FUNDAMENTOS DE TESTING

PRINCIPIOS DE LAS PRUEBAS

¿QUÉ SON LAS PRUEBAS?

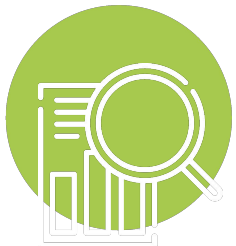
Es el proceso de evaluar un producto, aprendiendo a través de la exploración y experimentación, lo cual incluye: Cuestionar, estudiar, modelar, observar e inferir, checar salidas de datos, etc

¿Pasando todas las pruebas tendré un software sin errores?

Apple aseguraba que esta nueva tecnología, Face ID, "**falla apenas una vez en un millón**", cosa que no concuerda con la realidad, al menos no fue así durante la presentación oficial.

Hace 20 años Windows 98 debutó, y lo hacía con BSOD (Blue Screen of Death) incluida delante de Bill Gates

Uber despide a su jefe de seguridad tras reconocer el robo de datos de **57 millones de clientes**



No son parte de la solución

No siempre se entrega la evidencia al cliente

Sin una estructura o especificaciones son difíciles de mantener

No hay tiempo, o nadie del equipo hace pruebas

Razones para hacer las pruebas

Tenemos un problema o el resultado no es el esperado

Costo alto o fuera de presupuesto

Implicaciones legales o de estándares tecnológicos

PROCESO DE PRUEBAS DE SOFTWARE

Metodología

Herramientas

Recursos

PRUEBAS EN EL CICLO DE VIDA DEL SOFTWARE

Hay pruebas desde la concepción de los requisitos hasta su puesta final en producción.

De acuerdo a un estudio de IBM Systems Sciences Institute, 64% de los errores se producen durante el análisis y diseño.

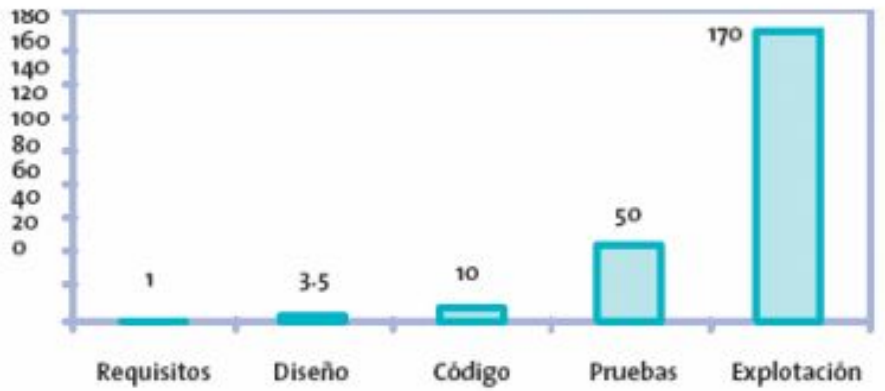
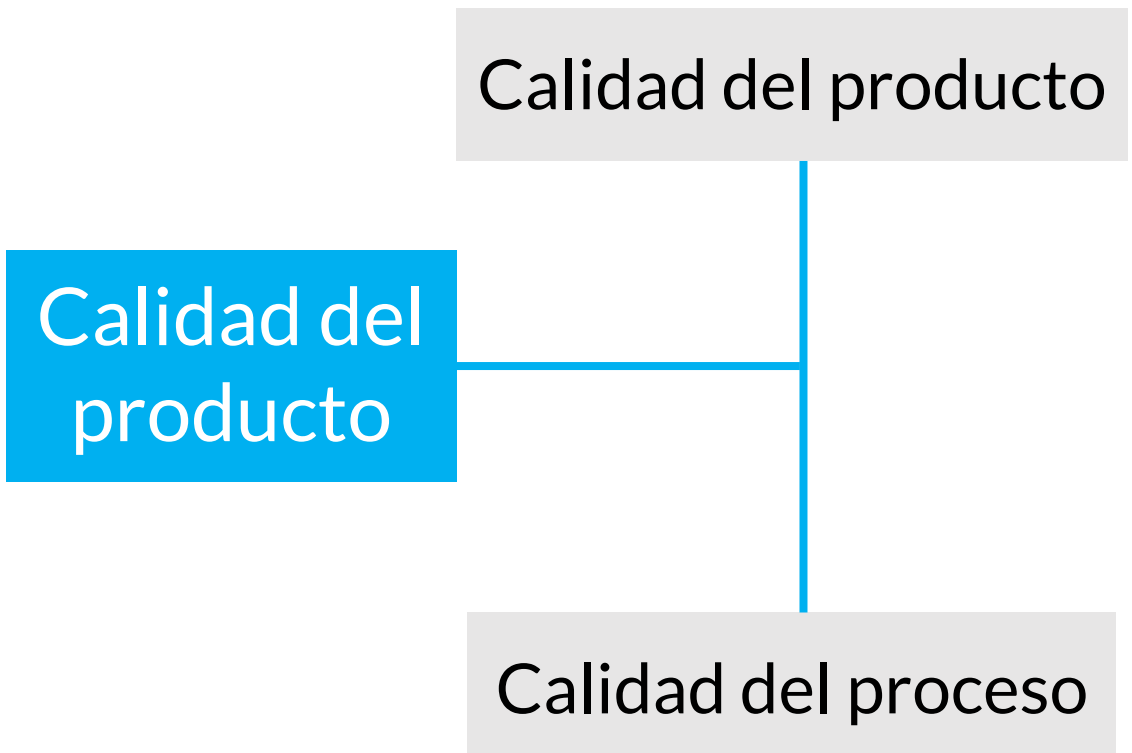


