de trabajo. • El escriba es obligatorio. • El uso de listas de comprobación es opcional. • Se pueden elaborar registros de posibles defectos e informes de revisión. • En la práctica puede variar de bastante informal a muy formal. 	<ul style="list-style-type: none"> • Objetivos principales: lograr un consenso, detectar defectos potenciales. • Posibles objetivos adicionales: evaluar la calidad y generar confianza en el producto de trabajo, generar nuevas ideas, motivar y capacitar a los autores para mejorar los futuros productos de trabajo, considerar implementaciones alternativas. • Los revisores deben ser pares técnicos del autor y expertos técnicos en la misma u otras disciplinas. • Es necesario que haya una preparación individual antes de la reunión de revisión. • La reunión de revisión es opcional, lo ideal es que la dirija un facilitador capacitado (normalmente no el autor). • El escriba es obligatorio, lo ideal es que no sea el autor. • El uso de listas de comprobación es opcional. • Normalmente se elaboran registros de defectos potenciales e informes de revisión. 	<ul style="list-style-type: none"> • Objetivos principales: detectar defectos potenciales, evaluar la calidad y generar confianza en el producto de trabajo, prevenir futuros defectos similares mediante el aprendizaje del autor y el análisis de la causa raíz. • Posibles objetivos adicionales: motivar y capacitar a los autores para que mejoren los futuros productos de trabajo y el proceso de desarrollo de software, alcanzar un consenso. • Sigue un proceso definido con resultados documentados formales, basados en reglas y listas de comprobación. • Utiliza roles claramente definidos, que son obligatorios. • Es necesario que haya una preparación individual antes de la reunión de revisión. • Los revisores son pares del autor o expertos en otras disciplinas que son relevantes para el producto de trabajo. • Se utilizan criterios especificados de entrada y salida. • El escriba es obligatorio. • La reunión de revisión es dirigida por un facilitador capacitado (no el autor). • El autor no puede actuar como líder de revisión, lector o escriba. • Se elaboran registros de defectos potenciales e informes de revisión. • Se recopilan y utilizan métricas para mejorar el proceso de desarrollo.

Proceso de revisión de productos de trabajo



Factores de éxito para las revisiones

- Las revisiones se deben desarrollar orientadas al logro de objetivos , es decir, las desviaciones en el objeto revisado deben ser establecidas de forma imparcial
- El autor del objeto revisado deberá ser motivado de una forma positiva por la revisión (“su documento será aún mejor” en lugar de “su documento es de baja calidad”)
- Uso sistemático de las Técnicas y plantillas implantadas
- El uso de listas de comprobación mejorará la eficiencia de la revisión
- Para la ejecución apropiada de las revisiones será necesario un presupuesto apropiado(10% al 15% del costo total del desarrollo)
- Hacer uso del efecto de las lecciones aprendidas, utilizar la retroalimentación para implementar un proceso de Mejora continua

Taller

- Introducción y Objetivos
- I. Fundamentos de Pruebas
- II. Pruebas a través del Ciclo de Vida de Software
- III. Técnicas Estáticas
- IV. Técnicas de Prueba**
- V. Gestión de Pruebas
- VI. Soporte de Herramientas para Pruebas

CAPITULO IV

Son procedimientos, reglas, normas o protocolos que ayudan a identificar las condiciones de prueba, los casos de prueba y los datos de prueba.

Especificaciones IEEE829

Diseño de Pruebas

Se constituye de una colección de casos de prueba y condiciones.

- Secciones:
 - Identificador
 - Características a ser probadas
 - Refinamiento de métodos
 - Identificación de las pruebas
 - Criterios de paso/fallo de características
 - Secuencia dirigida por riesgo y prioridad

Casos de Prueba

Cuentan con las siguientes secciones:

- Identificador
- Elementos de prueba
- Especificaciones de entrada
- Especificaciones de salida
- Necesidades ambientales
- Requisitos de procedimientos especiales
- Dependencias entre casos

Los CP varían en esfuerzo, duración, número de condiciones de pruebas cubiertas

Procedimientos de Prueba

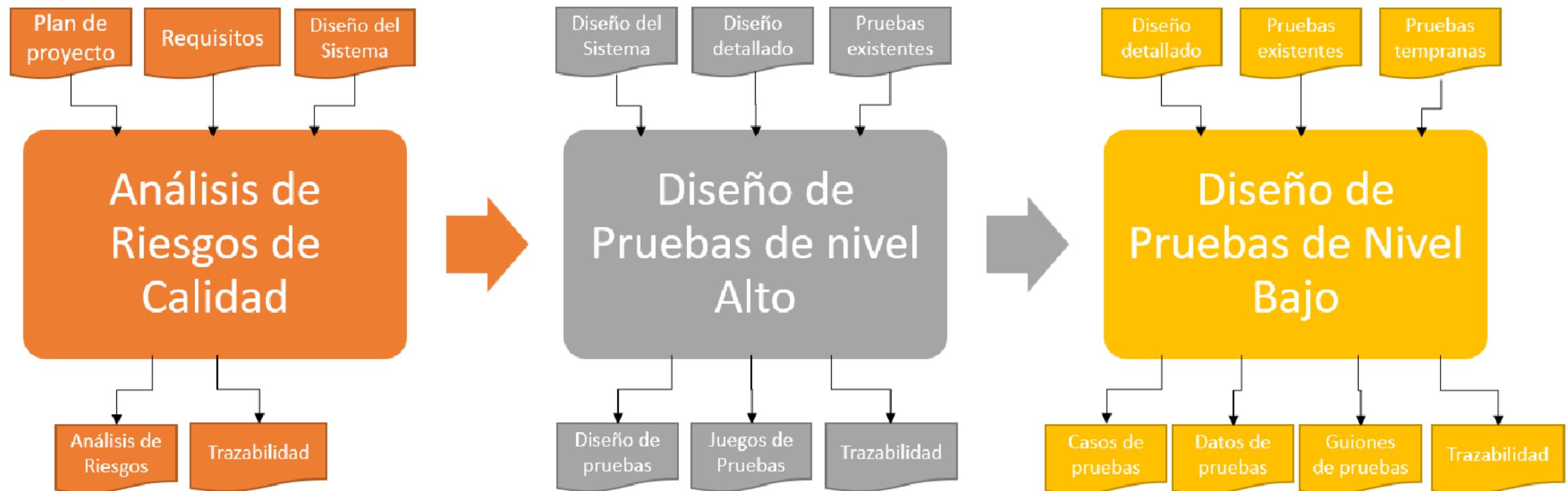
Describen como ejecutar uno o más Casos de Prueba, tiene las secciones:

- Identificador
- Propósito
- Requisitos especiales
- Pasos del procedimiento

Se encuentran embebidos en los casos de prueba.

También llamado guión de pruebas, puede ser manual o automatizado.

Fases del desarrollo de pruebas



Riesgos de producto o calidad

Riesgos

Posibilidad de un resultado negativo o indeseable.

Probabilidad de que llegue a ser un resultado

- $>0, < 1$ en el futuro
- 0 o 1 en el pasado

Riesgos de producto o calidad

Posibilidad de que el sistema falle para satisfacer a los clientes, usuarios u otros interesados.

Pruebas basadas en Riesgos

Reducen riesgos de calidad a lo largo de todo el proyecto
Guían el proceso de pruebas
Involucra a interesados del negocio a través del proyecto
Parte de la gestión de riesgos

¿Cómo analizar los riesgos de Calidad?



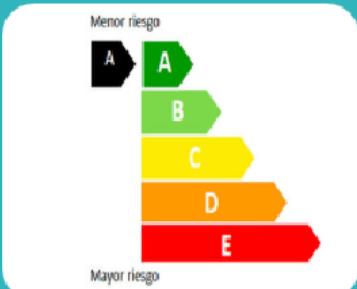
INFORMAL	SEMIFORMAL (ISO 9126)	ANÁLISIS DEL MODO DE FALLO Y EFECTO (AMFE)
Empezar categorizando riesgos de calidad	Empezar considerando las Seis Características de la Calidad	Empezar categorizando características o subsistemas (FMEA) (<i>Failure Mode and Effect Analysis</i>)
Agrupaciones clásicas: Funcionalidad, estados y transacciones, calidad de datos, capacidad y volumen, manejo de errores y recuperación, rendimiento, estándares y localización, usabilidad, etc.	Funcionalidad, Fiabilidad, Usabilidad, Eficiencia, Mantenibilidad, Portabilidad. (FFUEMP) Descomponerlos en características clave del sistema.	Los interesados clave listan los modos de falla posible, predicen sus efectos en el sistema, en los usuarios, en la sociedad, etc. Asignan severidad, prioridad, probabilidad, luego calculan el número de prioridad de riesgo (NPR)
Establecer prioridad de pruebas en función a cada riesgo con los interesados clave	Establecer prioridad de pruebas por subcaracterística con los interesados clave	Interesados usan el NPR para guiar la amplitud y profundidad de las pruebas

Pautas para el Análisis de Riesgos



Utilice un equipo multifuncional

- Lluvia de ideas



Identifique ítems de riesgo

- Asigne niveles de riesgo
- Sepárelos por niveles si es necesario



Considere riesgos técnicos y de negocio

- R. Técnico: Probabilidad del problema
- R. Negocio: Probabilidad del uso

Cobertura de casos de prueba

- Mide o correlación las pruebas con áreas de interés
- Usada como forma de medir las prueba
- Tipos prácticos
 - Especificaciones de requisitos
 - Especificaciones de diseño
 - Áreas funcionales
 - Riesgos de calidad
 - Configuraciones
- De uso común para asegurar la cobertura rigurosa de pruebas

Requirement Identifiers	Reqs Tested	REQ1	REQ1	REQ1	REQ1	REQ1	REQ1	REQ1	REQ1	REQ1	REQ1	REQ1	REQ1
		UC 1.1	UC 1.2	UC 1.3	UC 2.1	UC 2.2	UC 2.3.1	UC 2.3.2	UC 2.3.3	UC 2.4	UC 3.1		
Test Cases	321	3	2	3	1	1	1	1	1	1	1	1	2
Tested Implicitly	77												
1.1.1	1	x											
1.1.2	2		x	x									
1.1.3	2	x											
1.1.4	1				x								
1.1.5	2	x											
1.1.6	1		x										
1.1.7	1			x									
1.2.1	2					x		x		x			
1.2.2	2						x			x			

Categoría de Técnicas de Prueba

**Basadas en la especificación
(caja negra)**

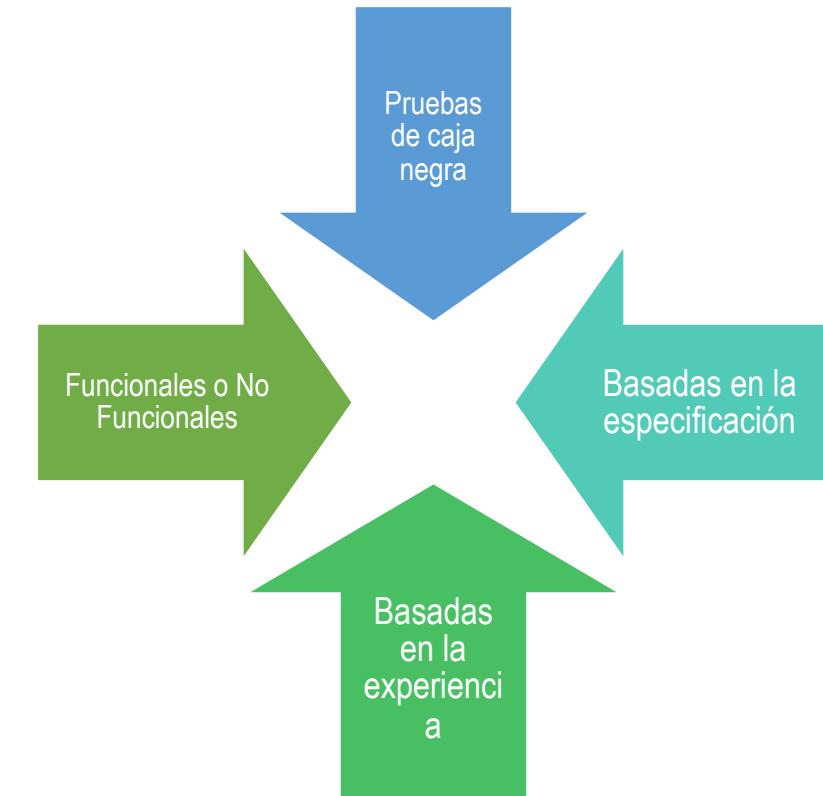
- Realiza análisis condiciones de prueba, casos de prueba y datos que se deducen de una base de prueba.
- Buscan defectos de la manera en que el sistema de comporta
 - Los Casos de prueba son derivados de modelos que especifican el problema, el software o sus componentes.
 - La cobertura se mide en función de los elementos probados y la técnica aplicada a la base de prueba.

**Basadas en la estructura
(caja blanca)**

- Crean pruebas por medio del análisis de estructura del componente o sistema (código fuente, arquitectura, diseño detallado u otros)
- Buscan defectos de la manera en que el sistema es construido
 - Las especificaciones se utilizan como una fuente adicional de información para determinar el resultado esperado de los casos de prueba.
 - La cobertura se mide en base a los elementos probados.

Basadas en la experiencia

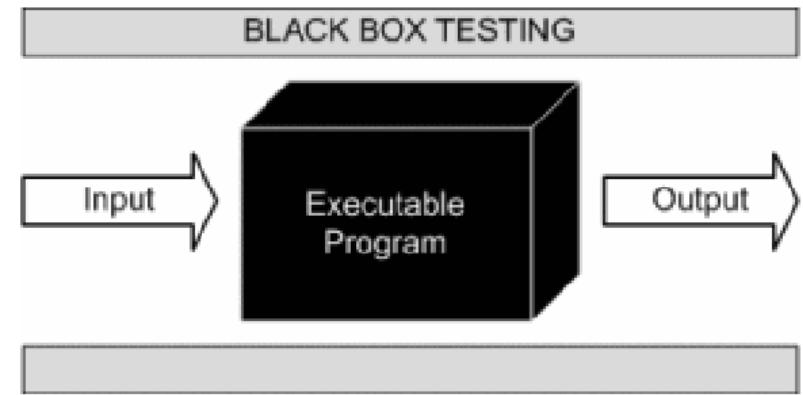
- Crean pruebas basadas en la comprensión del sistema, la experiencia pasada, y las predicciones bien informadas acerca de los defectos.
- Busca defectos en los lugares en que otros sistemas tienen defectos.



Ejemplos de técnicas por categoría

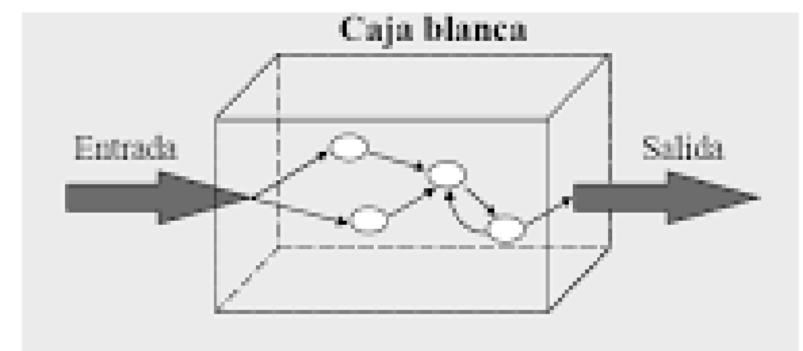
Pruebas de caja negra:

- Partición de equivalencias y análisis de valores límite.
- Diagramas de transición de estados.
- Tablas de decisión.



Pruebas de caja blanca:

- Cobertura de sentencias
- Cobertura de ramas o cobertura de decisión
- Cobertura de condición, cobertura de caminos



Pruebas basadas en la experiencia:

- Ataques
- Listas de comprobación
- Exploratorio

Técnicas basadas en la especificación o Caja Negra



Particionamiento de equivalencias

Dividir las entradas, salidas, comportamientos.

Definir al menos un Caso de Prueba en cada partición



Análisis de valores límite

Refina la partición de equivalencias seleccionando los bordes o puntos finales de cada partición de pruebas

- Busca defectos en el código que se encarga de cada equivalencia
- Los valores límite pertenecen a la clase de equivalencia y buscan defectos en las definiciones de los bordes

Los límites no funcionales (capacidad, volumen, etc.) también pueden ser usados para pruebas no funcionales

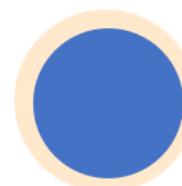


Tablas de decisiones

Especificación compacta de las reglas de negocio

Pueden mostrarse como diagramas o tablas

Las tablas pueden conducir a Casos de Prueba instantáneos



Modelos de estados finitos

Comprende los estados que el sistema tiene, incluyendo los iniciales y finales.

