TESTING

QUÉ ES

Probar si funciona todo de acuerdo a lo esperado. Un tester parte siempre de la suposición de que el programa tiene errores. El testing brinda calidad a los desarrollos. La calidad, es la satisfacción del cliente ante un producto o servicio. El testing brinda confianza. Realizar pruebas frecuentes es vital en el proceso de desarrollo. El testing sirve para:

1.Encontrar defectos y remediarlos 2. Asegurar el buen funcionamiento del producto.

3. Logran un mayor grado de calidad.

PRINCIPIOS (7)

1.La prueba muestra la presencia de defectos, no su ausencia. 2. La prueba exhaustiva es imposible. 3. La prueba temprana ahorra tiempo y dinero. 4. Los defectos se agrupan. 5. Cuidado con la prueba del pesticida.

6. La prueba se realiza de manera diferente según el contexto. 7. La ausencia de errores es una falacia.

La forma en que se implementa la independencia de la prueba varía dependiendo del modelo de ciclo de vida de desarrollo de software. Por ejemplo, en el desarrollo ágil, los probadores pueden formar parte de un equipo de desarrollo.

PRUEBA INDEPENDIENTE

BENEFICIOS:

1.Es probable que los testers independientes reconozcan diferentes tipos de fallos en comparación con los desarrolladores, debido a sus diferentes contextos, perspectivas técnicas y sesgos. 2.Un tester independiente puede verificar, cuestionar o refutar las suposiciones hechas por los implicados durante la especificación e implementación del sistema.

DESVENTAJAS 1.Los desarrolladores pueden perder el sentido de la

responsabilidad con respecto a la calidad. 2. Los testers independientes pueder ser vistos como un cuello de botella o culpados de retrasos en el lanzamiento o liberación. 3. Los testers independientes pueden carecer de información importante.

MESA DE 3 PATAS

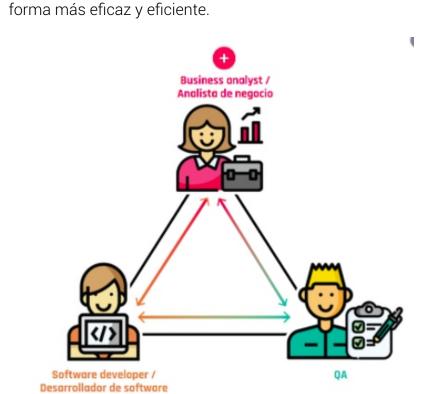
Si bien cada actor tiene un rol definido, es necesario un trabajo en comunión entre los 3 actores. Es decir, es necesario que trabajen como equipo. Por eso, utilizamos la analogía con una mesa de 3 patas, pues si falta alguna de ellas, la mesa no podría estar de pie.

En algunas empresas de software pequeñas o "start up" es posible que una misma persona tenga más de un rol.

Business analyst: Se encarga de detectar los factores clave del negocio y es el intermediario entre el departamento de sistemas y el cliente final.

Software developer: Su función es diseñar, producir,

programar o mantener componentes o subconjuntos de software conforme a especificaciones funcionales y técnicas para ser integrados en aplicaciones. QA: La principal función es probar los sistemas informáticos para que funcionen correctamente de acuerdo a los requerimientos del cliente, documentar los errores encontrados y desarrollar procedimientos de prueba para hacer un seguimiento de los problemas de los productos de



DEFECTOS

QUÉ ES

Error: se produce por una equivocación de una persona. El error genera un defecto en el software que desencadena en un fallo del sistema al ejecutarse. Error del programador, defecto en el software, fallo del



CICLO DE VIDA DE UN DEFECTO



Nuevo/Inicial: Se recopila la información y se registra el

Asignado: Si es un defecto válido y debe solucionarse se asigna al equipo de desarrollo, sino se puede rechazar o diferir (bug triage).

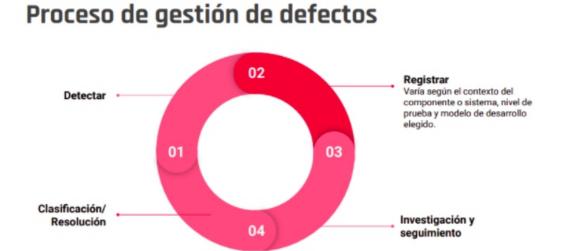
• Devuelto o rechazado: Se solicita más información o el

• Duplicado: Si el defecto se repite o existe otro con una

- receptor rechaza el defecto.
- Diferido: el defecto no es prioritario y se solucionará en una próxima versión. En progreso: Se analiza y trabaja en la solución
- Corregido: Se realizan los cambios de código para solucionar el defecto. En espera de verificación: En espera de que sea asignado a un
- probador. El desarrollador está a la expectativa del resultado
- de la verificación En verificación: El probador ejecuta una prueba de
- confirmación. • Reabierto: debe contener la siguiente descripción "La prueba de confirmación indica que el defecto no se ha
- confirmación. Cerrado: El defecto fue corregido y se encuentra disponible para el usuario final.

Verificado: Se obtiene el resultado esperado en la prueba de

PROCESO GENERAL



Objetivos



Brindar información sobre cualquier evento adverso que haya ocurrido, para poder identificar efectos específicos, aislar el problema con una prueba de reproducción mínima y corregir los defectos potenciales.



prueba un medio para mejora de los procesos hacer un seguimiento de la de desarrollo y prueba. calidad del producto de trabajo y del impacto en la

PROCESO DE GESTIÓN DE DEFECTOS

Si el defecto se reporta eficientemente, las probabilidades de que sea solucionado rápidamente es mayor. Entonces, la solución de un defecto dependerá de la eficiencia con que se reporte.