Figura 8-1.
Coste relativo de corrección de errores

Definir la falta de calidad

Detectar y corregir la falta de calidad

Calidad de software



Lo que la gente produce

- Requerimientos
- Diseño
- Código
- El sistema

Cómo lo hace la gente

- Estándares
- Procedimientos
- Procesos del proyecto

Certificaciones, estándares y metodologías para:

- Para individuos
- Para procesos
- Para empresas
- Para servicios/productos = software/hardware
- Para tipo de industrias

ESTÁNDARES

ISTQB

(International Software Testing Qualifications Board)

IEEE

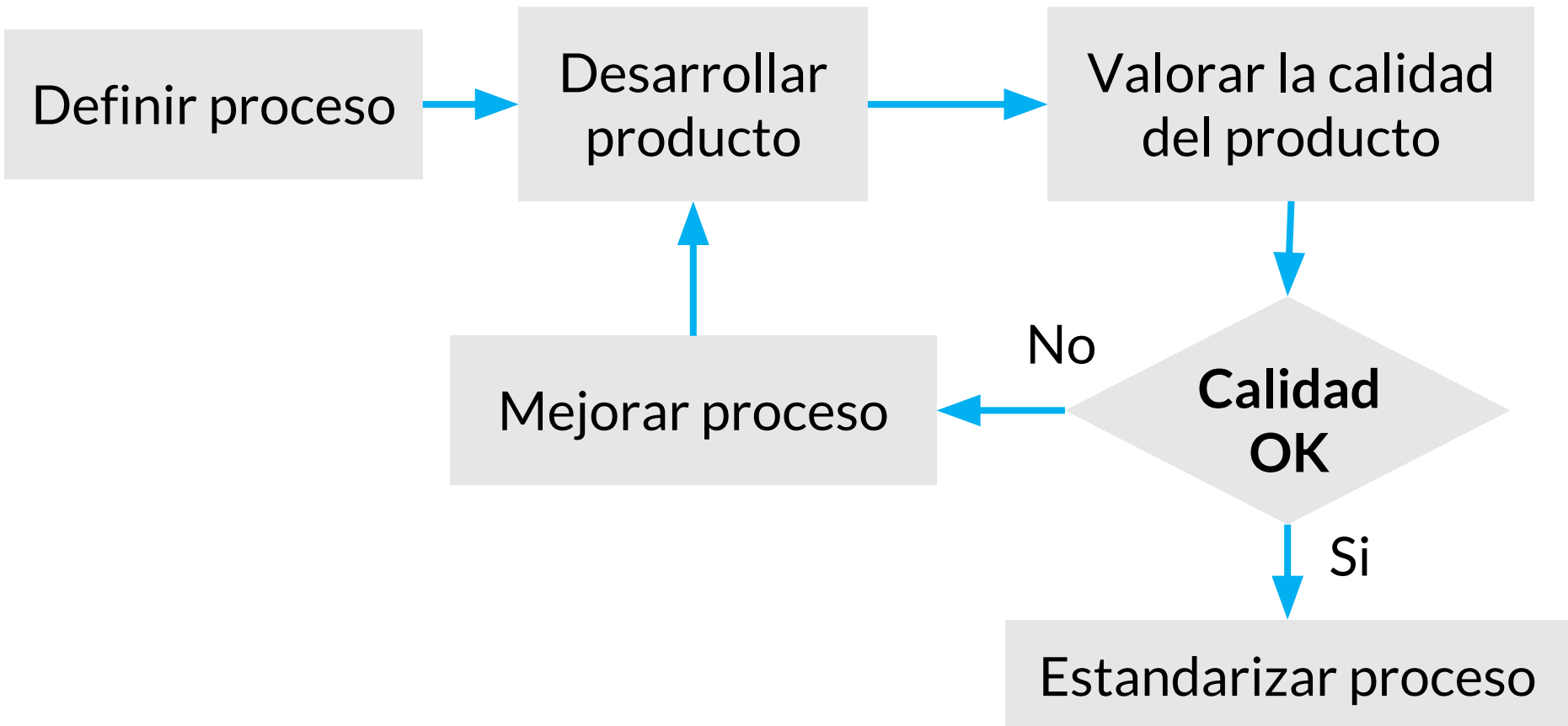
(Institute of Electrical and Electronics Engineers)