Identifica transiciones, eventos, condiciones y acciones en cada estado

Para cada evento y condición verifican la acción

Partición de equivalencias

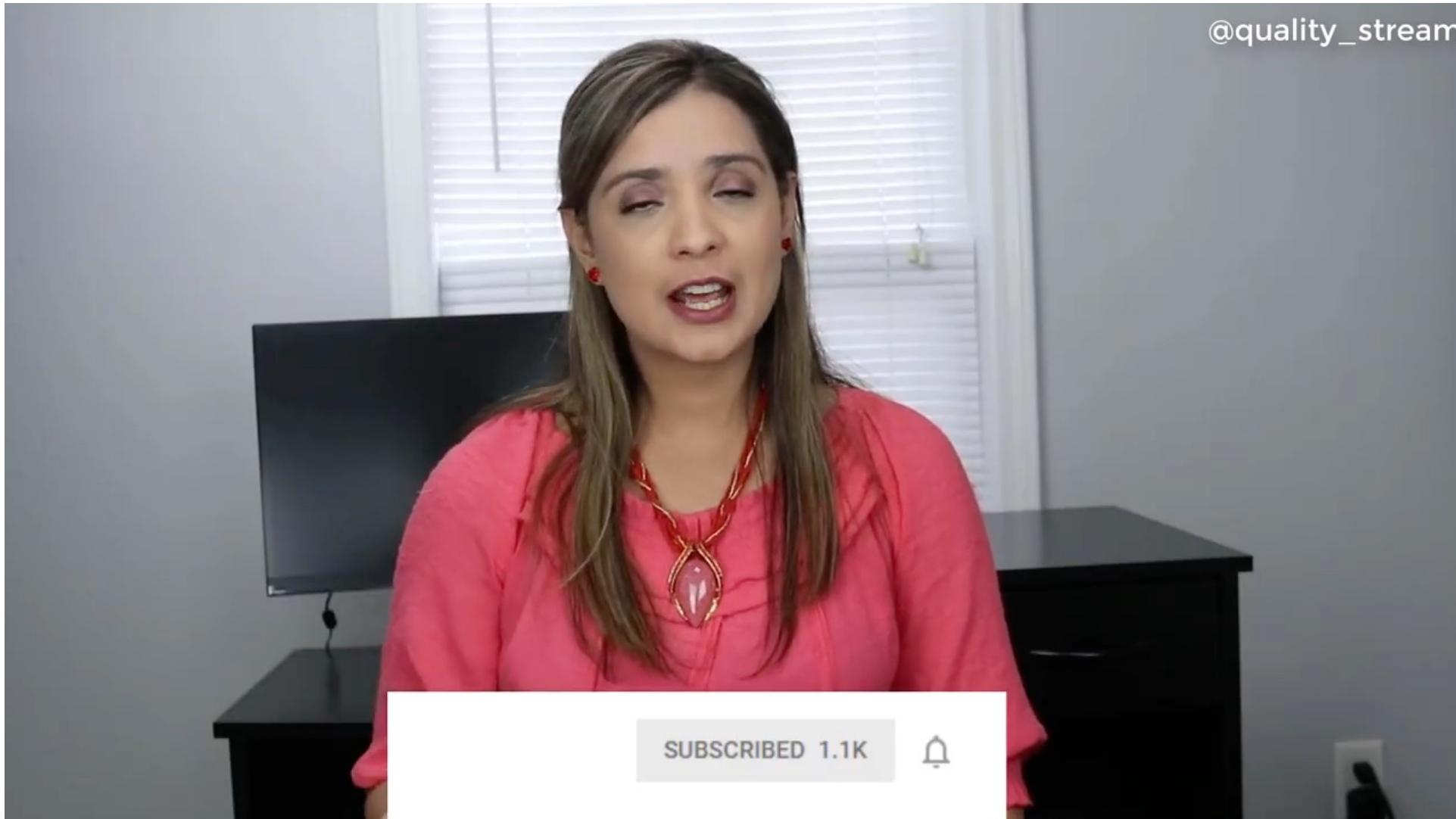


- Se espera que todos los miembros de una partición dada sean procesados de la misma manera.
 - Cualquier partición se puede dividir en subparticiones si fuera necesario.
 - Cada valor debe pertenecer a una y sólo a una partición de equivalencia.

Aplicación bancaria en la que el operador debe proporcionar un código, un nombre y una operación.

Condición de Entrada	Clases Válidas	Clases Inválidas
Código de Área # de 3 dígitos que no empieza con 0 ni 1:	1) $200 \leq \text{código} \leq 999$	2) Código < 200. 3) Código > 999. 4) No es número.
Nombre Para identificar la operación	5) Seis caracteres.	6) Menos de 6 caracteres. 7) Más de 6 caracteres.
Orden Una de las Siguientes	8) "Cheque" 9) "Depósito" 10) "Pago factura" 11) "Retiro de fondos"	12) Ninguna orden válida

Partición de equivalencias



Análisis de Valores Límite

- También se conoce como Análisis de Valores Frontera.
- Solo se puede utilizar cuando la partición está ordenada, y consiste en datos numéricos o secuenciales.
- El comportamiento en las fronteras de las particiones de equivalencia es más probable que sea incorrecto que el comportamiento dentro de las particiones.
- Se puede aplicar en todos los niveles de prueba

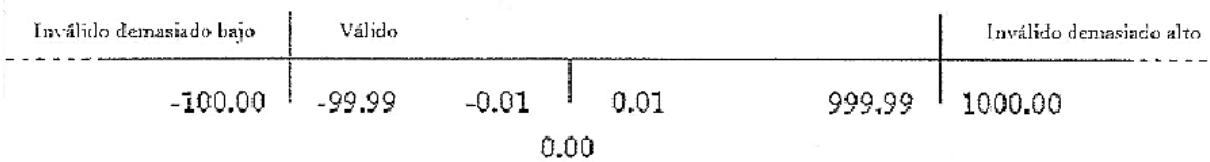
Entero

- “¿Cuantos ítems le gustaría pedir?”



Número Real

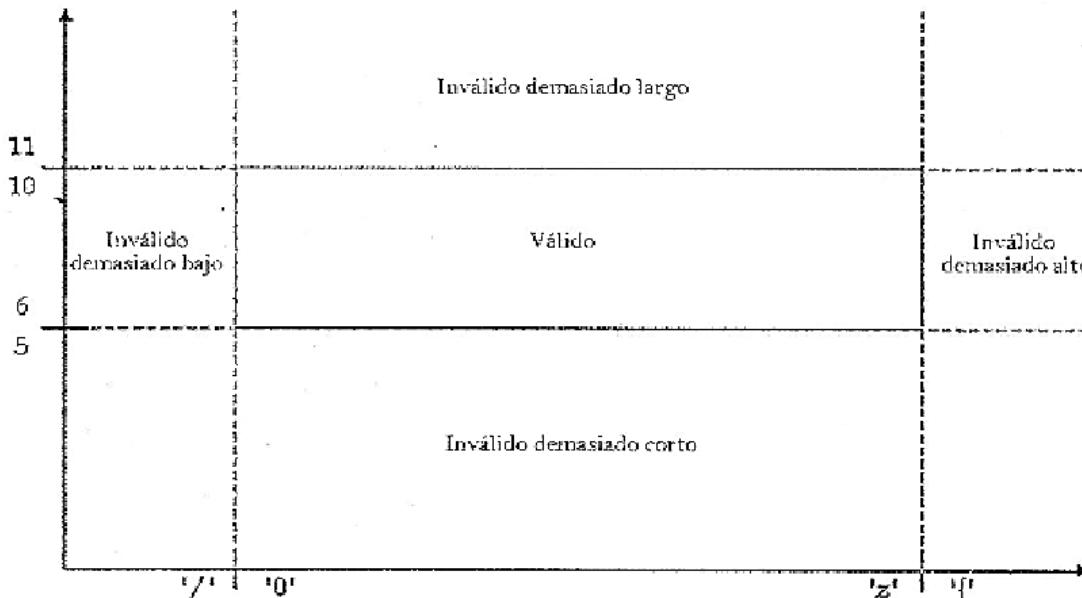
- “¿Cual es la temperatura promedio en Duluth?”



Análisis de Valores Límite

Carácter y Cadena de Texto

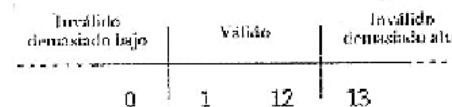
- “Clave (6-10 carácter alfanumerico)”



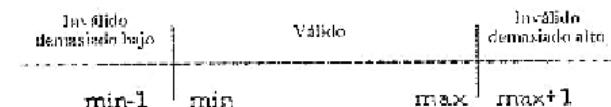
Fecha

- “Ingrese la fecha de salida para su vuelo”

DD/MM/YY



El número máximo de días depende del mes en once casos, y el año en un caso.

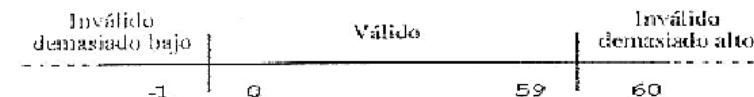
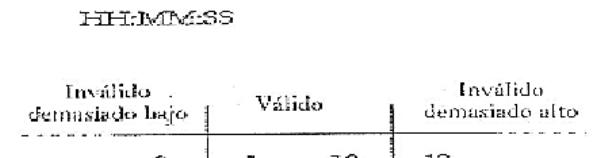


El año mínimo y máximo depende de la aplicación en muchos casos.

Análisis de Valores Límite

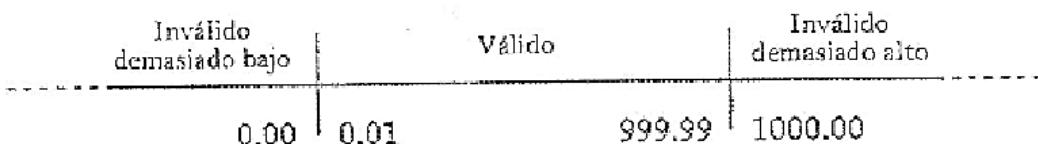
Hora

- “Ingrese la hora de salida para su vuelo”



Moneda

- “Ingrese un precio de oferta(menos de \$1000):”



Técnica:	Análisis de valor límite
Proceso:	Adición de Beneficiario hijo a Afiliado de EPS
Variáble:	Edad de Beneficiario
Valores:	[0 - 18]
Casos:	0 años
	1 año
	5 años
	17 años
	18 años

Análisis de Valores Límite



Análisis de Valores Límite



ISTQB®
Certified Tester
Foundation Level

Tablas de decisiones

- Son una buena manera de documentar reglas de negocio complejas que un sistema debe implementar.
- Al crear tablas de decisión, el probador identifica las condiciones (a menudo entradas) y las acciones resultantes (a menudo salidas) del sistema.

Tabla de Decisiones de un Cajero Automático

Condición	1	2	3	4	5
Tarjeta Válida	N	Y	Y	Y	Y
PIN Válido	-	N	N	Y	Y
PIN Inválido=3	-	N	Y	N	N
Balance OK	-	-	-	N	Y
<hr/>					
Acción					
Rechazar Tarjeta	Y	N	N	N	N
Reingresar PIN	N	Y	N	N	N
Guardar Tarjeta	N	N	Y	N	N
Reingresar Pedido	N	N	N	Y	N
Dispensar efectivo	N	N	N	N	Y

Esta tabla de decisiones muestra la lógica de negocio de un cajero automático. Observe que los guiones “-” indican condiciones que no son alcanzadas como parte de esta regla. Las reglas son mutuamente exclusivas, en que sólo una regla puede aplicar en cualquier momento en el tiempo.

Note que la capa de lógica de negocios está usualmente bajo la capa de interfaz de usuario, de modo que en este punto la comprobación de la sanidad básica de las entradas deberían haber sido realizadas.

Tablas de decisiones

- Una tabla de decisión completa tiene suficientes columnas para cubrir cada combinación de condiciones.
- La tabla se puede colapsar borrando las columnas que contienen combinaciones de condiciones Imposibles.
- La cobertura estándar mínima habitual para la prueba de tabla de decisión es tener al menos un caso de prueba por regla de decisión en la tabla

La Tabla de Decisiones de un Sistema de la Policía

Condición	1	2	3	4	5	6	7	8
Licencia OK	No	-	-	-	-	-	-	-
Autorización	-	Si	-	-	-	-	-	-
Registro OK	-	-	No	-	-	-	-	-
Vehículo OK	-	-	-	No	-	-	-	-
Exceso de velocidad	-	-	-	-	1-10	11-20	21-25	>25
Acción								
Arresto	Si	Si	-	-	-	-	-	Si
Ticket de infracción	-	-	Si	Si	-	-	-	-
Advertencia	-	-	-	-	Si	-	-	-
Multa	+250	+250	+25	+25	+0	+75	+150	+250

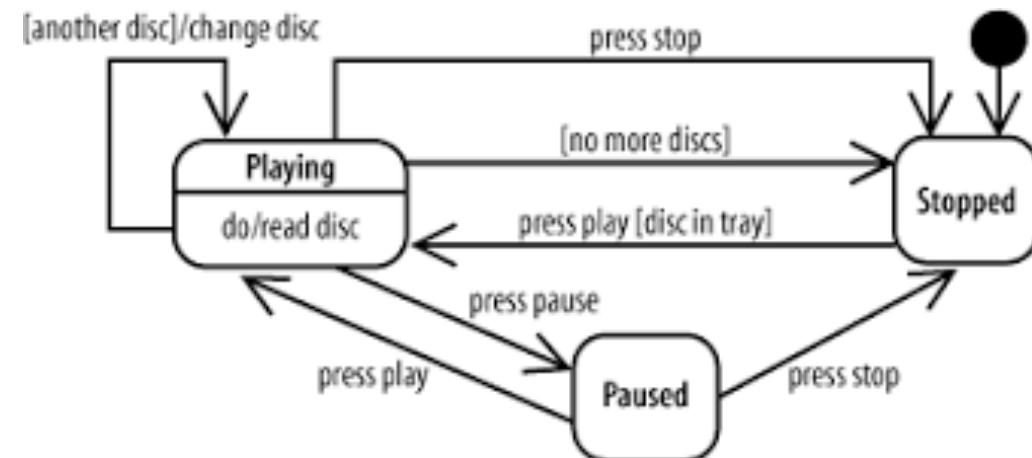
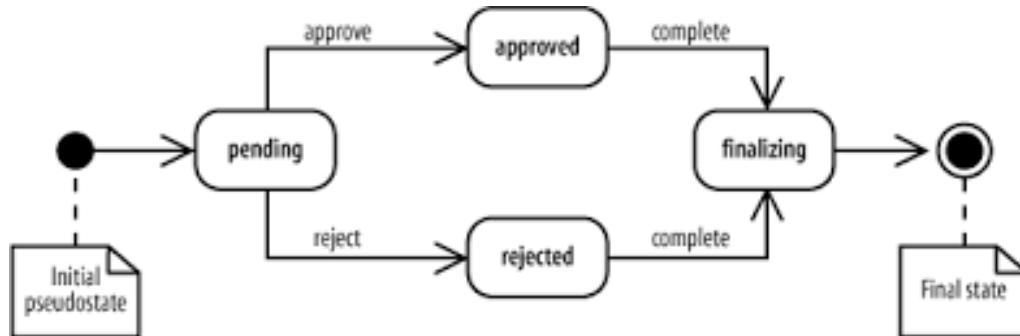
Tablas de decisiones



Modelos de estado finito

- Muestra todas las transiciones válidas y las transiciones potencialmente inválidas entre estados, así como los eventos, las condiciones de guarda y las acciones resultantes para las transiciones válidas.*
- Los diagramas de transición de estado, normalmente, sólo muestran las transiciones válidas y excluyen las transiciones no válidas.*