REPORTE O INFORME DE DEFECTOS

- **CONDICIONES A TENER EN CUENTA** 1. Los bugs deben tener identificadores únicos.
- 2. Una falla debe ser reproducible para reportarla. Si el defecto no es reproducible, no es un
- 3. Ser específico: no se deben escribir suposiciones u otra cosa que no sea información relevante para poder reproducir el defecto. 4. Reportar cada paso realizado para reproducirlo. No debemos obviar ningún paso que sea
- relevante para llegar al error en cuestión.

PROBLEMAS COMUNES EN LA INFORMACIÓN DE DEFECTOS

- Redactar un defecto de manera excesivamente coloquial y ambigua. • Dar solo una captura del defecto sin indicar qué se estaba haciendo cuando sucedió.
- No incluir en la descripción del defecto cuál era el resultado esperado para los pasos
- No determinar un patrón con el cual el defecto ocurre antes de reportar el mismo —es importante para ser directos en cuál es el problema—.
- No leer el defecto reportado siguiendo los pasos uno mismo para ver que la descripción es clara.
- No incluir información que dada las características del defecto, la misma es de relevancia.

PARTES DE UN REPORTE DE DEFECTOS

	Atributo Descripción		
ID	Abreviatura de identificador, un código único e irrepetible que puede ser número o letras.	001 - Test01	
Título	El título debe ser corto y específico, que se entienda en este lo que queremos reportar. Cuando el desarrollador o el equipo vean el título pueden interpretar rápidamente qué es, dónde está y cuán importante es ese defecto.	Login - Ingresa con campos en blanco	
Descripción	Describir un poco más sobre el error, es decir, desarrollar lo que dejamos afuera en el título lo podríamos explicar acá.	En la pantalla login si dejo vacío los campos nombre y password y apretó ingresar, me lleva a la página principal.	
Fecha del informe del defecto	La fecha que detectó el defecto para saber posteriormente el tiempo en que se resolvió.	23/04/21	
Autor	El nombre del tester que descubrió el defecto, por si el desarrollador tiene una duda, sabe a quién consultar.	Pepito Román	
Identificación del elemento de prueba	Nombre de la aplicación o componente que estamos probando.	Carrito compras	
Versión	Es un número que nos indica en qué versión está la aplicación.	1.0.0	
Entorno	El entorno en el que probamos (desarrollo, QA, producción).	Desarrollo	
Pasos a reproducir	Los pasos a seguir para llegar al defecto encontrado.	1) Ingresar a la aplicación. 2) Dejar en blanco el campo nombre. 3) Dejar en blanco el campo password. 4) Hacer click en el botón "Ingresar".	
Resultado esperado	Es lo que esperamos que suceda o muestre la aplicación muchas veces según los requerimientos de la misma.	No debe ingresar a la aplicación sin un usuario y una contraseña válidos.	
Resultado obtenido o actual	Es lo que sucedió realmente o lo que nos mostró la aplicación. Puede coincidir o no con el resultado esperado, si no coincide, hemos detectado un error o bug.	Ingresa a la aplicación sin usuario y sin contraseña.	
Severidad	Cuán grave es el defecto que hemos encontrado, puede ser: bloqueado, crítico, alto, medio, bajo o trivial.	Crítico	
Prioridad	Con esto decimos qué tan rápido se debe solucionar el defecto, puede ser: alta, media, baja	Alta	
Estado del defecto	Los estados pueden ser: nuevo, diferido, duplicado, rechazado, asignado, en progreso, corregido, en espera de verificación, en verificación, verificado, reabierto y cerrado.	Nuevo	
Referencias	Link al caso de prueba con el cual encontramos el error.	https://repositorio com.ar/TC-001-User Login	
lmagen	Se puede adjuntar una captura de pantalla del error, esto nos permite demostrar que el error sucedió y al desarrollador lo ayuda a ubicar el error.	Error!	

VALIDACIÓN VS VERIFICACIÓN

Verificación es testear que el proyecto se adecue a las especificaciones y requerimientos establecidos. Construí lo que dije que iba a hacer?

Validación es testear que tan bien se adecua a las necesidades de negocio y del usuario que derivaron en la redacción de los requerimientos.



¿Estamos construyendo el producto correctamente?



¿Estamos construyendo el producto correcto?

QUÉ ES

Conjunto de pre condiciones, entradas y resultados esperados, desarrollados para impulsar la ejecución de un elemento de prueba para cumplir con los objetivos de la prueba, incluyendo la implementación correcta, identificación de errores, el chequeo de la calidad y otra información valiosa.

Es un documento escrito que proporciona información sobre **qué y cómo probar.**

CARACTERÍSTICAS DE UN BUEN CASO DE PRUEBA 1. Deben ser simples, para que otra persona que no sea el autor, pueda ejecutarlos.

2. El título debe ser fuerte. Solo leyendo el mismo,

cualquier tester deberia comprender el objetivo del caso de prueba. 3. Tener en cuenta al usuario final.

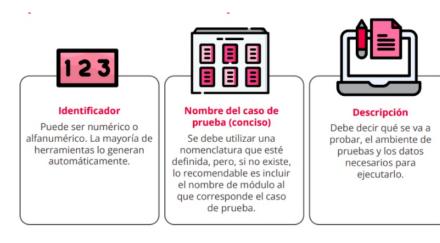
4. No asumir la funcionalidad y las caracteristicas de la aplicación mientras se prepara el caso. Ser fiel a los documentos de especificación.