TPI

(Testing Process Improvement)

CALIDAD Y DEFECTOS

¿Qué es la calidad?



Calidad basada en procesos

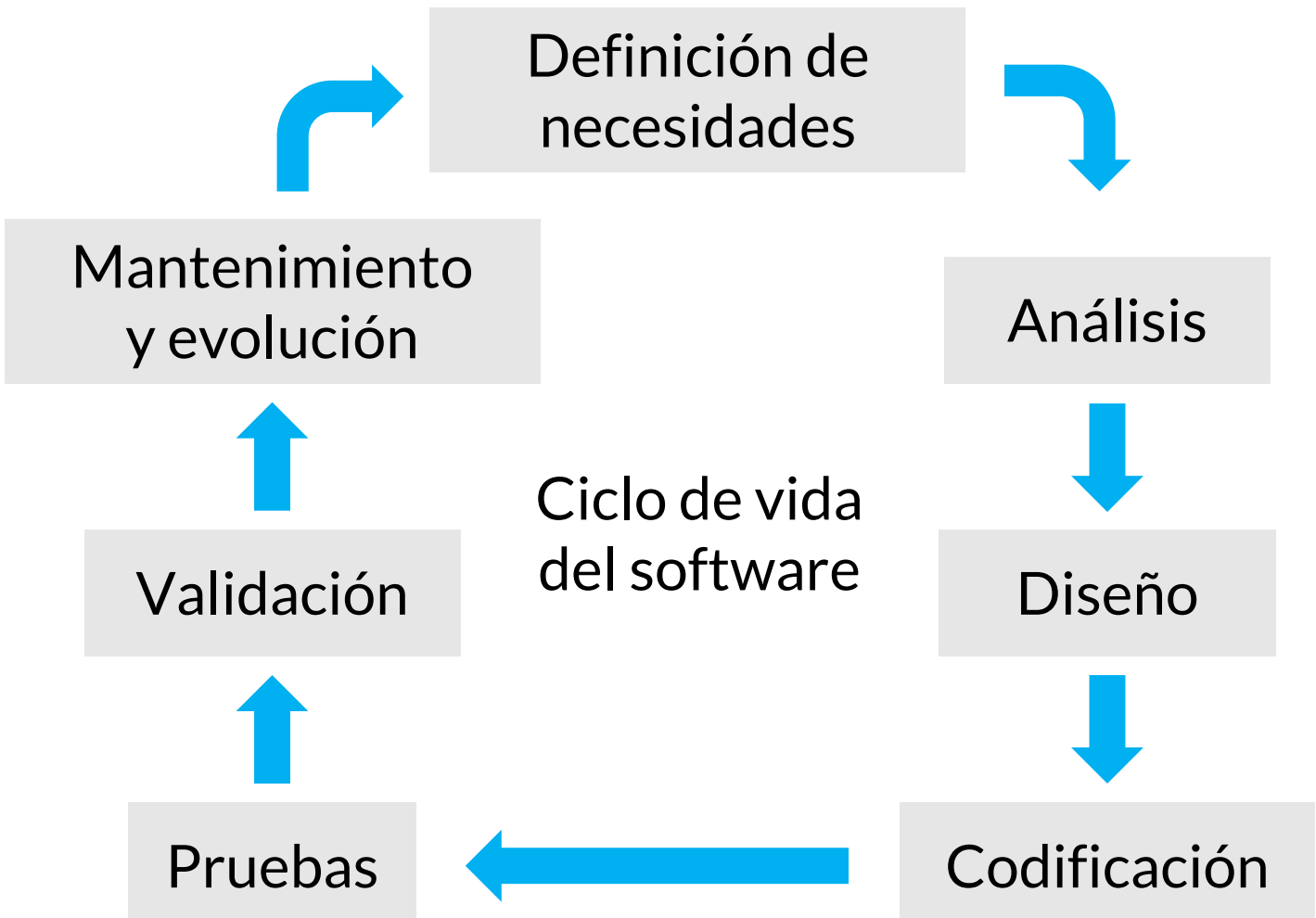
“