Transición de estados

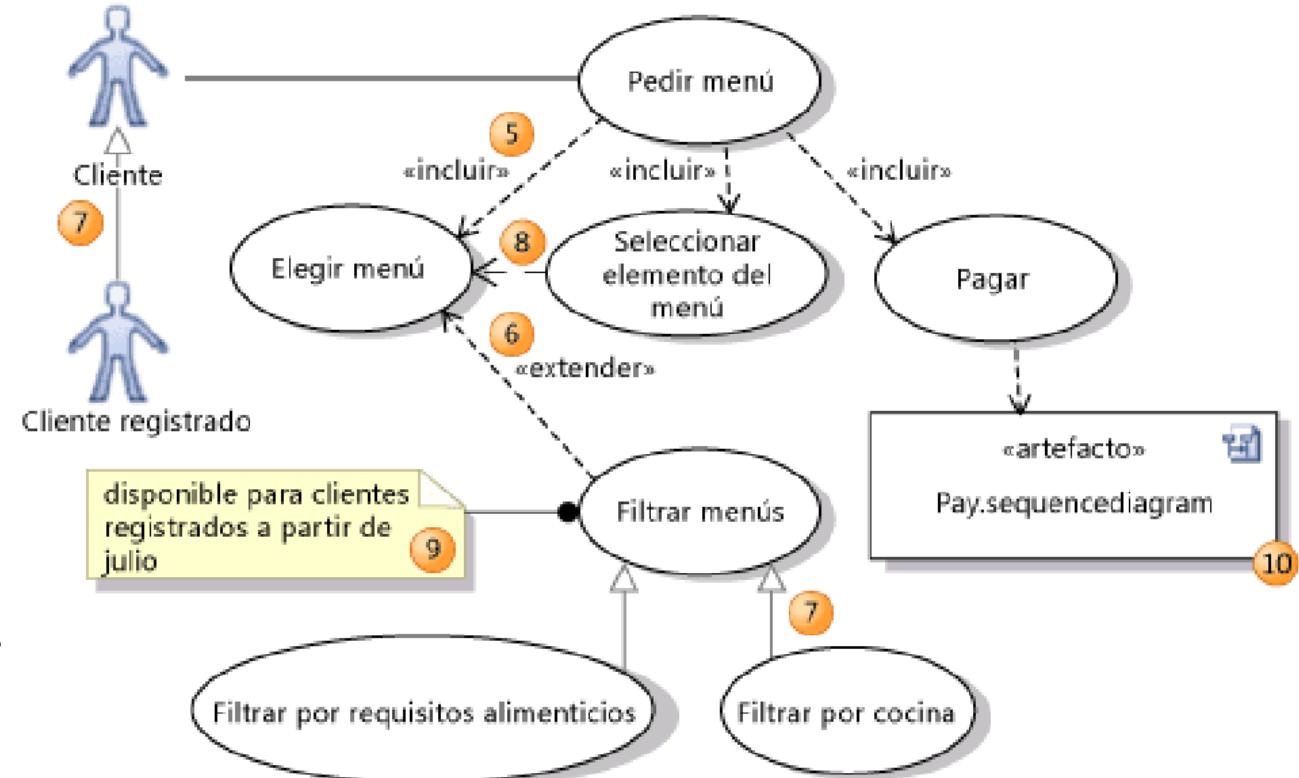


Modelos de estado finito



Prueba de caso de Uso

- Los Test Cases se pueden obtener a partir del diseño de las interacciones con elementos software (Use Cases).
- Los casos de uso están asociados con actores y sujetos.
- Un caso de uso puede describirse mediante interacciones y actividades, así como mediante precondiciones, pos condiciones y lenguaje natural cuando resulte adecuado.



Técnicas basadas en la estructura Caja Blanca

Cobertura de Código
Usada para describir el grado en que el código fuente de un programa se ha probado por medio de un conjunto de pruebas.

- Niveles de cobertura:
- C. De Sentencia / Línea
 - C. De Rama
 - C. De condición
 - C. De condición múltiple
 - C. De condición modificada
 - C. De Bucle

Prueba y Cobertura de decisión
los casos de prueba siguen los flujos de control que se producen desde un punto de decisión.

La cobertura se mide como el número de resultados de decisión ejecutados por las pruebas dividido por el número total de resultados de decisión en el objeto de prueba, normalmente expresado como un porcentaje.

La cobertura de sentencia ayuda a encontrar defectos en el código que no fueron practicados por otras pruebas.

La cobertura de decisión ayuda a encontrar defectos en el código donde otras pruebas no han tenido ambos resultados, verdadero y falso.

Lograr una cobertura del 100% de decisión garantiza una cobertura del 100% de sentencia, pero no al revés.

Técnicas basadas en la estructura Caja Blanca

Cobertura de Código

Usada para describir el grado en que el código fuente de un programa se ha probado por medio de un conjunto de pruebas.

Niveles de cobertura:

- C. De Sentencia / Línea
- C. De Rama
- C. De condición
- C. De condición múltiple
- C. De condición modificada
- C. De Bucle

Prueba y Cobertura de decisión

- los casos de prueba siguen los flujos de control que se producen desde un punto de decisión.
- La cobertura se mide como el número de resultados de decisión ejecutados por las pruebas dividido por el número total de resultados de decisión en el objeto de prueba, normalmente expresado como un porcentaje.

Complejidad Ciclomática de McCabe

Mide la complejidad del flujo de control

- Puede representarse gráficamente por nodos y aristas.
- Nodos representan entradas, salidas, decisiones.
- Aristas representan sentencias sin ramas.

Los nodos de alta complejidad son inherentemente defectuosos

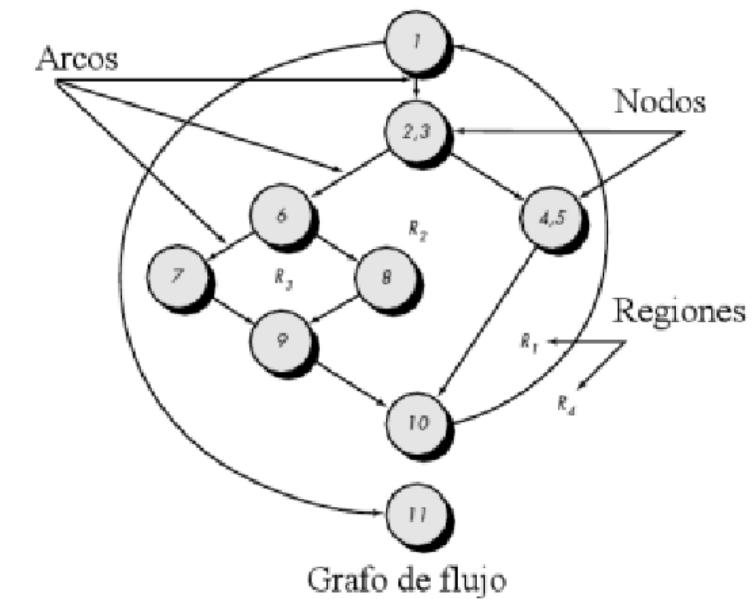
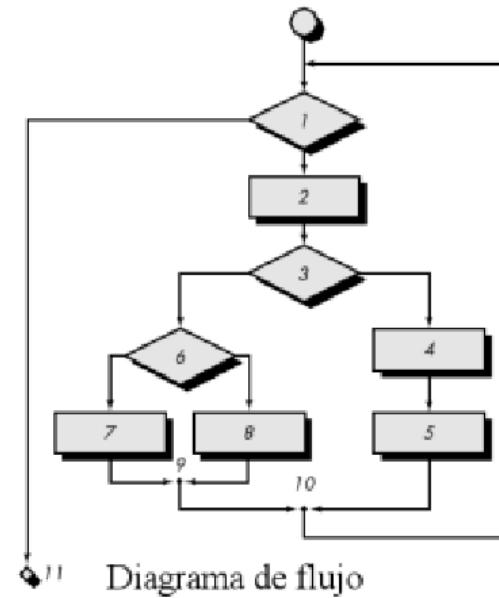
El número de caminos básicos es igual al número de pruebas básicas para cubrir la gráfica

Prueba de ruta básica Complejidad ciclomática

Se derivan las rutas independientes ($V(G) = 4$)
1, 11
1, 2, 3, 4, 5, 10, 1, 11
1, 2, 3, 6, 8, 9, 10, 1, 11
1, 2, 3, 6, 7, 9, 10, 1, 11

Se derivan casos de prueba para ejercitar cada ruta

Cálculo de la complejidad ciclomática
 $V(G) = R = 4$ $V(G) = E - N + 2 = 11 - 9 + 2 = 4$



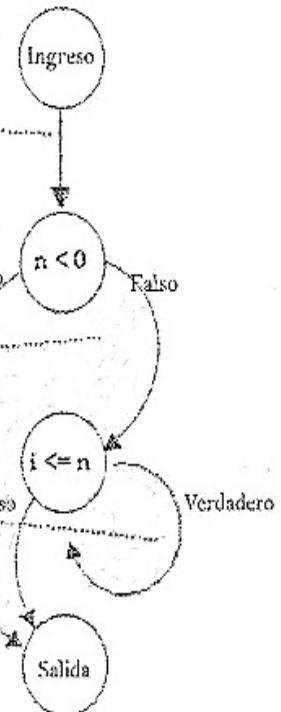
Ejemplos de Complejidad ciclomática

La Complejidad Ciclomática para el Factorial

Programa

```
main()
{
    int i, n, f;
    printf("n = ");
    scanf("%d", &n);
    if (n < 0) {
        printf("Invalid: %d\n", n);
        n = -1;
    } else {
        f = 1;
        for (i = 1; i <= n; i++) {
            f *= i;
        }
        printf("%d! = %d.\n", n, f);
    }
    return n;
}
```

Diagrama de Flujo



Complejidad Ciclomática

$$C = \#R + 1 = \\ 2 + 1 = 3$$

or

$$C = \#E - \#N + 2 = \\ 5 - 4 + 2 = 3$$

Definiciones

C = Complejidad ciclomática

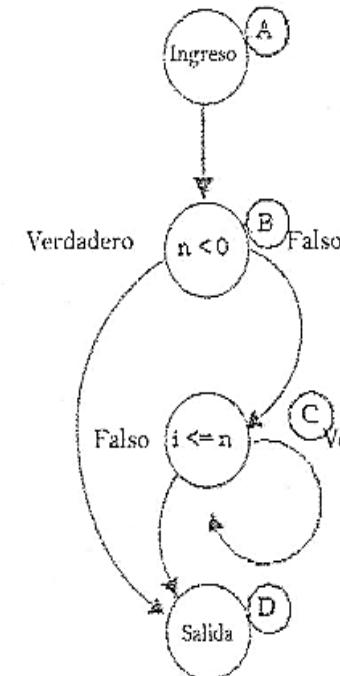
R = Región encerrada

E = Arista (flecha)

N = Nodo (burbuja)

Caminos y Pruebas Básicas

Diagrama de Flujo



Caminos Básicos

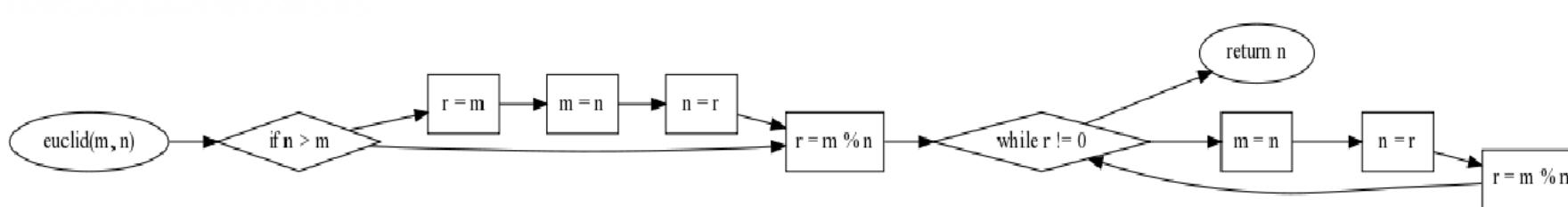
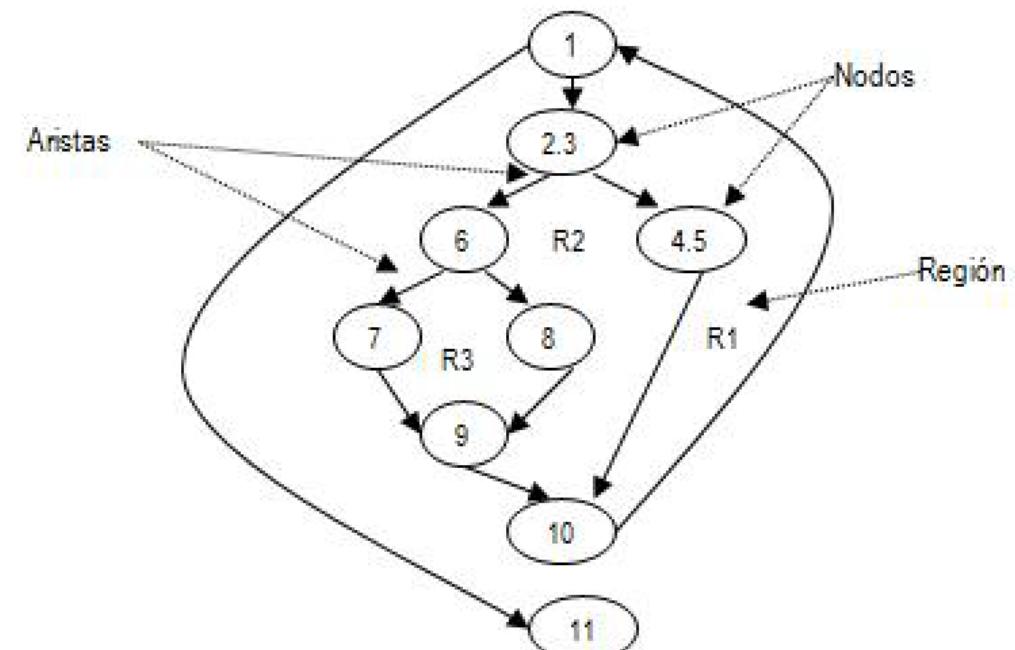
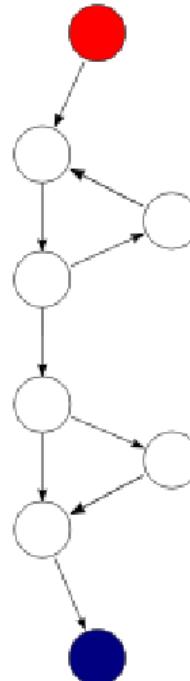
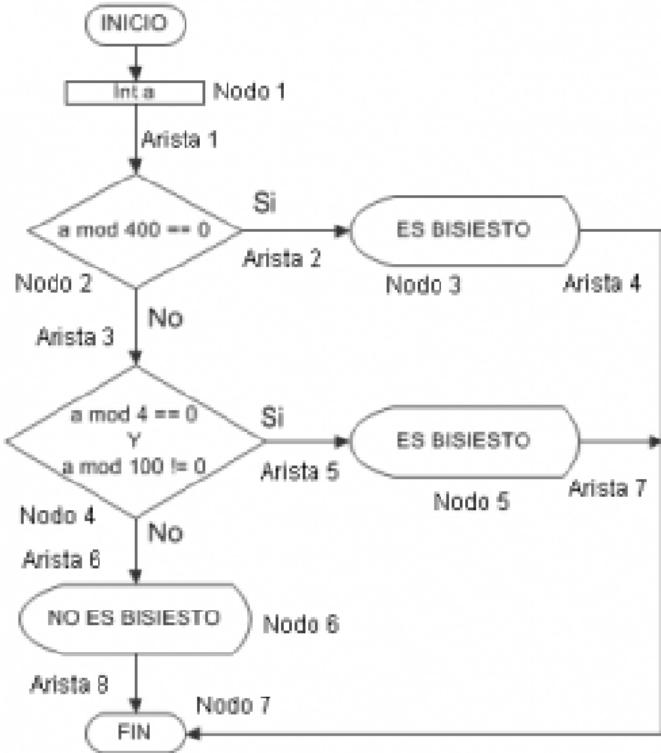
1. ABD
2. ABCD
3. ABCCD

Pruebas Básicas

Entrada	Esperado
1. -1	Inválido: -1
2. 0	0! = 1.
3. 1	1! = 1.