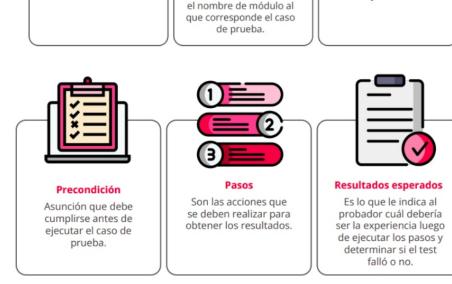
5. Asegurar la mayor cobertura posible de los requisitos

especificados. 6. Autonomía. El caso debe generar los mismos resultados siempre sin importar quien lo pruebe.

7. Evitar la repetición de casos de prueba.

QUÉ DEBE CONTENER





Estados de un caso de prueba <u>v</u> **×** Estados de Ejecución Estados de Diseño zado durante la fase de ejecución de s utilizado durante la fase de diseño la prueba v puede tomar alguno de los de la prueba y puede tomar alguno siguientes valores de los siguientes valores: En Diseño PasadoFallado Diseñado En Revisión No Aplica Revisado. Bloqueado Diferido

* Los estados dependen de lo definido en el Plan de Pruebas y/o de la herramienta utilizada

TESTING POSITIVO VS TESTING NEGATIVO

Testing positivo (+)

Testing negativo (-)

Son aquellos casos de prueba que validan el flujo normal de un sistema bajo prueba. Es decir, flujos que están relacionados a los requisitos funcionales del

sistema bajo prueba.

Son aquellos casos de prueba que validan flujos no contemplados dentro de los requisitos de un sistema bajo prueba.

HAPPY PATH

Es el único camino con el que se prueba una aplicación a través de escenarios de prueba cuidadosamente diseñados, que deberían recorrer el mismo flujo que realiza un usuario final cuando usa la aplicación de manera regular.

Generalmente es la primera forma de prueba que se realiza en una app, y se incluye en la categoría de PRUEBA POSITIVA. Su propósito no es encontrar defectos, sino ver que un producto o procedimiento funcione como se ha diseñado.

VENTAJAS

1. Se utiliza para conocer los estándares básico de la app. Es la primera prueba que se realiza. 2. Se utiliza para determinar la estabilidad de la

aplicación antes de comenzar otros niveles de prueba 3. Ayuda a identificar cualquier problema en una etapa temprana y ahorrar esfuerzos posteriores.

DESVENTAJAS

1.No garantiza la calidad del producto porque el proceso solo usa escenarios de prueba positivos. 2. Encontrar este camino único requiere un gran conocimiento del uso de la app y necesidades del

CASO DE USO

Un caso de uso cuenta la historia de como un usuario interactua con un sistema de software para lograr o abandonar un objetivo. Cada caso puede contener multiples rutas que el usuario sigue, estos caminos son denominados escenarios de caso de uso. Son documentos que van a ser la base para la generación de los casos de prueba. Estos documentos van a asegurar los requisitos del cliente. Generalmente, estos requisitos se encuentran escritos como casos de uso. Lo hace el analista de negocio.

En cambio un caso de prueba, cubre el software en mayor profundidad y detalle que un caso de uso. Incluyen todas las funciones que el programa es capas de realizar. Tienen en cuenta el uso de todo tipo de datos de entrada / salida, cada comportamiento esperado y todos los elementos de diseño. Se escriben 1 o mas casos de prueba para cada escenario de uso.

Partes del caso de uso	Partes del caso de prueba		
Nombre del caso de uso	Nombre del caso de prueba		
Precondiciones del caso de uso	Precondiciones del caso de prueba		
Secuencia normal y secuencia alternativa	Pasos del caso de prueba		
Resultados en la secuencia normal o alternativa. Poscondiciones del caso de uso	Resultado esperado		

La capacidad para crear casos de prueba a partir de los casos de uso y hacer la trazabilidad de unos a otros es una habilidad vital para asegurar un producto de calidad.

PRUEBAS DE CASOS DE USO

Es una técnica de **caja negra** donde se verifica si la ruta utilizada por el usuario está funcionando según lo esperado o no. Se pueden crear uno o más casos de prueba para cada comportamiento detallado en los casos de uso -comportamiento básico o normal, excepcionales o alternativos y de tratamiento de errores-.

La cobertura se mide de la siguiente manera: Cobertura = Comportamientos o rutas del caso de uso probadas Comportamientos o rutas del caso de uso totales

Tener en cuenta lo siguiente cuando se utiliza esta técnica de generación de pruebas a partir de casos de uso:

 Solo con las pruebas de casos de uso no se puede decidir la calidad del software. • Incluso si es un tipo de prueba de extremo a extremo, no garantizará la cobertura completa de la aplicación del usuario.

• Los defectos pueden ser descubiertos posteriormente durante las pruebas de integración.

PRUFBAS

CICLO DE VIDA DE LAS PRUEBAS **DE SOFTWARE (STLC)**

Planificación Implementación Seguimiento y control Ejecución **Análisis** Conclusión Diseño

Planificación: se definen los objetivos y el enfoque de la prueba dentro de las restricciones impuestas por el contexto. Algunas subactividades realizadas son: Determinar el alcance, los objetivos y los riesgos. Definir el enfoque y estrategia general. Definir las especificaciones de técnicas. Documentos de salida: Plan de prueba -general y/o por nivel de prueba-.