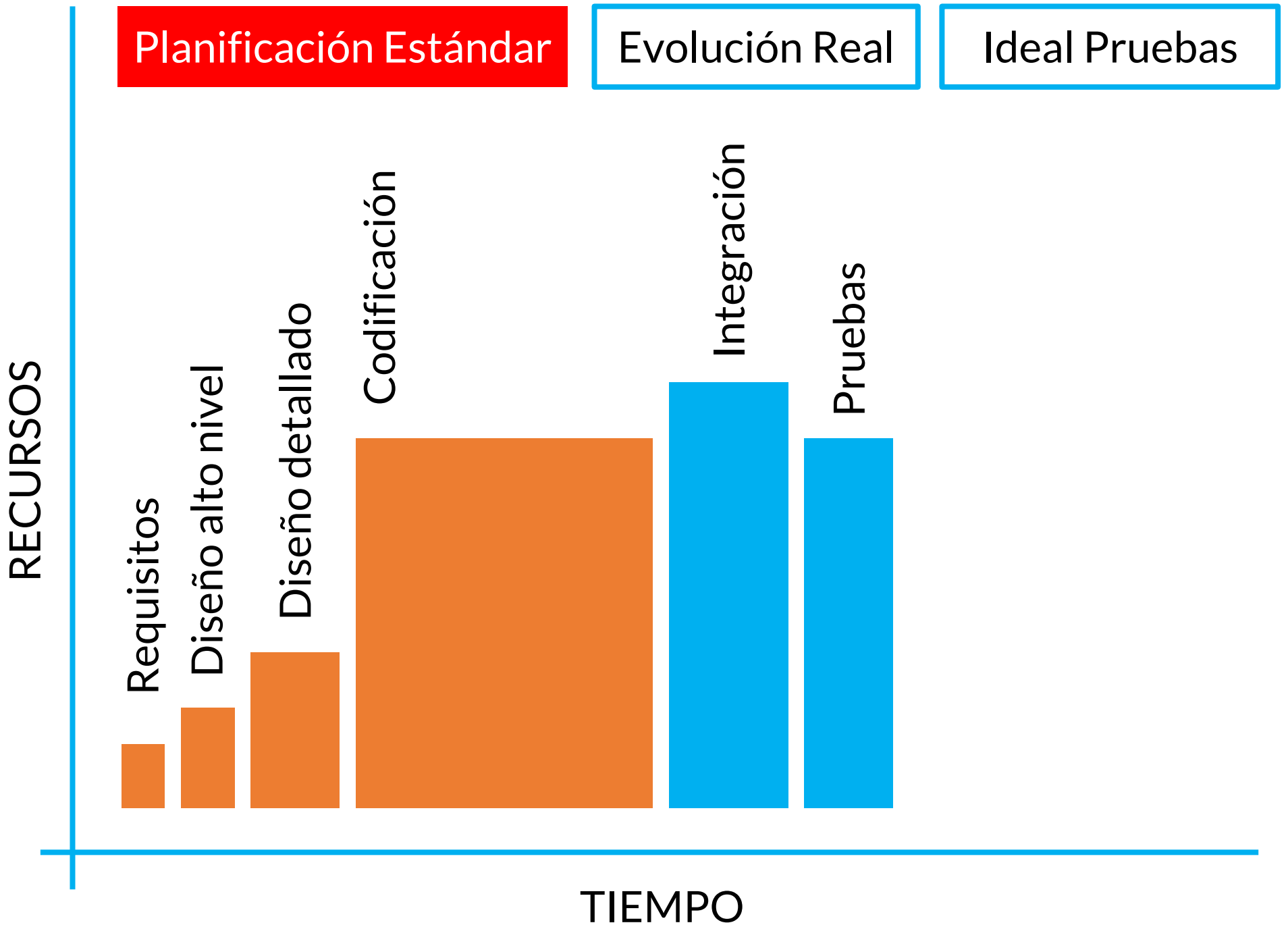
El grado con el que un sistema, componente o proceso cumple con los requisitos especificados y las necesidades o expectativas del cliente o usuario.