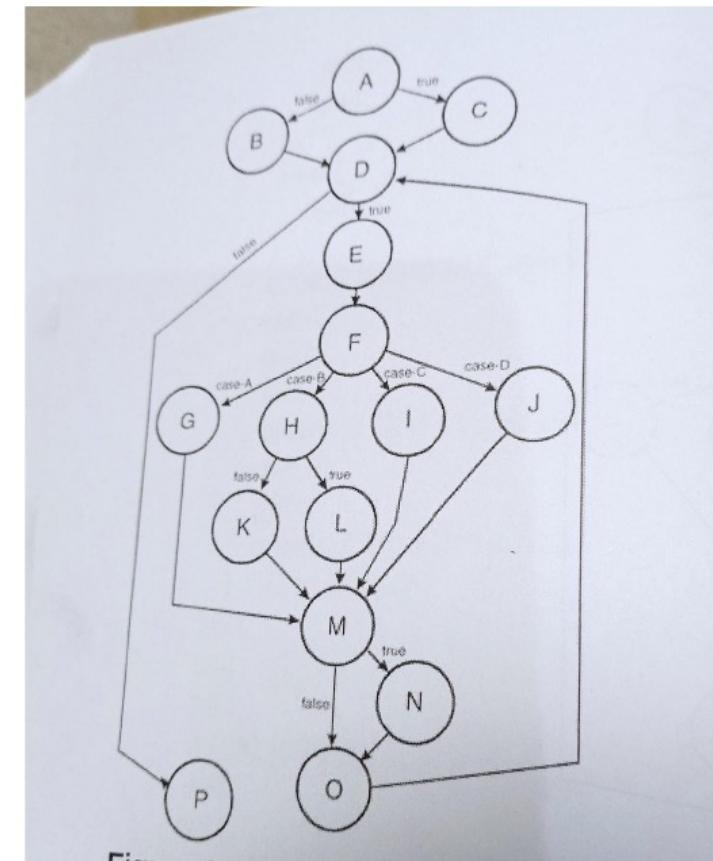
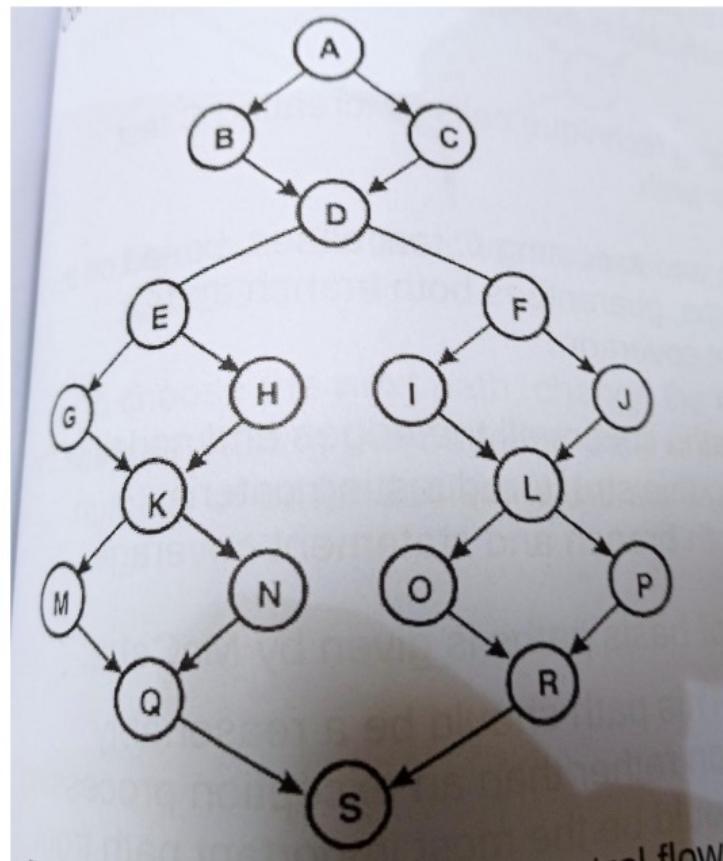
Ejemplos de Complejidad ciclomática

$$v(G) = e - n + 2$$



Ejemplos de Complejidad ciclomática

$$v(G) = e - n + 2$$



Técnicas basadas en la Experiencia

Técnicas basadas en Experiencia

Se basan en:

Habilidad e intuición
Experiencia con aplicaciones similares
Experiencia con tecnologías similares

Más que prediseñadas, son creadas durante la ejecución de pruebas, es decir la estrategia es dinámica

Son encajonadas en el tiempo (periodos cortos enfocadas en condiciones)

Ejemplos:

Predicción de errores, cacería de defectos, “rompimiento del software”, y pruebas exploratorias.

VENTAJAS

Efectivas para encontrar defectos

Resisten la paradoja del pesticida debido a la alta varianza

Eficiente (Mantenimiento liviano de registros)

Buena comprobación de pruebas separadas

Divertidas y creativas

DESVENTAJAS

Cobertura incompleta, especialmente bajo presión

Difícil de estimar

No hay prevención de defectos

Informes extensos y las discusiones no escalan a grandes equipos

No todos los probadores tienen el nivel necesario de habilidad, experiencia

Técnicas basadas en la Experiencia

Pueden ser dirigidos por



- Listas de comprobación
- Taxonomía de defectos
- Lista de ataques
- Método de cacería de defectos
- Conjunto de test charters

Caso de Estudio de Pruebas Exploratorias

Personal	7 Técnicos	3 Ingenieros + 1 Jefe
Experiencia	< 10 años en total	> 20 años en total
Tipo de Prueba	Guiones Precisos	Exploratorias en base a Charters
Prueba Hrs./Día	42	6
Defectos Encontrados	928 (78%)	261 (22%)
Efectividad	22 Defec./Hr.	44 Defec./Hr.

Este caso de estudio muestra de que las pruebas exploratorias son alrededor de dos veces más efectivas al encontrar defectos en base a hora por hora. Es significativamente más eficiente, porque las pruebas de guiones requirieron esfuerzo extensivo para ser creadas. Sin embargo, ¿Hasta qué punto es importante el nivel relativo de la experiencia? ¿Cuál es el valor de probar más allá de encontrar defectos? ¿Qué cubrieron las pruebas exploratorias? ¿Cuál es el valor de reutilización de los guiones?

Selección de Técnicas de pruebas

Factores

- El tipo de sistema
- Los estándares reguladores
- Requisitos del cliente o del contrato
- El Nivel y el tipo de riesgo
- Los objetivos de las pruebas
- La documentación disponible
- El conocimiento de los probadores
- El tiempo y presupuesto
- El ciclo de vida de desarrollo
- Experiencias previas en los tipos de defectos encontrados
- Otros

Detalle y precisión del CP

- Pruebas precisas requieren probadores con menos destreza, pero no muy flexibles
- Pruebas imprecisas pueden cubrir más condiciones, pero no son tan reproducibles
- Pruebas precisas proveen criterios de prueba transparentes pero son difícil y caras de mantener
- Pruebas imprecisas son rápidas de escribir pero la cobertura es difícil de definir o medir

“Exploratorias puras”

“Exploratorias en base a contratos”



“Estilo IEE829”

“De puros guiones”

Taller

- Introducción y Objetivos
- I. Fundamentos de Pruebas
- II. Pruebas a través del Ciclo de Vida de Software
- III. Técnicas Estáticas
- IV. Técnicas de Prueba
- V. Gestión de Pruebas
- VI. Soporte de Herramientas para Pruebas

CAPITULO V

Gestión de Pruebas



Organización del proceso de Pruebas



Planificación y Estimación



Seguimiento y Control



Gestión de la configuración



Riesgo y proceso de pruebas



Gestión de incidencias

Organización de procesos de pruebas



Trabajo de un equipo de pruebas

- Control de pruebas / Calidad
 - Gestión de riesgos
 - Evaluación de la calidad
- Aseguramiento de la Calidad
 - Gestión de pruebas
 - Gestión del control de la calidad (QC)
 - Aseguramiento de la calidad a través del proceso



Pruebas e independencia

- Para proyectos grandes, complejos y críticos para la seguridad, normalmente lo mejor es contar con varios niveles de pruebas y poner alguno o todos los niveles a cargo de probadores independientes
- El personal de desarrollo puede participar en las pruebas, especialmente en niveles más bajos
- Los probadores independientes pueden tener potestad para exigir y definir procesos y reglas de prueba

Prueba Independiente

Los probadores pueden ser más efectivos para encontrar defectos debido a las diferencias entre los sesgos asociados al conocimiento del autor y del probador, sin embargo, la independencia no es un sustituto de la familiaridad.

Para la mayoría de los tipos de proyectos, generalmente es mejor que tengan diferentes niveles de prueba, con algunos de estos niveles tratados por probadores independientes.

La forma en que se implementa la independencia de la prueba varía dependiendo del modelo de ciclo de vida de desarrollo de software.

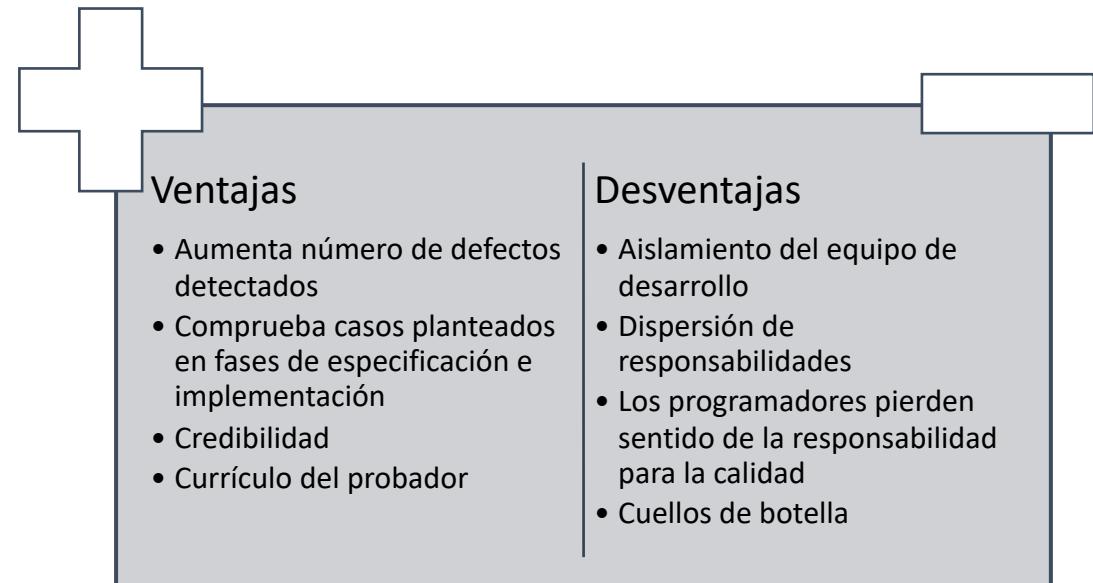
En algunas organizaciones que utilizan métodos Ágiles, estos probadores también pueden ser considerados parte de un equipo de prueba independiente más grande.

los propietarios de producto pueden realizar la prueba de aceptación para validar las historias de usuario al final de cada iteración

Grado de independencia



“Sino hay probadores independientes, entonces los desarrolladores prueban su propio código”



Organización del proceso de pruebas

Las tareas de prueba pueden realizarlas personas con una función de pruebas específica o por personas con otras funciones añadidas: jefes de proyecto, jefes de calidad, desarrolladores, expertos de negocio, etc...

- *Líder de pruebas:*
 - *Jefe de proyecto*
 - *Jefe de desarrollo*
 - *Jefe específico de pruebas*
- *Probador:*
 - *Desarrollador*
 - *Analista*
 - *Experto*

Organización del proceso de pruebas

Líder de pruebas

- Desarrollar políticas o estrategias de pruebas
- Coordinar los PDP
- Redactar y revisar PDPs
- Aportar perspectiva
- Planificar
- Iniciar proceso
- Adaptar planificación
- Gestionar
- Establecer métricas
- Decidir automatización
- Seleccionar herramientas
- Implementar entorno
- Sacar informes

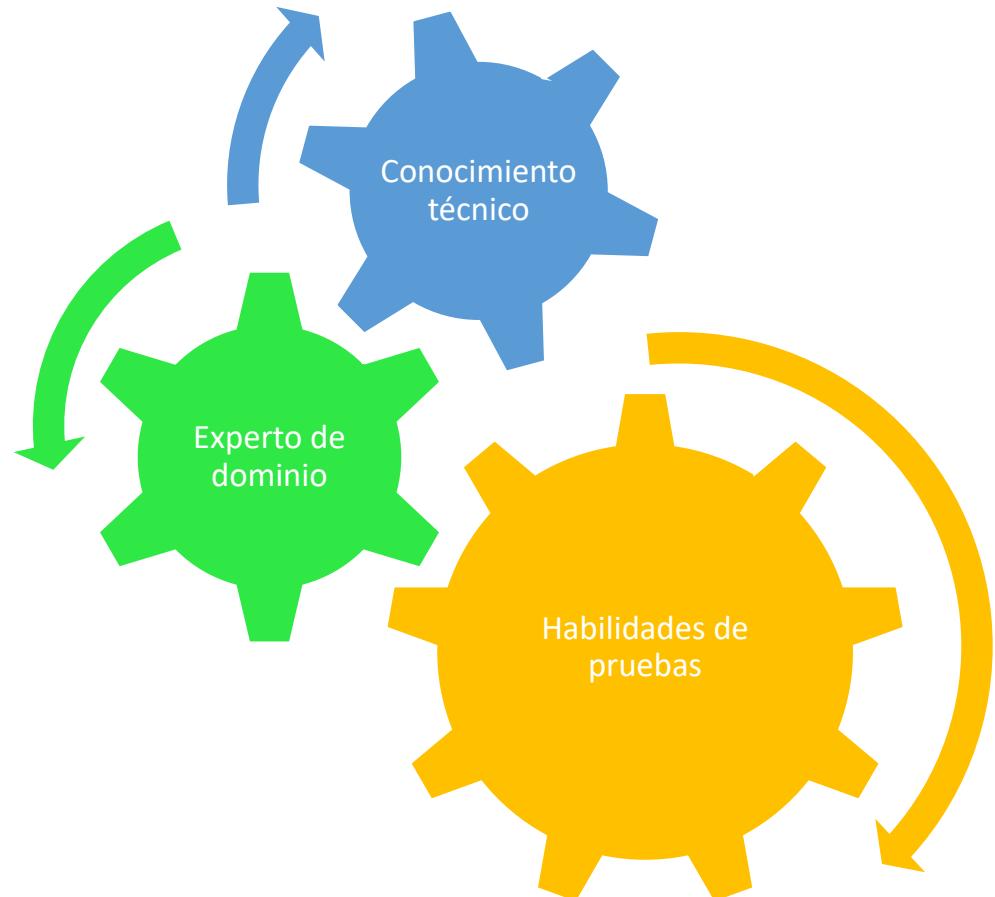
Probador

- Revisar y contribuir a los PDP
- Analizar y Evaluar requisitos, HU, criterios de aceptación, etc.
- Crear casos y especificaciones
- Configurar entorno
- Preparar y obtener datos
- Implementar
- Utilizar herramientas de administración
- Utilizar herramientas de gestión
- Automatizar
- Medir rendimientos
- Revisar

- *Las personas dedicadas al análisis de pruebas, al diseño de pruebas o a la automatización de pruebas pueden variar en cuanto a grado de especialización.*
- *Hay distintos perfiles que pueden adquirir el rol de probador siempre que se mantenga cierto grado de independencia.*
- *En general, los probadores a nivel de componente e integración son los desarrolladores y los probadores a nivel de aceptación son los expertos de negocio y usuarios*

Equilibrio de habilidades

- Los equipos de pruebas deben tener la mezcla correcta de habilidades basadas en tareas y actividades
- ***El experto de dominio: Entiende el comportamiento requerido***
- ***El probador hábil: Conoce los riesgos de la calidad y las técnicas de pruebas***
- ***El Gurú Técnico: Es consciente de las cuestiones técnicas y limitaciones***



Planificación y Estimación del Proceso de Pruebas

● PLANIFICACIÓN DE PRUEBAS

● *Conjunto de decisiones que tienen como objetivo llevar a cabo el desarrollo de las pruebas y llevarlas a buen término.*

● *A nivel de:*

- Desarrollo
- Implementación

- Mantenimiento

- *Tipos de pruebas a planificar:*

- Pruebas de sistema

- Pruebas de implementación

- Pruebas de aceptación

- Afectada por:

- Alcance de las pruebas

- Objetivos

- Riesgos

- Limitaciones

- Criticidad

- Testabilidad

- Recursos

Actividades de planificación de pruebas

- Determinar alcance e identificar riesgos
- Definir enfoque
- Integrar y coordinar
- Decidir qué probar
- Programar actividades, implementación, ejecución y evaluación
- Asignar recursos
- Definir contexto de pruebas
- Seleccionar métricas
- Establecer nivel de detalle de pruebas

Plan de pruebas IEEE829

- Un PDP es un subproyecto que tiene como propósito comunicar a todos los involucrados del proyecto los entregables, las características a ser o no probadas, los criterios de aprobación y fallo, necesidades ambientales, riesgos, etc. Referente a las pruebas de software.
- El esquema de IEEE89 puede adaptarse para el PDP del proyecto o para el PDP maestro de la organización.
- El PDP es influenciado por las políticas de la organización, el alcance, objetivos, riesgos, criticidad, comprobabilidad, y la disponibilidad de recursos.