Seguimiento y control: El objetivo de esta actividad es reunir información y proporcionar retroalimentación y visibilidad sobre las actividades de prueba. Como parte del control, se pueden tomar acciones correctivas. Algunas subactividades realizadas son: Comprobar los resultados y los registros de la prueba en relación con los criterios de cobertura especificados. Documento de salida: Informe de avance de la prueba.

Análisis: se determina "qué probar". Algunas subactividades realizadas son: Analizar la base de prueba correspondiente al nivel de prueba, Identificar defectos de distintos tipos en las bases de prueba. **Documento de salida**: Contratos de prueba que contienen las condiciones de prueba.

Diseño: Durante esta actividad se determina "cómo probar". Algunas subactividades realizadas son: Diseñar y priorizar casos de prueba y conjuntos de casos de prueba de alto nivel. Identificar los datos de prueba necesarios. Documento de salida: Casos de prueba de alto nivel diseñados y priorizados.

Implementación: Se completan los productos de prueba necesarios para la ejecución de la prueba, incluyendo la secuenciación de los casos de prueba en procedimientos de prueba. Algunas subactividades realizadas son: Desarrollar y priorizar procedimientos de prueba. Crear juegos de prueba (test suite) a partir de los procedimientos de prueba. **Documento de salida:** Procedimientos y datos de prueba. Calendario de ejecución. Test suite.

Ejecución: se realiza la ejecución de los casos de prueba. Algunas subactividades realizadas son: Registrar los identificadores y las versiones de los elementos u objetos de prueba. Ejecutar y registrar el resultado de las pruebas de forma manual o utilizando herramientas. **Documento de salida:** Reporte de defectos. Informe de ejecución de pruebas.

Conclusión: Algunas subactividades realizadas son: Comprobar que todos los informes de defecto están cerrados. Finalizar, archivar y almacenar el entorno de prueba, los datos de prueba, la infraestructura de prueba y otros productos de prueba para su posterior reutilización. **Documento de salida**: Informe resumen de prueba. Lecciones aprendidas.

NIVELES DE PRUEBA (C.I.S.A)

Cada nivel de prueba es una instancia del proceso de prueba, desde componentes individuales hasta sistemas completos.



PRUEBA UNITARIA O DE COMPONENTE

Una prueba unitaria es una forma de comprobar el correcto funcionamiento de una unidad de código. Por ejemplo en diseño estructurado o en diseño funcional una función o un procedimiento, en diseño orientado a objetos una clase. Esto sirve para asegurar que cada unidad funcione correctamente y eficientemente por separado.

OBJETIVOS ESPECÍFICOS

Reducir el riesgo. Verificar que los comportamientos funcionales y no funcionales del componente son los diseñados y especificados.

Generar confianza en la calidad del componente. Encontrar defectos en el componente. Prevenir la propagación de defectos a niveles de prueba superiores

BASES DE PRUEBA:

Diseño detallado. Código. Modelo de datos. Especificaciones de los componentes.

OBJETO DE PRUEBA:

Componentes, unidades o módulos. Código y estructuras de datos. Clases. Módulos de base de datos

Funcionamiento incorrecto. Problemas de flujo de datos. Código y lógica incorrectos.

DEFECTOS Y FALLOS CARACTERÍSTICOS:

ENFOQUES Y RESPONSABILIDADES ESPECÍFICAS: En general el desarrollador que escribió el código realiza la prueba de componente.

PRUEBA DE INTEGRACIÓN

Se centra en las interacciones entre componentes o sistemas.

OBJETIVOS ESPECÍFICOS: Reducir el riesgo. Verificar que los comportamientos funcionales y no funcionales de las interfaces sean diseñados

y especificados. Generar confianza en la calidad de las interfaces. Encontrar defectos (en las interfaces o componentes o sistemas). Prevenir la propagación de defectos a niveles superiores.

BASES DE PRUEBA:

Diseño de software y sistemas. Diagramas de secuencia. Especificaciones de interfaz y protocolos de comunicación. Casos de uso. Arquitectura a nivel de componente o sistema. Flujos de trabajo. Definiciones de interfaces externas.

OBJETO DE PRUEBA:

Subsistemas. Bases de datos. Infraestructura. Interfaces. Interfaces de programación de aplicaciones (API). Microservicios.

DEFECTOS Y FALLOS CARACTERÍSTICOS:

Datos incorrectos, datos faltantes o codificación incorrecta de datos. Secuenciación o sincronización incorrecta de las llamadas a la interfaz. Incompatibilidad de la interfaz. Fallos en la comunicación entre componentes. Fallos de comunicación entre componentes no tratados o tratados de forma incorrecta. Suposiciones incorrectas sobre el significado, las unidades o las fronteras de los datos que se transmiten entre componentes.

ENFOQUES Y RESPONSABILIDADES ESPECÍFICAS:

La prueba de integración debe concentrarse en la integración propiamente dicha. Se puede utilizar tipos de prueba funcional, no funcional y estructural. En general es responsabilidad de los testers

PRUEBA DE SISTEMA

OBJETIVOS ESPECÍFICOS

Reducir el riesgo. Verificar que los comportamientos funcionales y no funcionales del sistema son los diseñados y especificados. Validad que el sistema está completo y funcionará como se espera. Generar confianza en la calidad del sistema considerado como un todo. Encontrar defectos. Prevenir la propagación de defectos a niveles de prueba superiores o a producción.

riesgo. Épicas e historias de usuario. Modelos de comportamiento del sistema. Diagramas de estado. Manuales del sistema y del usuario.