”

[IEEE.Std.610-1990]

PROCESO DE PRUEBAS DE SOFTWARE




Planificación Estándar
Evolución Real
Ideal Pruebas
RECURSOS
Requisitos
Diseño alto nivel
Diseño detallado
Codificación
Integración
Pruebas
TIEMPO

Verificación

Validación

ANOMALÍA
Cualquier insatisfactoria condición

DEFECTO
No desempeña funciones

FALLO

Incapacidad dentro de márgenes

ERROR
Acción humana incorrecta

TABLERO AUTOMOTRIZ

“

El **error** humano cometido inyecta un **defecto** en el software que, ocasionalmente, se observa como una **anomalía** a causa de un comportamiento incorrecto, no acorde a lo especificado, que finalmente provoca el **fallo** del sistema software

”

TESTING MODERNO

7 PRINCIPIOS DEL TESTING MODERNO

“Los testers podemos comenzar a pasar de ser los dueños de las pruebas o la calidad, a ser los embajadores de la calidad del producto”

1. Nuestra prioridad es mejorar el negocio

2. Nosotros aceleramos al equipo, usamos modelos como **Lean Thinking** y **Teoría de las Restricciones** para ayudar a identificar, priorizar y mitigar cuellos de botella en el sistema

3. Somos la fuerza para la mejora continua, ayudando al equipo a **adaptarse y optimizar** para tener éxito, en lugar de proporcionar una red de seguridad para detectar fallas