Esquema IEEE829

- Identificador del PDP
- Introducción
- Ítems de pruebas
- Características a ser probadas
- Características a no ser probadas
- Método (estrategias, organización y alcance)
- Criterios paso/falla
- Criterios de pruebas (Entrada, salida, suspensión y reanudación)
- Entregables de pruebas (informes, diagramas, ..)
- Tareas de pruebas
- Necesidades de entorno
- Responsabilidades
- Asignación de personal y necesidades de capacitación
- Calendario
- Riesgos y contingencias (de producto y proyecto)
- Aprobaciones

Estimación de pruebas

Paso 1: Divide el proyecto entero en pequeñas tareas.

Paso 2: Asigna cada tarea a miembros del equipo.

Paso 3: Estima el tiempo requerido para cada área.

Paso 4: Valida la estimación.

¿Qué Estimar?

Recursos

Tiempos

Habilidades Humanas

Costos

La podemos basar en:

- Métricas
- Expertos

Factores que determinan el esfuerzo:

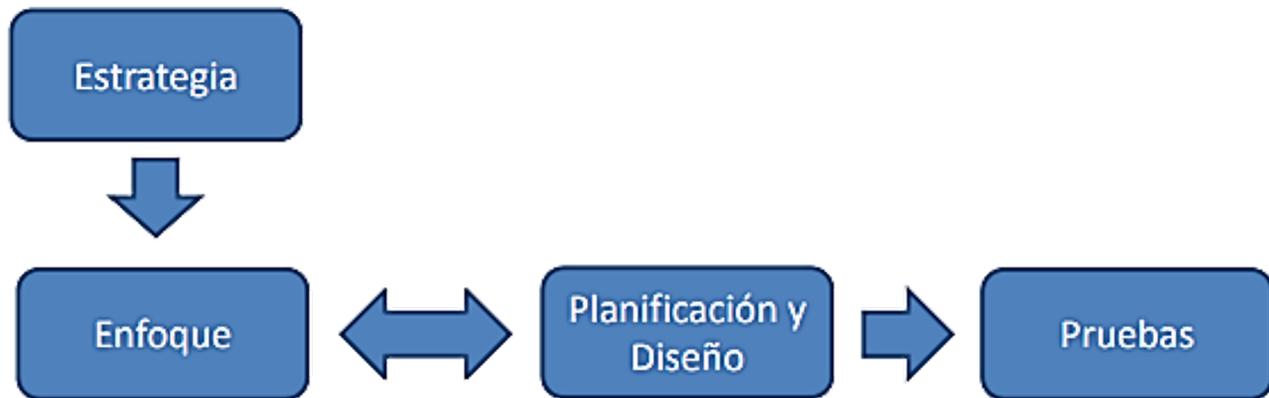
- Características del producto
- Características del proceso
- Resultado de las pruebas

Estrategia y enfoque

- El enfoque de pruebas es la aplicación de la estrategia de pruebas para un proyecto específico
- Se define (y redefine) en los planes y diseño de pruebas
- Constituyen el punto de inicio para planificar el proceso de pruebas

Enfoques:

- **Analítico**
- **Basado en modelos**
- **Metódico**
- **Conforme a Proceso**
- **Dirigida / consultiva**
- **Adversa a la regresión (Orientado a la toma de decisiones)**
- **Reactiva**



Criterios de Entrada

Miden si el sistema está listo para una fase de pruebas

¿Están listos los comprobables y ejecutables?

¿Está listo el laboratorio?

¿Están listos los equipos de trabajo? (Desarrolladores, probadores, etc.)

Se basa en:

- **Entorno**
- **Herramientas**
- **Código**
- **Datos de prueba**

Ejemplo de Criterios de Entrada

La prueba de sistema puede comenzar cuando:

1. Los sistemas de seguimiento de defectos y seguimiento de pruebas estén instalados.
2. Todas las componentes estén formalmente bajo la configuración y el control automatizado de la gestión de versiones.
3. El equipo de operaciones haya configurado el entorno del servidor para la prueba del sistema, incluyendo todas las componentes y subsistemas del hardware meta. El equipo de pruebas tenga el acceso adecuado a estos sistemas.
4. Los equipos de desarrollo hayan terminado todas las características y las correcciones planificadas de los defectos para la versión.
5. Los equipos de desarrollo hayan realizado las pruebas de unidad de todas las características y las correcciones planificadas de los defectos para la versión.
6. Haya menos de 50 defectos abiertos para la versión, que deben ser corregidos según Ventas, Marketing y Servicio al Cliente.
7. Los equipos de desarrollo hayan proporcionado el software al equipo de pruebas 3 días hábiles antes de comenzar la prueba de sistema.
8. El equipo de pruebas haya realizado 3 días de “pruebas de humo” e informe acerca de los resultados en la reunión inicial de la fase prueba de sistema.
9. El equipo de gestión del proyecto haya acordado de continuar en la Reunión acerca de la Entrada a la Fase Prueba de Sistema. Y los siguientes puntos hayan sido resueltos en la reunión:
 - ✓ Si el código está terminado.
 - ✓ Si las pruebas de unidad están terminadas.
 - ✓ Asignación de una fecha meta para la corrección de cualquier defecto conocido que debe ser corregido (no más tarde que una semana después de la Entrada a la Fase Prueba de Sistema).

Criterios de Entrada

- Entre los criterios de entrada habituales se encuentran:
- Disponibilidad de requisitos, historias de usuarios y/o modelos (por ejemplo, cuando se sigue una estrategia de prueba basada en modelos) con capacidad de ser probados.
- Disponibilidad de elementos de prueba que han cumplido los criterios de salida para cualquiera de los niveles de prueba anteriores.
- Disponibilidad del entorno de prueba.
- Disponibilidad de las herramientas de prueba necesarias.
- Disponibilidad de datos de prueba y otros recursos necesarios.

“Definición de Preparado”

Criterios de Continuación

Miden si las pruebas pueden seguir eficientemente y efectivamente

Problemas del entorno de pruebas

Defectos que bloquean las pruebas

La Prueba de Sistema continuará si:

1. Todo el software publicado para el Equipo de Pruebas está acompañado de las Notas de la Versión.
2. Ninguna modificación ha sido realizada en el sistema, ni en el código fuente, ni en los archivos de configuración, o en otras instrucciones de instalación o en los procesos, sin el acompañamiento de un informe de defecto. Si un cambio es realizado sin un informe de defecto, entonces el Jefe de Pruebas abrirá un informe de defecto urgente solicitando información y lo escalará a su jefe.
3. El retraso de los defectos abiertos (“el vacío de la calidad”) sigue siendo menos de 50. Los períodos de cierre diarios y escalonados siguen siendo menos de 14 días (en promedio, los defectos son corregidos dentro de dos ciclos de versión semanales).
4. Las reuniones de revisión de defectos ocurren 2 veces por semana hasta La Salida de la Fase de Prueba de Sistema para gestionar el retraso de los defectos abiertos y los tiempos de cierre de defectos.

Criterios de Salida

Miden si la fase de pruebas puede ser considerada como terminada

Medidas de cobertura de código, funcionalidad o riesgos

Estimaciones de la densidad de defectos o fiabilidad

Los costos

Los riesgos residuales, tales como defectos no corregidos o falta de cobertura

Calendarios o tiempo para llegada al mercado

Ejemplos de Criterios de Salida

La Prueba de Sistema terminará cuando:

1. No hayan ocurrido cambios (de diseño/código/características), excepto para abordar los defectos de La Prueba de Sistema, que ocurrieron en las 3 semanas anteriores.
2. No haya ocurrido pánico, caída, paro, conflicto de recursos, suspensión del procesamiento, terminación inesperada de un proceso, u otro paro de proceso en algún software de servidor o hardware en las últimas 3 semanas .
3. Ninguno de los sistemas cliente se hayan vuelto inoperables debido a una actualización que falló durante la Prueba de Sistema.
4. El Equipo de Pruebas haya ejecutado todas las pruebas planificadas contra el software candidato de disponibilidad general.
5. Los Equipos de Desarrollo hayan resuelto todos los defectos que debían ser corregidos según Ventas, Marketing y Servicio al Cliente.
6. El Equipo de Pruebas haya comprobado que todas las cuestiones en el sistema de seguimiento de defectos estén ya sea cerrados o postergados, y , donde sea apropiado, sean verificados por las pruebas de regresión y confirmación.
7. Las métricas de las pruebas indiquen: la estabilidad y la fiabilidad del producto; el cumplimiento de todas las pruebas planificadas; la cobertura adecuada de los riesgos de calidad crítica.
8. El Equipo de la Gestión del Proyecto acuerde que el producto, como se lo ha definido durante el ciclo final de la Prueba de Sistema, satisfará las expectativas razonables de calidad del cliente.
9. El Equipo de Gestión del Proyecto tenga una Reunión de Salida de la Fase de Prueba de Sistema y esté de acuerdo que hemos terminado la Prueba de Sistema.

Criterios de Salida

- Las pruebas planificadas han sido ejecutadas.
- Se ha logrado un nivel de cobertura definido (por ejemplo, de requisitos, historias de usuario,
- criterios de aceptación, riesgos, código).
- El número de defectos no resueltos se encuentra dentro de un límite acordado.
- El número estimado de defectos restantes es suficientemente bajo.
- Los niveles de fiabilidad, eficiencia de desempeño, usabilidad, seguridad y otras características de calidad relevantes son suficientes.

“Definición de Hecho”

Calendario de Ejecución de Prueba

- Se debe tener en cuenta factores tales como la priorización, las dependencias, las pruebas de confirmación, las pruebas de regresión y la secuencia más eficaz para ejecutar las pruebas.
- Los casos de prueba se ordenarían en función de sus niveles de prioridad, generalmente ejecutando primero los casos de prueba con mayor prioridad. Sin embargo, esta práctica puede no funcionar si los casos de prueba tienen dependencias o si las características que se están probando tienen dependencias.

Factores que influyen en el esfuerzo de prueba

Características del Producto	Características del Proceso de Desarrollo	Características de las Personas	Resultados de la prueba
<ul style="list-style-type: none"> • Los riesgos asociados al producto. • La calidad de la base de prueba. • El tamaño del producto. • La complejidad del dominio del producto. • Los requisitos para las características de calidad (por ejemplo, seguridad, fiabilidad). • El nivel de detalle necesario para la documentación de la prueba. • Requisitos para el cumplimiento de normas legales y reglamentarias. 	<ul style="list-style-type: none"> • La estabilidad y madurez de la organización. • El modelo de desarrollo en uso. • El enfoque de prueba. • Las herramientas utilizadas. • El proceso de prueba. • Presión con respecto al tiempo. 	<ul style="list-style-type: none"> • Las competencias y la experiencia de las personas involucradas, especialmente con proyectos y productos similares (por ejemplo, conocimiento del dominio). • Cohesión y liderazgo del equipo. 	<ul style="list-style-type: none"> • El número y la severidad de los defectos detectados. • La cantidad de reconstrucción necesaria.

Técnicas de estimación de Pruebas

La técnica basada en métricas

- Estimación del esfuerzo de prueba basada en métricas de proyectos similares anteriores o en valores típicos
- Ej. Ágil: los gráficos de trabajo pendiente son ejemplos del enfoque basado en métricas
- Ej. Secuencial: modelos de eliminación de defectos

La técnica basada en expertos

- Estimar el esfuerzo de la prueba basándose en la experiencia de los propietarios de las tareas de prueba o por expertos
- Ej.: Poker de planificación para aproximación basada en expertos
- Ej. Secuencial: técnica de estimación Delphi de Banda (técnica de estimación basada en expertos en la que los grupos de expertos realizan estimaciones basadas en su experiencia)

Seguimiento y Control del Estado de las Pruebas

Objetivo

- Facilitar feedback
- Proporcionar métricas
 - Trabajo casos de prueba
 - Trabajo entorno
 - Ejecución de casos de prueba
 - Defectos
 - Cobertura
 - Confianza
 - Tiempo
 - Coste

Informe de pruebas

- ¿Qué ha pasado?
- Análisis
- Métricas
 - ¿Objetivo cumplido?
 - Enfoque
 - Efectividad
- Resumen

Control de pruebas

- Acciones orientativas o correctiva resultado de las pruebas
- Acciones:
 - Tomar decisiones
 - Establecer prioridades
 - Ajustar calendario
 - ¿Es necesario repetir?

Control de pruebas

- **Acciones guías y correctivas debido a la información de pruebas y métricas.**
 - **Pueden afectar las actividades de prueba u otras actividades del ciclo de vida de software.**

Ejemplos:

- Repriorización de pruebas originadas por riesgos.
- Ajustes del calendario debido a la disponibilidad del entorno de pruebas
- Establecimiento de un criterio de entrada que necesita la repetición de pruebas de las correcciones de defectos por los desarrolladores antes de la integración en una compilación.



Métricas de pruebas

Informe de Pruebas

Resumen / Analizan los resultados de pruebas

- Los eventos clave (ej. Alcance de criterios de salida)
- Análisis de:
 - ... Defectos sobrantes
 - ... Costo / beneficio de más pruebas
 - ... Riesgos pendientes
 - ... Nivel de confianza
- Métricas reunidas para evaluar:
 - La adecuación de los objetivos de pruebas para el nivel de pruebas.
 - Adecuación de los métodos de pruebas
 - La efectividad de pruebas por objetivo

Métricas de Pruebas Comunes

- El % de finalización de la preparación de CP.
- El % de finalización de preparación del entorno de pruebas.
- La ejecución de CP. (Ejecutados/No Ejecutados, Pasados / Fallidos)
- La información de Defectos.
- La cobertura de requisitos, riesgos o código por pruebas, incluyendo los resultados paso/falla.
- El nivel de confianza (subjetivo) en el producto por parte de los probadores.
- Fechas de los hitos de prueba.
- Costo de las pruebas.

Objetivos, Contenidos y Audiencias de los Informes de Prueba

- El propósito del informe de prueba es resumir y comunicar la información de la actividad de prueba.
- Pueden denominarse informe de avance e Informe de Resumen.
- El jefe de prueba emite periódicamente el avance a los implicados.
- Los Informes Incluyen:
 - Resumen de la prueba realizada.
 - Información sobre lo ocurrido durante un período de prueba.
 - Desviaciones con respecto al plan, incluidas las desviaciones en el calendario, la duración o el esfuerzo de las actividades de prueba.
 - Estado de la prueba y calidad del producto con respecto a los criterios de salida o definición de hecho.
 - Factores que han bloqueado o continúan bloqueando el avance.
 - Métricas de defectos, casos de prueba, cobertura de la prueba, avance de la actividad y consumo de recursos. (por ejemplo, como se describe en el punto 5.3.1)
 - Riesgos residuales (véase el apartado 5.5).
 - Productos de trabajo de la prueba reutilizables desarrollados.
 - El contenido de un informe de prueba variará dependiendo del proyecto
- Estos informes también pueden incluir:
 - El estado de las actividades de prueba y el avance con respecto al plan de prueba.
 - Factores que impiden el avance.
 - Pruebas previstas para el próximo período objeto de informe.
 - La calidad del objeto de prueba.

Informe y Registro de pruebas IEEE 829

Informe del Resumen de Pruebas

Describe los resultados de un nivel o fase de pruebas, e incluye las siguientes secciones:

- Identificador del informe
- Resumen (ej. Lo que fue probado, las conclusiones, etc.)
- Varianzas (del plan, de los casos, de los procedimientos)
- Evaluación comprensiva
- Resumen de resultados (ej. Métricas finales, conteos)
- Evaluación (de cada ítem con respecto a los criterios paso/falla)
- Resumen de actividades (utilización de recursos, eficiencia, etc)
- Aprobaciones

Puede ser entregado al medio o al final de un nivel de prueba.

Registro de Pruebas

- Identificador
- La descripción (ítems sometidos a pruebas con números de versiones, entorno de pruebas, etc)

Las entradas de actividades y eventos (Prueba tras prueba, la información de evento tras evento sobre el pGraba los detalles relevantes sobre la ejecución de pruebas, e incluye las siguientes secciones:

roceso de ejecución de pruebas, los resultados de pruebas, los cambios o las cuestiones de entorno, los defectos, los incidentes o anomalías observadas, los probadores involucrados, alguna suspensión o obstrucción de las pruebas, los cambios de plan, el impacto del cambio , etc.)

El seguimiento de hojas de cálculo que capturan estos y muchos detalles son a menudo usados para este objetivo.