Especificaciones de requisitos del sistema y del software, funcionales y no funcionales. Informes de análisis de

Aplicaciones. Sistemas hardware/software. Sistemas operativos. Sistema sujeto a prueba (SSP). Configuración del

sistema y datos de configuración. DEFECTOS Y FALLOS CARACTERÍSTICOS:

Cálculos incorrectos. Comportamiento funcional o no funcional del sistema incorrecto o inesperado. Control y/o flujos de datos incorrectos dentro del sistema. Incapacidad para llevar a cabo de forma adecuada y completa, las tareas funcionales de extremo a extremo. Fallo del sistema para operar correctamente en los entornos de producción. Fallo del sistema para funcionar como se describe en los manuales de sistema y de usuario.

ENFOQUES Y RESPONSABILIDADES ESPECÍFICAS:

Debe centrarse en el comportamiento global de extremo a extremo del sistema en su conjunto, tanto funcional como no funcional. Deben usarse las técnicas más apropiadas para los aspectos del sistema que serán probados. Los testers independientes en general llevan a cabo la prueba de sistema.

PRUEBA DE ACEPTACIÓN

Al igual que la de sistema, se centra en el comportamiento y las capacidades de todo un sistema o producto.

OBJETIVOS ESPECÍFICOS:

Establecer confianza en la calidad del sistema en su conjunto. Validar que el sistema esté completo y que funcionará como se espera. Verificar que los comportamientos funcionales y no funcionales del sistema sean los especificados.

Procesos de negocio. Requisitos de usuario o de negocio. Normativas, contratos legales y estándares. Casos de uso. Requisitos de sistema. Documentación del sistema o del usuario. Procedimientos de instalación. Informes de análisis de riesgo.

OBJETO DE PRUEBA:

Sistema sujeto a prueba. Configuración del sistema y datos de configuración. Procesos de negocio para un sistema totalmente integrado. Sistemas de recuperación y sitios criticos, para pruebas de continuidad del negocio y recuperación de desastres. Procesos operativos y de mantenimiento. Formularios. Informes. Datos de producción existentes y transformados.

DEFECTOS Y FALLOS CARACTERÍSTICOS:

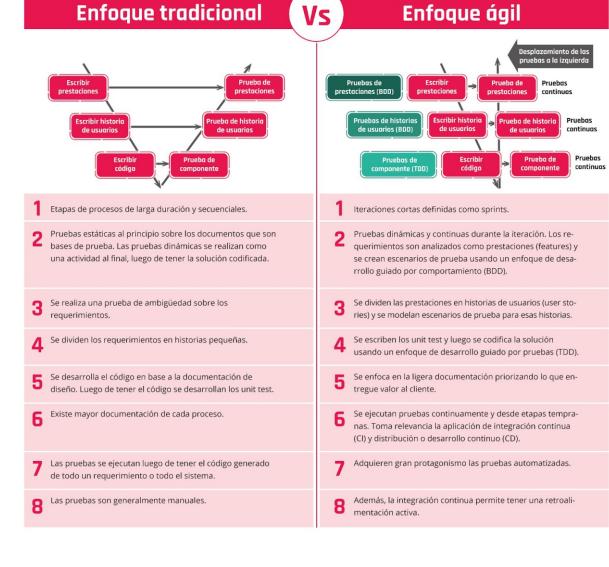
Los flujos de trabajo del sistema no cumplen con los requisitos de negocio o de usuario. Las reglas de negocio no se implementan de forma correcta. El sistema no satisface los requisitos contractuales o reglamentarios. Fallos no funcionales, como vulnerabilidades de seguridad o funcionamiento inadecuado en una plataforma soportada.

ENFOQUES Y RESPONSABILIDADES ESPECÍFICAS:

Es responsabilidad de los clientes, usuarios de negocio, propietarios de producto u operadores de un sistema. La prueba de aceptación se considera como el ultimo nivel de prueba en un ciclo de vida de desarrollo secuencial.

IMPLEMENTACIÓN

Desde el análisis a la implementación



MODELO EN V



EJECUCIÓN DE LAS PRUEBAS

Al completarse la implementación de los cambios o nuevas funcionalidades solicitados por el cliente, y ya diseñados los casos de pruebas correspondientes, estamos en la etapa de ejecutar las pruebas como parte de la **fase de pruebas del ciclo** de desarrollo de software (SDLC)

El objetivo de la implementación de las pruebas es asegurar que se cumplen los requerimientos del usuario, comprobando que los resultados obtenidos coinciden con los esperados, al mismo tiempo que se identifican y reportan los defectos encontrados. Las tareas dentro de la ejecución se llevan a cabo en forma iterativa hasta conseguir un sistema lo más estable posible.

Durante la ejecución de las pruebas, los conjuntos de pruebas se ejecutan luego del despliegue de cambios en los ambientes de prueba como parte del desarrollo planificado dentro de un sprint. La ejecución de pruebas incluye las siguientes actividades principales:

- 1. Registrar los identificadores y las versiones de los elementos u objetos de prueba, las herramientas de prueba y los productos de prueba.
- 2. Ejecutar pruebas de forma manual o utilizando herramientas de ejecución de
- pruebas 3. Comparar resultados reales con resultados esperados.
- 4. Analizar las anomalías para establecer sus causas probables. 5. Informar sobre los defectos en función de los fallos observados.
- 6. Registrar el resultado de la ejecución de la prueba. 7. Repetir las actividades de prueba, ya sea como resultado de una acción

tomada para una anomalía o como parte de la prueba planificada.