Gestión de la Configuración

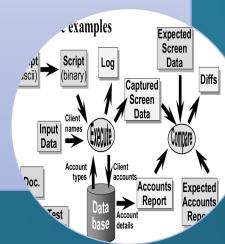
- Establecer y mantener la integridad del componente o sistema
- Implica:
 - Garantizar la trazabilidad
 - Elementos de Prueba
 - Productos de Prueba
 - Mantener la documentación

Objetivo



- La gestión de testware y los resultados
- Asegurar relación de ítems de prueba con componentes conocidos
- La entrega de una versión de pruebas al laboratorio de pruebas

Proporciona



- Guardar y controlar acceso a ítems que constituyen el sistema
- Identificar y documentar ítems bajo control
- Permitir cambio de los ítem bajo un proceso ordenado (ej. Comité de control de cambios)
- Informar sobre cambios pendientes, en proceso, y completos.
- Verificar completitud de la implementación

Tareas Clave



Nota: Durante la planificación de proyecto y pruebas, los procedimientos de gestión de configuraciones y la infraestructura deberían ser escogidos, documentados e implementados, de forma que no ocurran sorpresas en el momento de la ejecución.

Riesgo y proceso pruebas

Riesgos de Producto

- Posibles áreas de fallo en el software o sistema
 - Software proclive a fallos
 - Daños a personas o empresas
 - Malas características
 - Mala integridad y calidad de datos
 - No funciona
- “A más pruebas, menos riesgo”

Riesgos de Proyecto

- Riesgos relativos a la capacidad del proyecto de lograr los objetivos
 - Factores de organización
 - Aspectos técnicos
 - Aspectos de proveedores

Ejemplos de riesgos de Producto

- El software podría no realizar las funciones previstas de acuerdo con la especificación.
- El software puede no realizar las funciones previstas de acuerdo con las necesidades de los usuarios, clientes y/o implicados.
- Una arquitectura de sistema puede no soportar de forma adecuada algunos requisitos no funcionales.
- Un cálculo específico puede realizarse de forma incorrecta en algunas circunstancias.
- Una estructura de control de un bucle puede estar codificada de forma incorrecta.
- Los tiempos de respuesta pueden ser inadecuados para un sistema de procesamiento de transacciones de alto rendimiento.
- La retroalimentación de la experiencia de usuario (UX por sus siglas en inglés) podría no cumplir con las expectativas respecto del producto.

Ejemplos de riesgos de Proyecto

Cuestiones Asociadas al Proyecto:

Retrasos en la entrega, la finalización de tareas, el cumplimiento de los criterios de salida o la definición de hecho. Estimaciones inexactas, reasignación de fondos a proyectos de mayor prioridad o el recorte general de gastos. Cambios tardíos pueden resultar en cambios sustanciales de reelaboración.

Cuestiones Asociadas a la Organización:

Las competencias, la formación y el personal pueden no ser suficientes. Las cuestiones relacionadas con el personal pueden causar conflictos y problemas. Los usuarios, el personal de la empresa o los expertos en la materia pueden no estar disponibles debido a prioridades de negocio en conflicto.

Cuestiones de Carácter Político:

Los probadores pueden no comunicar adecuadamente sus necesidades y/o los resultados de la prueba. Los desarrolladores y/o probadores pueden no realizar un seguimiento de la información obtenida en la prueba y las revisiones (por ejemplo, no mejorar las prácticas de desarrollo y prueba). Puede haber una actitud y expectativas inadecuadas con respecto a la prueba (por ejemplo, no apreciar el valor de encontrar defectos durante la prueba).

Ejemplos de riesgos de Proyecto

Cuestiones de Carácter Técnico

Es posible que los requisitos no estén suficientemente bien definidos.

Es posible que no se cumplan los requisitos, dadas las restricciones existentes.

Es posible que el entorno de prueba no esté listo a tiempo.

La conversión de datos, la planificación de la migración y el soporte de herramientas se pueden retrasar.

Las debilidades en el proceso de desarrollo pueden afectar la consistencia o la calidad de los productos de trabajo del proyecto, como el diseño, el código, la configuración, los datos de prueba y los casos de prueba.

Una mala gestión de los defectos y problemas similares pueden dar lugar a defectos acumulados y otra deuda técnica.

Cuestiones Relacionados con los Proveedores

Un tercero puede no entregar un producto o servicio necesario, o declararse en quiebra.

Las cuestiones contractuales pueden causar problemas al proyecto.

La Prueba Basada en el Riesgo y la Calidad de Producto

- El riesgo se utiliza para distribuir el esfuerzo necesario durante la prueba.
- La prueba se utiliza para reducir la probabilidad o para reducir el impacto de un evento adverso.
- Implica el análisis del riesgo de producto, que incluye la identificación de los riesgos de producto y la evaluación de la probabilidad y el impacto de cada riesgo.
- En un enfoque basado en el riesgo, se utilizan los resultados del análisis del riesgo de producto para determinar:
 - Técnicas de prueba que se van a utilizar.
 - Los niveles y tipos particulares de pruebas que se realizarán (por ejemplo, pruebas de seguridad, pruebas de accesibilidad).
 - El alcance de la prueba que se llevará a cabo.
 - Priorización para encontrar los defectos críticos tan pronto como sea posible.
 - Si se podrían realizar otras actividades además de las pruebas para reducir el riesgo (por ejemplo, impartir formación a diseñadores sin experiencia).

Analizar

- Lo que puede salir mal



Determinar

- Qué riesgos es importante tratar



Implementar

- Acciones para mitigar esos riesgos

Gestión de Defectos

Objetivo

- Facilitar feedback
- Establecer medios de seguimiento
- Aportar ideas

Cualquier defecto identificado debe ser investigado y debe ser objeto de seguimiento desde su descubrimiento y clasificación hasta su resolución.

Para gestionar todos los defectos hasta su resolución, la organización debe establecer un proceso de gestión de defectos que incluya un flujo de trabajo y reglas de clasificación.

Los defectos pueden ser informados durante la codificación, el análisis estático, las revisiones, las pruebas dinámicas o el uso de un producto software

Elementos de un informe de defectos:

- Identificador
- Título y breve resumen
- Fecha del informe de defecto, organización emisora y autor
- Identificación del elemento de prueba
- La(s) fase(s) del ciclo de vida de desarrollo en la(s) que se observó el defecto
- Una descripción del defecto para permitir la reproducción y resolución, incluyendo registros, capturas de pantalla de volcados de bases de datos o grabaciones
- Resultados esperados y reales
- Alcance o grado de impacto (severidad)
- Urgencia/prioridad de la corrección
- Estado del informe de defecto
- Conclusiones, recomendaciones y aprobaciones
- Comentarios
- Historial
- Referencia

Diez pasos para un buen informe de defectos

- 1. Estructurar: Probar cuidadosamente**
- 2. Reproducir: Probarlo de nuevo**
- 3. Aislar: Probarlo de manera diferente**
- 4. Generalizar: Probarlo en otro lugar**
- 5. Comparar: Revisar resultados similares**
- 6. Resumir: Relacionar las pruebas con los clientes**
- 7. Condensar: Eliminar información innecesaria**
- 8. Desambiguar: usar palabras claras**
- 9. Neutralizar: Expresar el problema imparcialmente**
- 10. Revisar: estar seguro**



Estructurar

- Las pruebas estructuradas son el fundamento de buenos informes de defectos
 - Utilizar un método deliberado, cuidadoso para las pruebas
 - Seguir los casos de pruebas escritos o ejecutar los automatizados por proceso escrito o estandarizado
 - Utilizar charters para estructurar las pruebas exploratorias
 - Tomar notas cuidadosas
- Los reportes de defectos empiezan cuando los resultados esperados y observados se diferencian
- Pruebas poco rigurosas resultan en informes de defectos poco rigurosos

Reproducir

- Comprobar siempre la reproducibilidad de una falla como parte de la escritura de un informe de defectos.
- Documentar una secuencia sólida de acciones que reproducirán el fallo.
- Informar sobre los fallos intermitentes y difíciles de repetir
 - Anotar la tasa de incidencia de fallos (ej. 1 en 3 intentos)
 - El resumen debería mencionar la intermitencia
- Pasos claros para reproducir abordan el asunto “irreproducible” de frente

Aislar

- Cambiar las variables que puedan alterar el síntoma
 - Realizar los cambios uno tras otro
 - Se necesita cabeza y comprensión del sistema sometido a prueba
 - No puede ser inmediatamente obvio
- Puede ser extenso
 - Concordar la cantidad de esfuerzo para la severidad del problema
 - Evitar comenzar actividades de depuración
- Buen aislamiento muestra debida diligencia y da a los desarrolladores un comienzo en la depuración

Generalizar

- Buscar los fallos relacionados en el sistema sometido a pruebas
 - ¿Ocurre el mismo fallo en otros módulos o lugares?
 - ¿Hay más síntomas severos del mismo defecto?
- Evitar confundir los problemas no relacionados
 - El mismo síntoma puede surgir de diferentes defectos
- La generalización reduce los informes de defectos duplicados y refina la comprensión de la falla

Comparar

- Examinar los resultados para las pruebas similares
 - Las mismas ejecuciones de pruebas contra versiones anteriores
 - Condiciones similares, otras pruebas, la misma versión
- ¿Es la falla una regresión?
 - El cambio introduce el defecto no en versiones más tempranas
 - Usualmente encontradas cuando las pruebas pasadas fallan
- No es siempre posible
 - Las pruebas bloqueadas anteriormente, la reinstalación no es practicable
 - La característica probada no disponible en versiones tempranas

Resumir

- Ponga una “Línea de etiqueta” corta en cada informe
 - Capture el fallo y el impacto sobre el cliente
 - Analogía: El título del periódico
- Más difícil de lo que parece
 - Los probadores deben invertir tiempo en el resumen
- Ventajas de buenos resúmenes
 - Consiga la atención de la gerencia
 - Ayude a establecer prioridades
 - Nombre el informe de defectos para los desarrolladores
- El resumen es a menudo la única parte leída de un informe de defectos

Condensar

- Elimine palabras o pasos extraños
 - Lea de nuevo el informe cuidadosamente
 - Evite tanto el comentario misterioso como la conversación monótona
- ¿Hay algunos detalles o acciones irrelevantes?
- El tiempo de cada uno es valioso

Desambiguar

- Retire, reformule, o expanda palabras y afirmaciones vagas, erróneas o subjetivas
- Asegure que el informe no esté sujeto a mala interpretación
- Objetivo: Claro, afirmaciones de hecho indiscutibles
 - Conduzca al desarrollador hacia el defecto



Neutralizar

- Entregue las malas noticias gentilmente
- Sea imparcial en el vocabulario e implicaciones
- Evite:
 - El ataque a los desarrolladores
 - La crítica del error fundamental
 - La tentativa del humorismo
 - La utilización del sarcasmo
 - ¿Las rabietas?
- Confine los informes de defectos en afirmaciones de hecho
- Ud. Nunca sabe quién leerá los informes

Revisar

- Cada probador debería presentar cada informe de defectos a uno o más pares de pruebas para una revisión
- Los pares que revisan deberían:
 - Hacer sugerencias para mejorar el informe
 - Preguntar preguntas aclaratorias
 - Desafiar aún la “defectuosidad” si es apropiado
- El equipo de pruebas debería presentar solamente los mejores informes de defectos posibles, dadas las restricciones de tiempos apropiadas para la prioridad de los defectos

Informe de incidencias de pruebas IEEE829

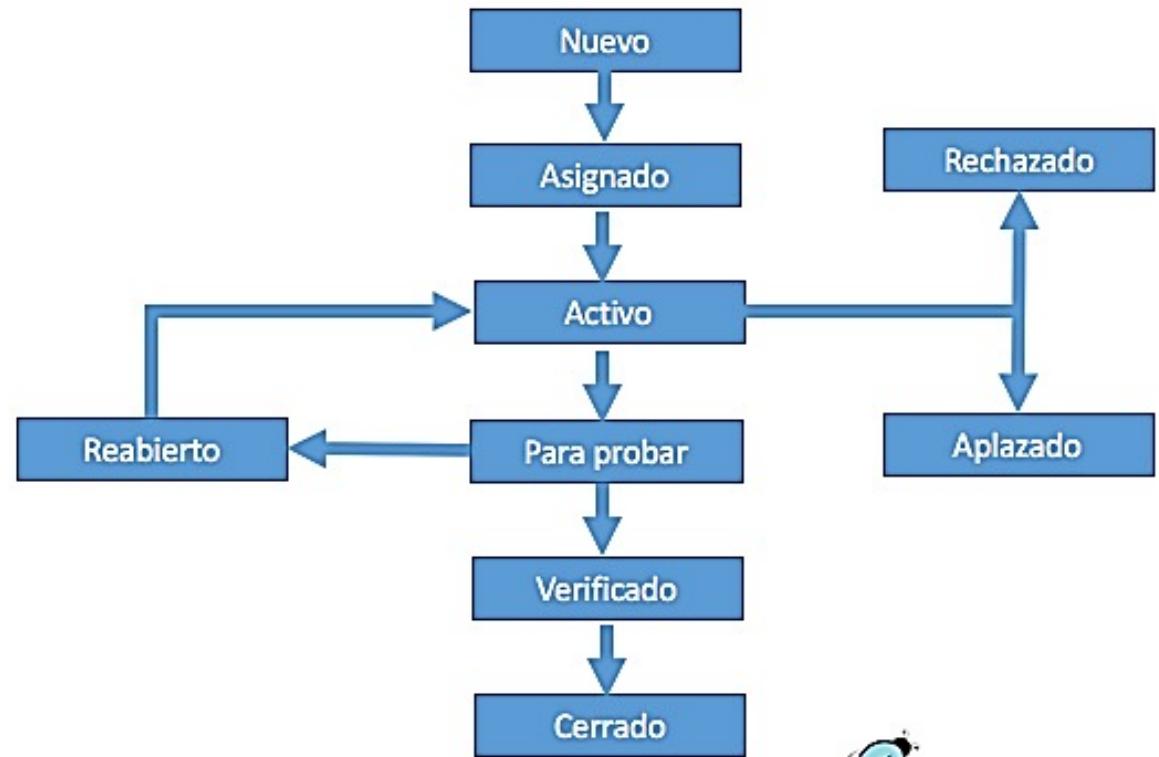
- Un informe de incidencias(o defectos) de pruebas describe un evento de pruebas que necesita más investigación, especialmente un defecto, e incluye las siguientes secciones:
 - Identificador
 - Resumen
 - Descripción de incidencias (entradas, resultados esperados, resultados reales, las anomalías, fecha y hora, entorno de pruebas en progreso, reproducibilidad, reportada por, interesados del negocio, etc.)
- Estrictamente, los informes de incidencias describen cualquier comportamiento cuestionable, mientras que los informes de defectos describen el comportamiento debido a los defectos(fallos) más que las pruebas o los datos de pruebas malos, los errores de los probadores, los problemas del entorno de prueba o similares.

Otra información a incluir

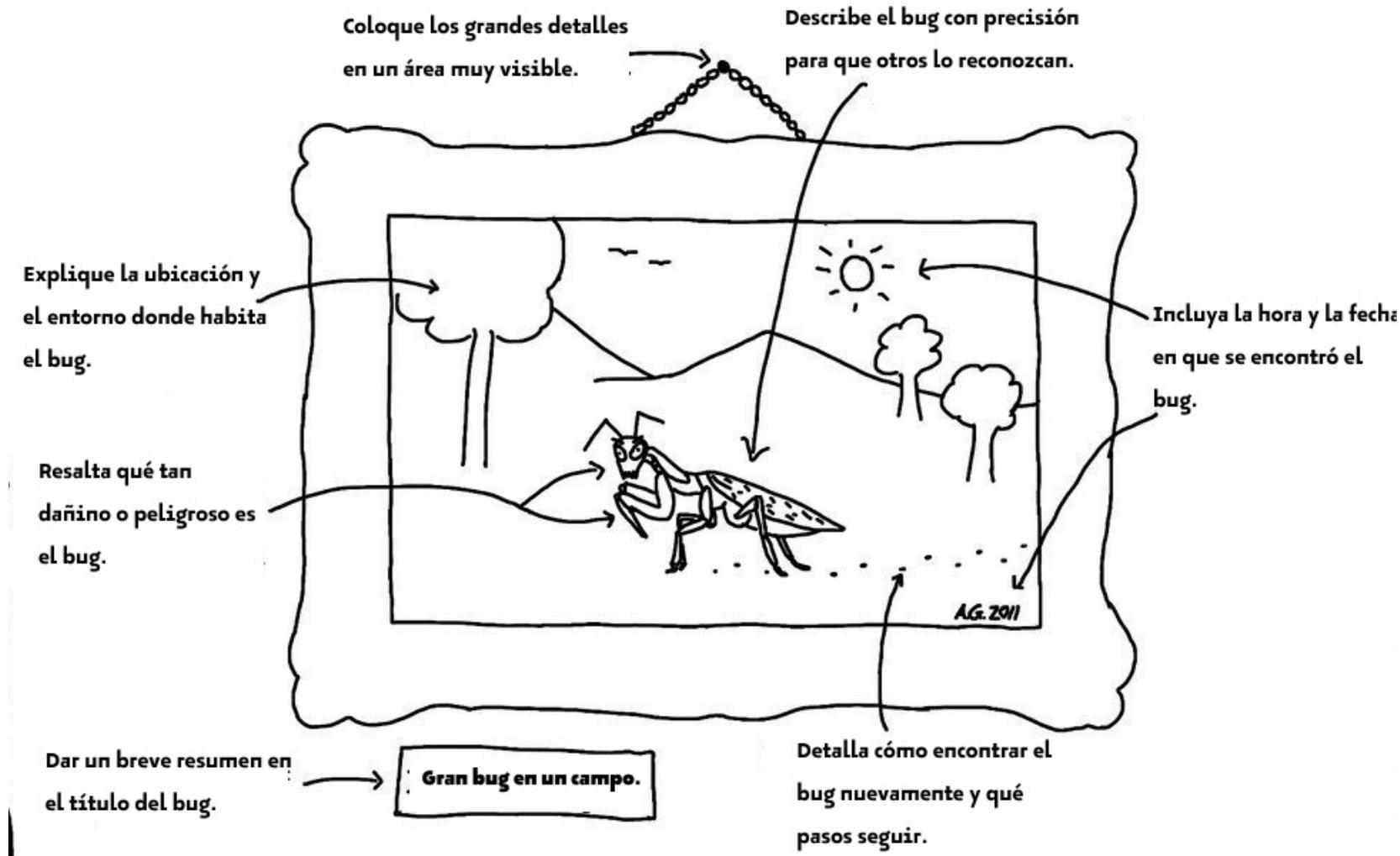
- La configuración del software o sistema
- La fase de introducción, detección y retiro del defecto
- La urgencia / prioridad a corregir
- Las conclusiones y recomendaciones
- Los riesgos, costos, las oportunidades, y los beneficios de corregir / no corregir
- La historia de cambios, especialmente para cambio de estado
- La fecha del informe, la organización que informa, y el autor
- Los resultados esperados y reales
- LA identificación del ítem y entorno de pruebas
- El proceso del ciclo de vida en el cual la incidencia fue observada
- El alcance o el grado de impacto en los intereses de los interesados del negocio
- La severidad del impacto en el sistema

Ciclo de vida o Flujo de Trabajo del informe de defectos

- Los informes de defectos se mueven a través de una serie de estados hasta la resolución
- En cada estado no terminal, un jefe o comité de clasificación de defectos especifica un dueño quien es el que mueve el defecto hacia el próximo estado
- Los sistemas de seguimiento de defectos pueden y deberían implementar y automatizar estos flujos de trabajo, pero el equipo del proyecto y el soporte de gerencia hacen que el flujo de trabajo funcione



El Arte de Informar Bugs



Los Bugs tambien tienen sentimientos

Si encuentra un Bug:
Reportalo.

A los bugs no les gusta que los alviden.



Si encuentra un Bug:
infórmelo rápido.

De lo contrario, los bugs se instalan y se hacen un hogar.



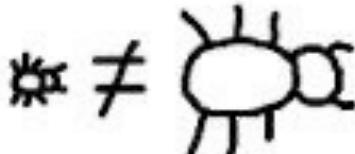
Si encuentra un Bug:
Acceda a ellos.

A los bugs les gusta que los entiendan.



Si encuentra un Bug:
sea honesto.

A los bugs no les gustan las mentiras.



Si encuentra un Bug:
Tome una foto.

A los bugs les gusta mantener los recuerdos de la ocasión.



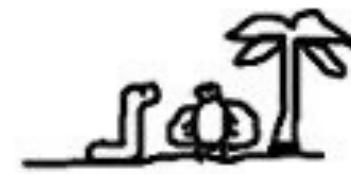
Si encuentra un Bug: Observe
cómo los encuentra.

Los bugs son románticos.



Si encuentra un Bug:
conozca a sus compañeros.

Los bugs son muy sociables.



Si encuentra un Bug:
No lo ignore.

Los bugs pueden dañar si no se les aprecia.



Taller

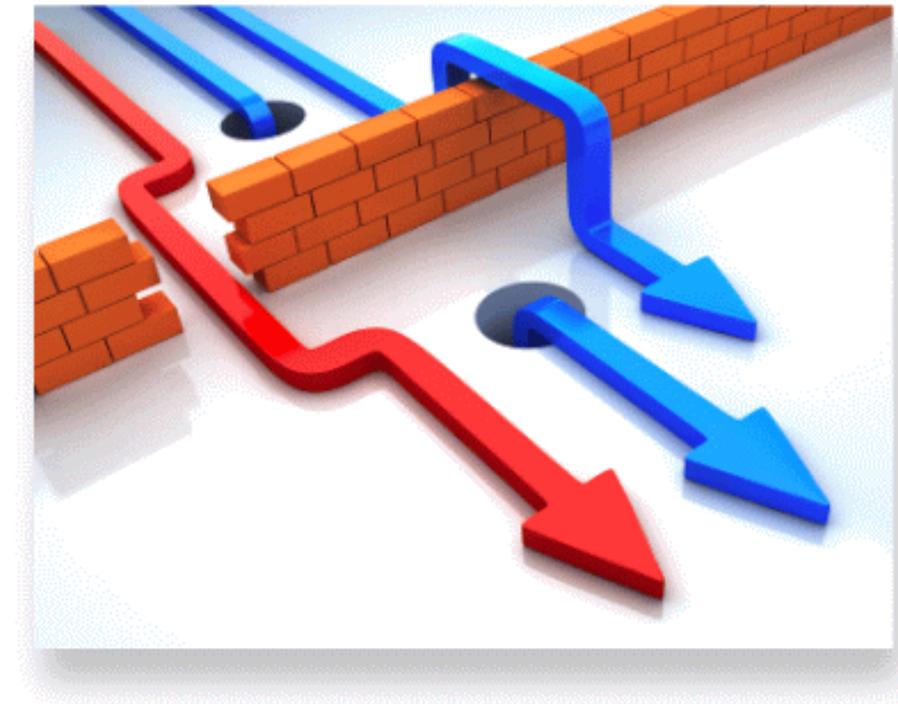
- Introducción y Objetivos
- I. Fundamentos de Pruebas
- II. Pruebas a través del Ciclo de Vida de Software
- III. Técnicas Estáticas
- IV. Técnicas de Prueba
- V. Gestión de Pruebas
- VI. Soporte de Herramientas para Pruebas**

CAPITULO VI

Soporte de Herramientas para Pruebas

Propósito de las herramientas de pruebas

- Pueden mejorar las eficiencias de las actividades de pruebas.
- Pueden automatizar actividades manuales difíciles o imposibles.
- Pueden incrementar la facilidad de las pruebas.
- Aumentar la fiabilidad de las pruebas.
- Pueden ser intrusivas o no intrusivas.



Clasificación de Herramientas según Actividades que soportan

Soporte de Herramientas para Gestión de la Prueba y Productos de Prueba

Se pueden utilizar en cualquier actividad de prueba a lo largo de todo el ciclo de vida de desarrollo de software.

Pueden incluir:

- Herramientas de gestión de prueba y herramientas de gestión del ciclo de vida de las aplicaciones.
- Herramientas de gestión de requisitos.
- Herramientas de gestión de defectos.
- Herramientas de gestión de la configuración.
- Herramientas de integración continua.



Rational
Quality
Manager



Quality Center



ZPHYR



TestLink



Herramientas de pruebas

Herramientas de gestión de pruebas

- Proporcionan trazabilidad de las pruebas, los resultados y las incidencias para las bases de pruebas
- Permiten el registro de los resultados de pruebas y la generación de informes de progreso
- Ayudan la gestión de pruebas y del proceso de pruebas
- Proporcionan una interfaz para la ejecución de pruebas, el seguimiento de defectos, y las herramientas de gestión de requisitos
- Proporcionan el control de versiones o la interfaz con una herramienta externa de gestión de configuraciones
- Realizan un análisis cuantitativo (métricas) relacionado con las pruebas

Herramientas de gestión de requisitos

- Guardan las declaraciones de los requisitos
- Comprueban la consistencia y los requisitos no definidos (faltantes)
- Permiten la priorización de los requisitos
- Hacén posible que las pruebas individuales sean trazables a (e informadas en los términos de) los requerimientos, las funciones, y/o las características

Clasificación de Herramientas según Actividades que soportan

Soporte de Herramientas para pruebas estáticas

Utilizadas principalmente por los desarrolladores:

- Encuentran defectos antes de las pruebas dinámicas
- Hacen valer los estándares de código
- Analizan las estructuras y dependencias
- Ayudan en la comprensión del código fuente
- Calculan las métricas del código

Pueden incluir:

- Herramientas de apoyo a las revisiones.
- Herramientas de análisis estático (D).



Clasificación de Herramientas según Actividades que soportan

Soporte de Herramientas para el Diseño e Implementación de Pruebas

Ayudan a crear productos de trabajo mantenibles en el diseño e implementación de pruebas, incluyendo casos de prueba, procedimientos de prueba y datos de prueba

Pueden incluir:

- Herramientas de diseño de pruebas.
- Herramientas de Prueba Basada en Modelos.
- Herramientas de preparación de datos de prueba.
- Herramientas de desarrollo guiado por prueba de aceptación (ATDD) y de desarrollo guiado por el comportamiento (TDD)
- Herramientas de desarrollo guiado por pruebas (D)

Clasificación de Herramientas según Actividades que soportan

Soporte de Herramientas para la ejecución y el Registro de Pruebas

Mejoran las actividades de ejecución y registro de pruebas.

Pueden incluir:

- Herramientas de ejecución de pruebas (para ejecutar pruebas de regresión)
- Herramientas de cobertura (D)
- Arneses de prueba (D)
- Herramientas de marco de trabajo de pruebas unitarias (D)

Clasificación de Herramientas según Actividades que soportan

Soporte de Herramientas para la Medición del Rendimiento y el Análisis Dinámico

son esenciales para dar soporte a las actividades de pruebas de rendimiento y de carga, ya que estas actividades no se pueden realizar de forma eficaz de forma manual.

Pueden incluir:

- Herramientas para la prueba de rendimiento.
- Herramientas de monitorización.
- Herramientas de análisis dinámico (D).

Clasificación de Herramientas según Actividades que soportan

Soporte de Herramientas para Necesidades de Prueba Especializadas

Dan soporte a cuestiones más específicas de la prueba

Pueden ser:

- Evaluación de la calidad de los datos.
- Conversión y migración de datos.
- Prueba de usabilidad.
- Prueba de accesibilidad.
- Prueba de localización.
- Prueba de seguridad.
- Prueba de portabilidad.

Otras herramientas de pruebas



Herramientas de seguimiento de Bugs (o de defectos o incidencias)

- Guardan y gestionan los informes de defectos
- Facilitan la priorización / clasificación de defectos
- Proporcionan un flujo de trabajo basado en los estados incluyendo la asignación de acciones a las personas (ej. Corregir, confirmar, etc)
- Hacen posible el monitoreo de los defectos del proyecto y del estado de los defectos con el tiempo, específicamente con respecto a las tendencias
- Proporcionan el soporte para el análisis estadístico muchas veces a través de la exportación
- Crean informes y gráficos

Herramientas de gestión de configuraciones

- Guardan la información acerca de las versiones y compilaciones del software y testware
- Hacen posible la trazabilidad entre el testware y productos de trabajos de software y las variantes del producto
- Ayudan con el desarrollo y las pruebas en múltiples configuraciones de entornos de hardware / software

Otras herramientas de pruebas

Herramientas de Soporte del Proceso de Revisión

- Guardan la información sobre los procesos de revisión
- Guardan y comunican los comentarios de las revisiones
- Informan sobre los defectos y el esfuerzo
- Gestionan las referencias para revisar las reglas y/o las listas de comprobación
- Proporcionan ayuda para las revisiones en línea
- Siguen la trazabilidad entre los documentos y el código fuente

Herramientas de Diseño de Pruebas

- Generan las entradas de pruebas o las pruebas reales de:
 - Los requisitos
 - La interfaz de usuario gráfica
 - Los modelos de diseño
 - El código
- Generan los resultados esperados (aunque la confiabilidad de tales oráculos de pruebas es a menudo limitada)
- Generan los marcos de trabajo de las pruebas y las plantilla

Otras herramientas de pruebas

Herramientas de Datos de Pruebas

- Manipulan o crean bases de datos, archivos o datos para utilizar durante la ejecución de pruebas
- Crean grandes volúmenes de datos de pruebas útiles
- Validan los datos de pruebas según reglas específicas
- Analizan los datos por la frecuencia de condiciones, etc.
- Mezclan y anonimizan los datos en vivo o de clientes

Herramientas de Seguridad

- Comprueban los virus de computadora
- Simulan los varios tipos de ataques
- Simulan las varias condiciones de seguridad
- Comprueban el código de violaciones de la seguridad conocidas

Otras herramientas de pruebas

Rendimiento, Monitoreo, Análisis Dinámico

- Utilizados principalmente por desarrolladores, aunque muchos probadores utilizan herramientas de rendimiento y de monitoreo
- Las herramientas de análisis dinámico son utilizadas a menudo para comprobar las dependencias de tiempo o fugas de memoria
- Monitorean e informan sobre cómo un sistema se comporta bajo condiciones de uso simuladas
- Generan varias condiciones (esperanzadamente realistas) de carga para la aplicación, una base de datos, red o un servidor, a menudo por algún guion o procedimiento programado
- Monitorean, analizan, verifican e informan sobre el uso de recursos del sistema específicos, y dan advertencias de problemas posibles

Beneficios y riesgos de la automatización

Beneficios

- Trabajo repetitivo reducido
- Consistencia y capacidad de repetición mejoradas
- La evaluación objetiva (Especialmente de cobertura, rendimiento, etc.)
- Fácil acceso a información sobre las pruebas

Riesgos

- Expectativas poco realistas
- Subestimación de tiempo y esfuerzo
- Aplicación errónea
- Subestimación de mantenimiento
- Problemas en gestión de la configuración
- Suspensión del código abierto
- Interoperabilidad
- Huérfano / negligente
- Punto ciego



Teoría y práctica de las pruebas automatizadas

La automatización de pruebas es típicamente desarrollo de software

- Proceso no trivial que requiere tiempo, habilidad y dinero significativos.
- El retorno de inversión (ROI) de la automatización toma típicamente más tiempo que un proyecto.
- Excepciones: Pruebas unitarias basadas en un API, de componente y de integración. Pruebas no funcionales

Normalmente, lo que es automatizado es solo la ejecución de las pruebas, no el análisis de resultados

- La automatización incorrecta puede complicar el análisis y aumentar el número de las pruebas fallidas.

Si solo se enfoca en la automatización de las pruebas, la automatización parece muy eficiente pero no debe olvidarse que las pruebas automatizadas tuvieron que ser automatizadas por alguien.



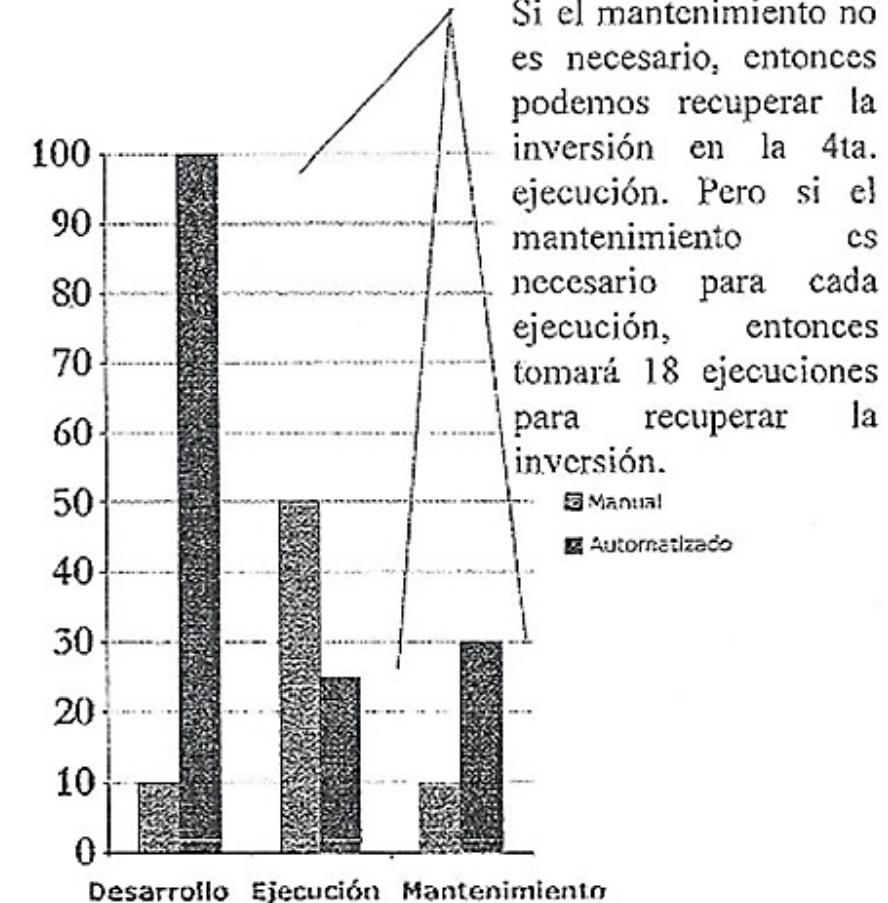
Retorno de Inversión (ROI) de la automatización de Pruebas

El retorno de la mayoría de los esfuerzos requiere:

Suficientemente altos riesgos de regresión para justificar la repetición de las pruebas.