SUITES DE CASOS DE PRUEBA Prueba de humo o smoke test:

Cubren las funcionalidades principales de un sistema. Su objetivo es verificar la estabilidad de la aplicación para continuar con pruebas mas exhaustivas. Las pruebas de humo se ejecutan en las etapas iniciales del ciclo de vida de desarrollo de software, cada vez que los desarrolladores entregan una nueva versión al equipo de testing. Controlan de forma sencilla que funcionen los flujos

Pruebas de regresión:

criticos.

Son mas detalladas. Nos permiten asegurar que despues de cualquier mejora, actualización o cambio en el código, las interfaces, componentes y sistemas existentes no sufran daños. Son suites mas completas. Las pruebas de regresión tienen la tarea de verificar y validar las funcionalidades existentes de la aplicación después de cada modificación o en la adición de nuevas funciones.

Las pruebas de humo son previas a las de regresión. Si se encuentra algún problema durante las de humo, la compilación no se encuentra estable por lo que retorna al equipo de desarrollo hasta que lo sea. Una vez que nos encontramos en una versión estable del sistema, se llevan a cabo las pruebas de regresión sobre las funcionalidades existentes de forma exhaustiva.

TIPOS DE PRUEBAS

Un tipo de prueba es un grupo de actividades de pruebas destinadas a probar las características específicas de un sistema de software, o de una parte de un sistema, basados en objetivos de pruebas específicas. Dichos **objetivos** pueden incluir:

1. Evaluar las características de calidad funcional tales como la completitud, corrección y

2. Evaluar características no funcionales de calidad, tales como la fiabilidad, eficiencia de desempeño, seguridad, confiabilidad y usabilidad. 3. Evaluar si la estructura o arquitectura del componente o sistema es correcta, completa y según lo

4. Evaluar los efectos de los cambios, tales como confirmar que los defectos han sido corregidos (prueba de confirmación) y buscar cambios no deseados en el comportamiento que resulten de los cambios en el software o en el entorno (prueba de regresión)

PRUEBA FUNCIONAL

DEFINICIÓN

IMPLEMENTACIÓN

Pruebas que evalúan las funciones que el sistema debe realizar. Las funciones describen qué hace

La prueba funcional observa el comportamiento del software.

NIVELES DE PRUEBA Se pueden realizar en TODOS los niveles de prueba

Los requisitos funcionales pueden estar detallados en documentos como: especificaciones de requisitos del negocio, épicas, historias de usuarios, casos de uso y/o especificaciones

La cobertura funcional es la medida en que algún tipo de elemento funcional ha sido practicado

por pruebas. Se expresa como un porcentaje del tipo o tipos de elementos cubiertos.

2 PRUEBA NO FUNCIONAL

DEFINICIÓN

Prueba "cómo de bien" se comporta el sistema.

Su diseño y ejecución puede implicar competencias y conocimientos especiales, como el conocimiento de las debilidades inherentes a un diseño. Por ej: vulnerabilidades de seguridad asociadas con determinados lenguajes de programación.

NIVELES DE PRUEBA

IMPLEMENTACIÓN

Pueden realizarse en **todos** los niveles de prueba.

Evalúa características como la usabilidad, la eficiencia de desempeño o la seguridad.

ALCANCE

COBERTURA Es la medida en que algún tipo de elemento no funcional ha sido practicado por pruebas. Se expresa como un porcentaje del tipo de elemento cubierto.

3 PRUEBAS ESTRUCTURALES

DEFINICIÓN Están basadas en la estructura interna del sistema o en su implementación. Dicha estructura puede incluir código, arquitectura, flujos de trabajo y/o flujos de datos dentro del sistema.

IMPLEMENTACIÓN Su diseño e implementación pueden implicar conocimientos especiales como la forma en que se

construye el código, cómo se almacenan los datos y como utilizar las herramientas de cobertura e interpretar correctamente los resultados.

NIVELES DE PRUEBA

Se puede realizar en nivel de **componente** y de **integración**.

ALCANCE

COBERTURA

En el nivel de integración de componentes, la prueba puede basarse en la arquitectura del sistema, como las interfaces entre componentes.

Es la medida en que algun tipo de elemento estructural ha sido practicado mediante pruebas y se expresa como un porcentaje del tipo de elemento cubierto.

4 PRUEBA ASOCIADA AL CAMBIO

DEFINICIÓN Hay dos tipos:

Prueba de confirmación: su objetivo es confirmar que el defecto original se ha solucionado. Una vez corregido el defecto, se vuelven a probar en la nueva versión de software, los casos de prueba que fallaron debido al defecto. Prueba de regresión: un cambio hecho en una parte del código, puede afectar accidentalmente el

comportamiento de otras partes, dentro del mismo componentes o en otros. La prueba de

Sobre todo en los ciclos de vida ágiles, las nuevas características, los cambios en las

regresión busca detectar estos efectos secundarios no deseados.

características existentes y la refactorización del código, dan como resultado cambios frecuentes en el código, lo que también requiere pruebas asociadas al cambio.

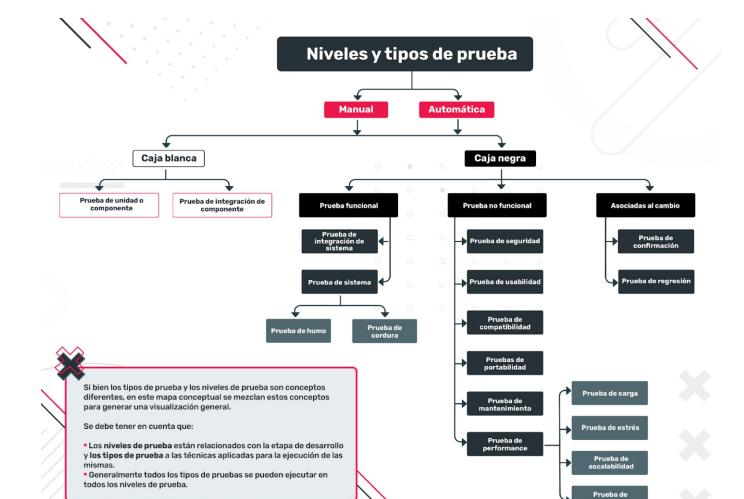
IMPLEMENTACION

NIVELES DE PRUEBA Todos los niveles de prueba.

COBERTURA

Los juegos de prueba de regresión se ejecutan muchas veces y suelen evolucionar lentamente, por lo que la prueba de regresión es un fuerte candidato para la **automatización**. La cobertura crece a

medida que se agregan mas funcionales al sistema, y por lo tanto mas pruebas de regresión.



DEFINICIONES

El objetivo de una técnica de prueba es ayudar a identificar las condiciones, los casos y los datos de prueba.

Elección de una técnica de prueba

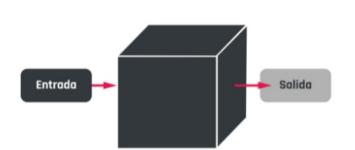
La elección de la técnica de prueba a utilizar depende

- de los siguientes factores:Tipo y complejidad del componente o sistema
- Estándares de regulación
- Requisitos del cliente o contractuales
- Clases y niveles de riesgo
 Objetivo de la prueba
- Documentación disponibleConocimientos y competencias del probador
- Modelo del ciclo de vida del softwareTiempo y presupuesto

TÉCNICAS DE CAJA NEGRA

Se basan en el comportamiento extraído del análisis de los documentos que son base de prueba (documentos de requisitos formales, casos de uso, historias de usuario, etc). Son aplicables tanto para pruebas funcionales como no funcionales. Se concentran en las entradas y salidas sin tener en cuenta la estructura interna.

- Las condiciones de prueba, casos de prueba y datos de prueba, se deducen de una base de prueba que puede incluir requisitos de software, especificaciones, casos de uso e historias de uso.
- Los casos de prueba se pueden utilizar para detectar diferencias entre los requisitos y su implementación, asi como desviaciones respecto a los requisitos
- La cobertura se mide en función de los elementos probados en la base de prueba y de la técnica aplicada a la base de prueba



Caja negra

1 PARTICIÓN DE EQUIVALENCIA

En esta técnica se dividen los datos en particiones conocidas como **clases de equivalencia** donde cada miembro de estas clases o particiones es procesado de la misma manera. Las características de esta técnica son:

1.La "partición de equivalencia **válida**" contiene valores que son aceptados por el componente o sistema.

2.La "partición de equivalencia **no válida**" contiene valores que son rechazados por

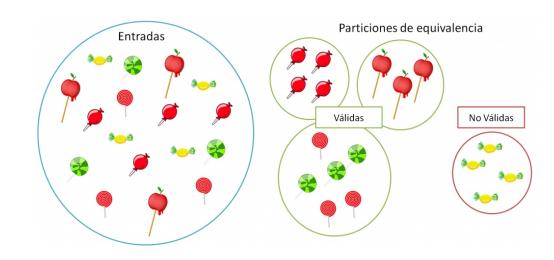
el componente o sistema. 3.Se pueden dividir las particiones en subparticiones.

3.Se pueden dividir las particiones en subparticiones.4.Cada valor pertenece a solo una partición de equivalencia.

5.Las particiones de equivalencia no válidas deben probarse en forma individual para evitar el enmascaramiento de fallos.

6. La cobertura se mide de la siguiente manera:

Cobertura = Particiones probadas Particiones identificadas



2 ANÁLISIS DE VALORES LÍMITES

Es una extensión de la técnica de partición de equivalencia que solo se puede usar cuando **la partición está ordenada**, y consiste en **datos numéricos o secuenciales**.

- Se deben identificar los valores límites mínimo y máximo (o valores inicial y
- tinal).

 Se pueden utilizar 2 o 3 valores límite
- Se pueden utilizar 2 o 3 valores límites.
 Para 2 valores límites se toma el valor que marca el límite (como valor que corresponde a la partición válida), y el valor anterior o posterior que
- corresponda a la partición de equivalencia inválida.

 Para 3 valores límites se toma el valor que marca el límite, un valor anterior y
- otro posterior a ese límite.La cobertura se mide de la siguiente manera:

Cobertura = Valores límites probados Valores límites identificados



3 TABLA DE DECISIÓN

Esta técnica se utiliza para **pruebas combinatorias**, formadas por reglas de negocio complejas que un sistema debe implementar. Las características de esta técnica son:

- Se deben identificar las condiciones (entradas) y las acciones resultantes
- (salidas). Estas conforman las filas de la tabla.
 Las columnas de la tabla corresponden a reglas de decisión. Cada columna forma una combinación única de condiciones y la ejecución de acciones asociadas a esa regla.
- Los valores de las condiciones y acciones pueden ser valores booleanos,
 discretos numéricos o intervalos de números
- discretos, numéricos o intervalos de números.