Un diseño de sistema de pruebas que mantenga el costo de ejecutar y mantener las pruebas automatizadas bastante más abajo que el costo de las pruebas manuales equivalentes.

Un sistema relativamente estable



Eligiendo Pruebas Manuales o Automatizadas

Pruebas adecuadas para las pruebas manuales	Pruebas adecuadas para las pruebas automatizadas	Manual, automatizado o combinado
<ul style="list-style-type: none"> • Operaciones y mantenimiento • Configuración y compatibilidad • Manejo de errores y recuperación • Localización • Usabilidad • Instalación y configuración • Documentación y ayuda 	<ul style="list-style-type: none"> • Regresión y confirmación • Pruebas de Mono (o aleatorias) • Carga, volumen y capacidad • Rendimiento y fiabilidad • Cumplimiento de estándares • Caja blanca, especialmente de unidad basadas en un API, de componente, integración • Complejidad estática y análisis de código 	<ul style="list-style-type: none"> • Funcional • Casos de Uso (escenarios de usuarios) • Manejo de fecha y hora

Las pruebas manuales inapropiadas engañan a la gente sobre la cobertura de las pruebas.

La automatización inapropiada normalmente falla. Algunas pruebas mezclan las técnicas.

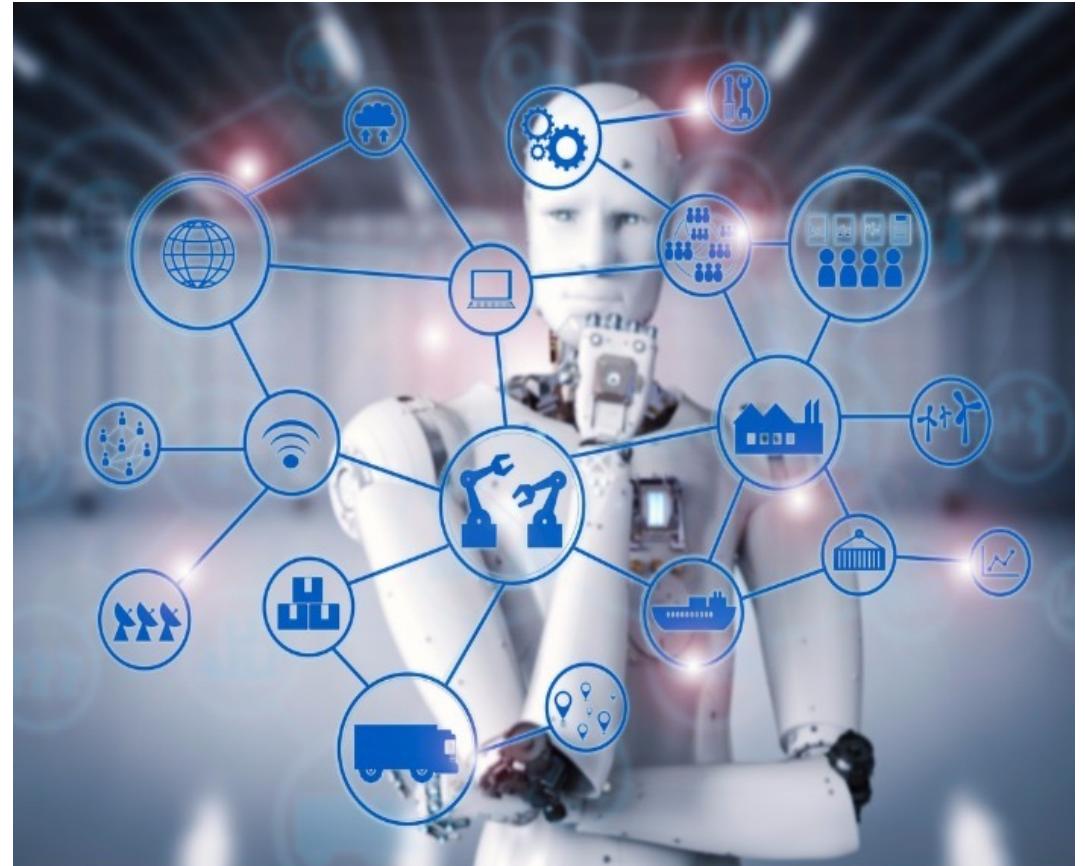
Pruebas para el Éxito con Herramientas de Ejecución de Pruebas

- La captura-repetición parece que rebaja los costos de ejecución, pero los costos de ejecución y mantenimiento para un gran número de pruebas son altos.
 - Las pruebas dirigidas por datos separan las entradas (datos) de los procedimientos (guiones) que utilizan los datos.
 - Los expertos, que no saben de automatización:
 - Pueden crear datos de pruebas.
 - Pueden crear archivos de palabras clave
 - Las pruebas dirigidas por palabras clave usan palabras clave (o palabras de acción) en un archivo que es leído por el guion.
 - Los expertos de automatización crean los guiones.

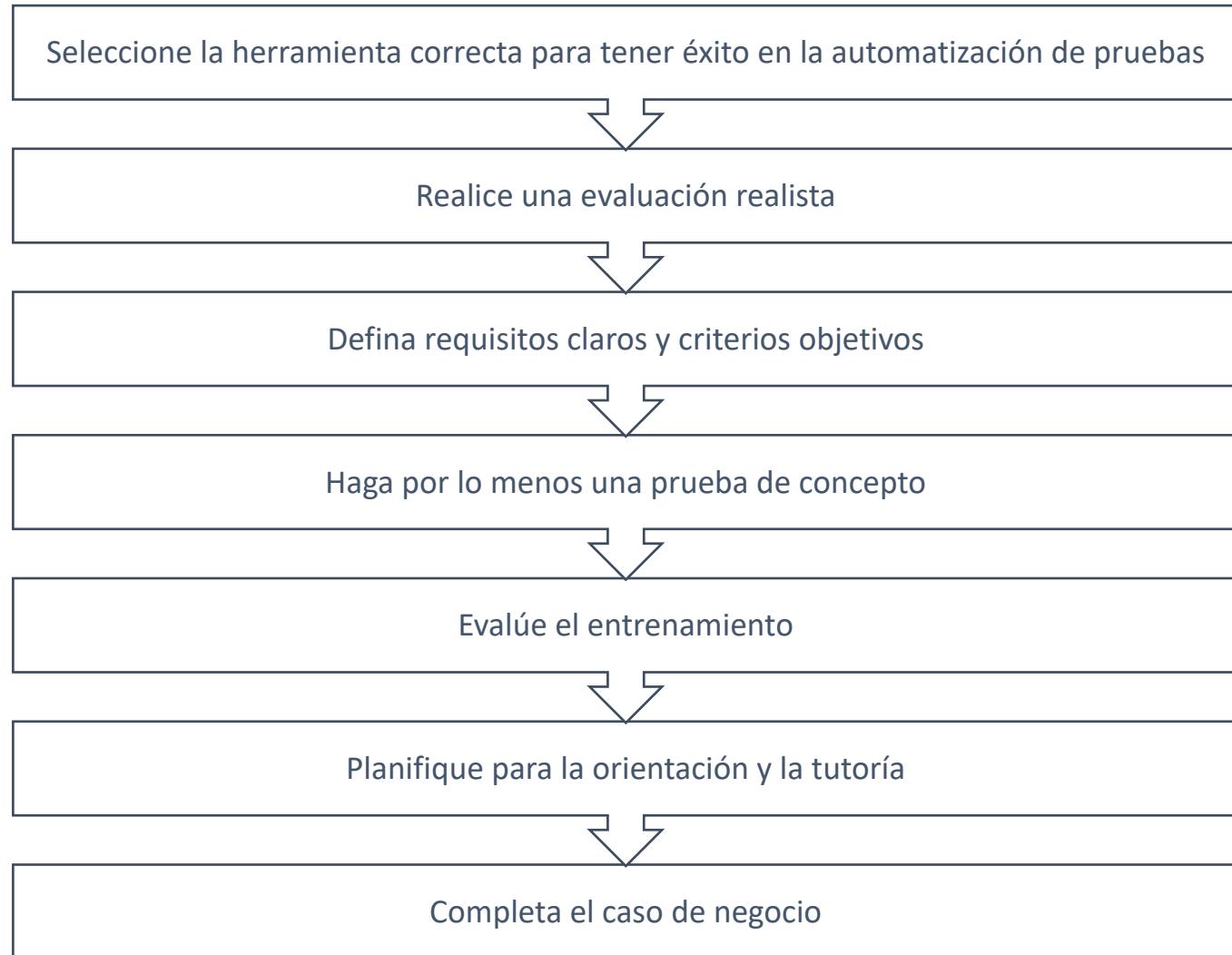


Pruebas para el Éxito con otras Herramientas

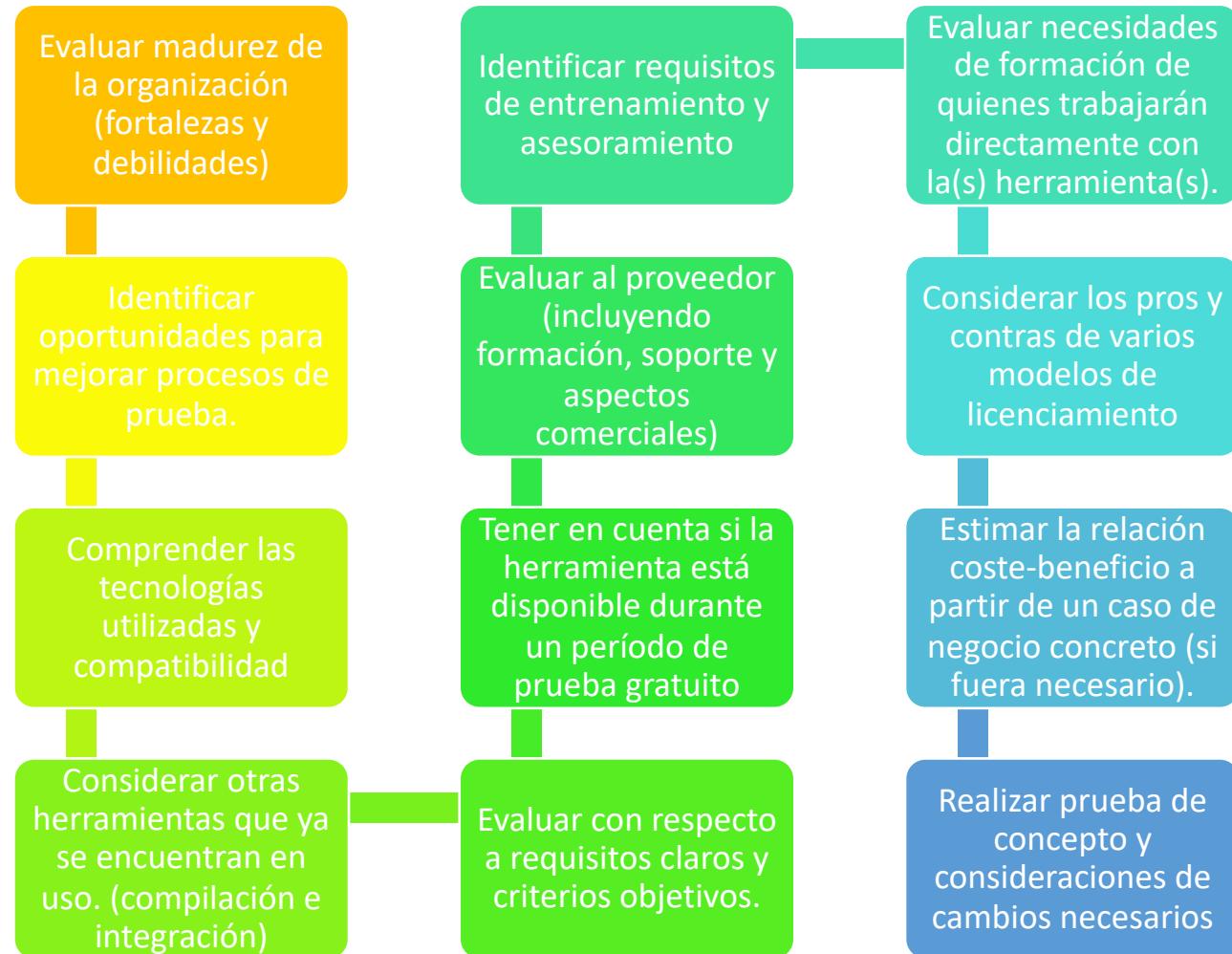
- Las herramientas de pruebas de rendimiento necesitan experticia en el rendimiento para diseñar las pruebas e interpretar los resultados.
- Las herramientas de análisis estático aplicadas al código fuente pueden hacer valer los estándares de código, pero una introducción en fases para el código existente es necesaria para gestionar el volumen de errores encontrados.
- Las herramientas de gestión de pruebas tienen frecuentemente una funcionalidad deficiente para la creación de informes, así que exportar datos a las hojas de cálculo es normalmente la mejor manera para producir los informes deseados.



Selección de una herramienta (v2011)



Principios Básicos para la Selección de Herramientas (v2018)



Rompiendo las Barreras para la Automatización

Problemas a ser superados

Cuando estos problemas permanecen, la automatización normalmente falla y se crea una resistencia organizacional a la automatización de proyectos posteriores

Proceso de pruebas

Personal

Estabilidad

Expectativas

Selección de herramientas

Caótico,
reactivo, falto
de tiempo

Ponga primero el proceso de pruebas manual bajo control, después automatice

Habilidad insuficiente

Contrate o capacite

Sistema inestable, que cambia rápidamente

Estabilice el sistema, GUI y API primero

Poco realistas por parte de la gerencia

Comunique de manera efectiva los beneficios y limitaciones de la automatización

No hay una apropiada u oráculo para los riesgos de calidad importantes

Construya su propia herramienta u oráculo o espere

Objetivos y factores de éxito

Objetivos de un Proyecto piloto de automatización

- Aprender más sobre la herramienta y cómo utilizarla
- Adaptar la herramienta y/o los procesos y prácticas para ajustarse a la organización y los sistemas
- Estandarizar las manera de utilización, gestión, almacenamiento y mantenimiento de la herramienta y los activos de pruebas
- Evaluar el retorno de la inversión

Factores de Éxito para Despliegue de Herramientas

- Despliegue la herramienta al resto de la organización incrementalmente
- Adapte y mejore los procesos de pruebas para ajustarse a la utilización de la herramienta
- Proporcione capacitación, orientación y tutoría para los nuevos usuarios
- Defina guías de uso de la herramienta
- Aprenda continuamente maneras para mejorar la utilización de la herramienta
- Proporcione soporte, orientación y tutoría
- Reúna las lecciones aprendidas
- Monitoree la utilización y los beneficios de la herramienta

Trampas de las Herramientas de Pruebas

- No hay estrategia clara
- Expectativas poco realistas
- Falta de compromiso de los interesados del negocio
- Entrenamiento inadecuado o de mala calidad de la herramienta
- Automatización de la cosa equivocada
- Selección de la herramienta equivocada
- Problemas de usabilidad con la herramienta
- Selección del proveedor equivocado
- El sistema sometido a pruebas inestable
- Hacer demasiado, demasiado pronto
- Subestimación del tiempo y los recursos necesarios
- Entorno de pruebas inadecuado o único
- Método correcto, mal momento
- Costo excesivos

GRACIAS!!!!