 Ayuda a identificar todas las combinaciones importantes de condiciones y a
- encontrar cualquier desfase en los requisitos.La cobertura se mide de la siguiente manera:

Cobertura = Número de reglas de decisión probadas Número de reglas de decisión totales

_									
	Condiciones	Reglas							
	Condición 1	S	S	S	S	N	N	N	N
	Condición 2	S	S	N	N	S	S	N	N
	Condición 3	S	N	S	N	S	N	S	N
	Acción 1	Х	Х						
	Acción 2				X		X		X
	Acción 3			X				X	
	Acción 4					X			

4 TRANSICIÓN DE ESTADOS

Un diagrama de transición de estado muestra los **posibles estados del software**, así como la forma en que el software entra, sale y **realiza las transiciones** entre estados. Las características de esta técnica son:

- Una tabla de transición de estado 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.
 La prueba de transición de estado se utiliza para aplicaciones basadas en menús y es extensamente utilizada en la industria del software embebido. La técnica también es adecuada para modelar un escenario de negocio con
- estados específicos o para probar la navegación en pantalla.

 La cobertura se mide de la siguiente manera:

Cobertura = Número de estados o transiciones identificados probados

Número total de estados o transiciones identificados en el objeto de prueba

Con esa información se valida que se pueda asignar a un cliente los distintos estados civiles como se

muestran en el flujo, desde el inicio hacia el final (end-to-end).

• En este caso tendremos como mínimo seis casos de pruebas:

Estado 1	Estado 2	Estado 3	Estado 4	Estado 5	Estado Final
nacido	soltero	con hijo/s	80		con hijo/s
nacido	soltero	casado	con hijo/s		con hijo/s
nacido	soltero	casado	viudo	con hijo/s	con hijo/s
nacido	soltero	casado	viudo	casado	casado
nacido	soltero	casado	divorciado	con hijo/s	con hijo/s
10.	1,000,000				

nacido soltero casado divorciado casado

CLASIFICACIÓN

TÉCNICAS DE CAJA BLANCA

Se basan en la estructura extraída de los documentos de arquitectura, diseño detallado, estructura interna o código del sistema. Se concentran en el procesamiento dentro del objeto de prueba.

- Las condiciones de la prueba, casos y datos de prueba se deducen de una base de prueba que puede incluir código, arquitectura de software, diseño detallado o
- cualquier otra fuente de info relacionada a la estructura de software.
 La cobertura se mide en base a los elementos probados dentro de la estructura
- seleccionada (ej, el código o las interfaces).
 Las especificaciones se utilizan a menudo como fuente adicional de información para detectar el resultado esperado de los casos de prueba.



- 1 PRUEBA Y COBERTURA DE SENTENCIA.
- PRUEBA Y COBERTURA DE DECISIÓN.

TÉCNICAS BASADAS EN LA EXPERIENCIA

aprovechan el conocimiento de desarrolladores, probadores y usuarios para diseñar, implementar y ejecutar las pruebas.

PREDICCIÓN DE ERRORES

Esta técnica se utiliza para **anticipar la ocurrencia de equivocaciones, defectos y fallos** basados en el conocimiento del probador. Se crea una lista teniendo en

- cuenta:Cómo ha funcionado la aplicación en el pasado.
- Equivocaciones comunes en los desarrolladores.
- Fallos en aplicaciones relacionadas.
 En base a esa lista se diseñan pruebas que expongan esos fallos y defectos.

2 PRUEBA EXPLORATORIA

En esta técnica se diseñan, ejecutan, registran y evalúan de forma dinámica pruebas informales durante la ejecución de la prueba. Los resultados de estas pruebas se utilizan para aprender más sobre el funcionamiento del componente o sistema. Generalmente se utilizan para complementar otras técnicas formales o cuando las especificaciones son escasas, inadecuadas o con restricciones de tiempo.

3 PRUEBAS BASADAS EN LISTAS DE COMPROBACIÓN

En esta técnica se diseñan, implementan y ejecutan casos de prueba que cubren las condiciones que se encuentran en una lista de comprobación definida. Se crean basadas en la experiencia y conocimiento de lo que el probador cree que es importante para el usuario y se utilizan debido a la falta de casos de prueba detallados. Durante la ejecución puede haber cierta variabilidad, dependiendo de quién ejecuta la prueba y condiciones del contexto. Esto da lugar a una mayor cobertura. Se utiliza tanto en pruebas funcionales como no funcionales.

CAJA NEGRA VS CAJA BLANCA

AJA NLU	NA VO	CAJA	DLANGA

Criterio de Comparación	Caja Negra	Caja Blanca
Base De Prueba	 Requisitos de software Especificaciones, Casos de uso Historias de usuario. 	 Arquitectura de software Diseño detallado de software Cualquier otra fuente de información relacionada a la estructura de software.
Uso de Especificaciones de Software	Son una fuente principal de consulta para el diseño de estos casos de prueba.	Se utilizan a menudo como una fuente adicional de información para detectar el resultado esperado de los casos de prueba.
Cobertura	Se mide en función de los elementos probados en la base de prueba y de la técnica aplicada a la base de prueba.	Se mide en base a los elementos probados dentro de la estructura seleccionada (por ejemplo, el código o las interfaces).