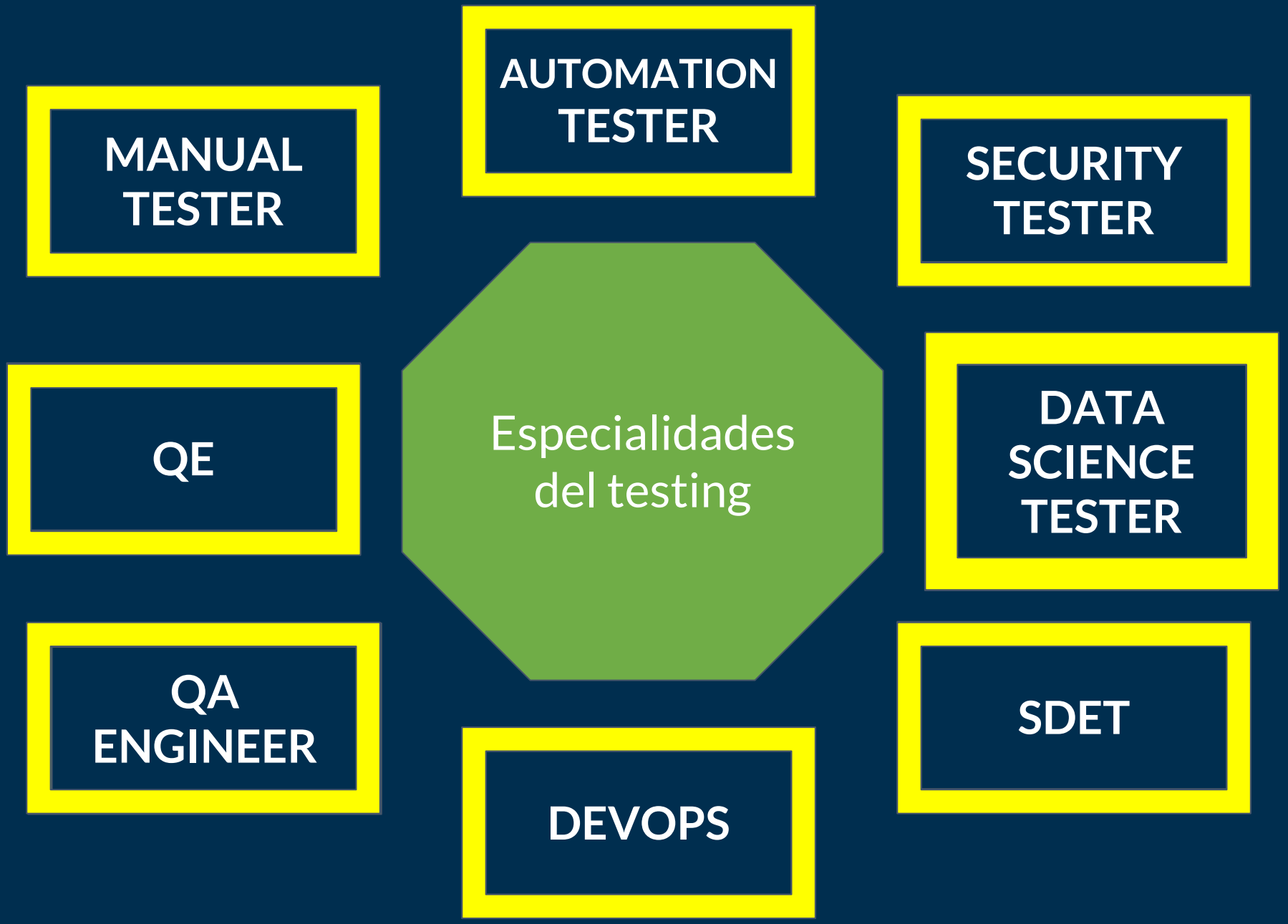
4. Nos preocupamos profundamente acerca de la **cultura de calidad** en el equipo, y asesoramos, lideramos y nutrimos el equipo para llevarlos a una cultura de calidad más madura.

5. Nosotros creemos que el cliente es el único capaz de **juzgar y evaluar** la calidad de nuestro producto

6. Nosotros usamos **datos de manera extensa y profunda** para entender los casos de uso del cliente y entonces cerrar huecos entre hipótesis del producto e impacto del negocio.

7. Expandimos las habilidades de testing y el conocimiento en **todo el equipo**; entendemos que esto reduce o elimina la necesidad de un especialista dedicado al testing.

LAS ESPECIALIDADES DE TESTING


AUTOMATION TESTER
MANUAL TESTER
SECURITY TESTER
Especialidades del testing
QE
DATA SCIENCE TESTER
QA ENGINEER
SDET
DEVOPS


PROGRAMACIÓN
PENSAMIENTO LATERAL
PROTOCOLOS Y ESTÁNDARES
Skills de testing
SOLUCIONES Y ESTRATEGIAS DE CALIDAD
ANÁLISIS Y LIMPIEZA DE DATOS
PROCESOS DE CALIDAD
INTEGRACIÓN CONTINUA
ENTREGA CONTINUA

TESTING DURANTE CICLO DE VIDA DEL DESARROLLO DE SOFTWARE

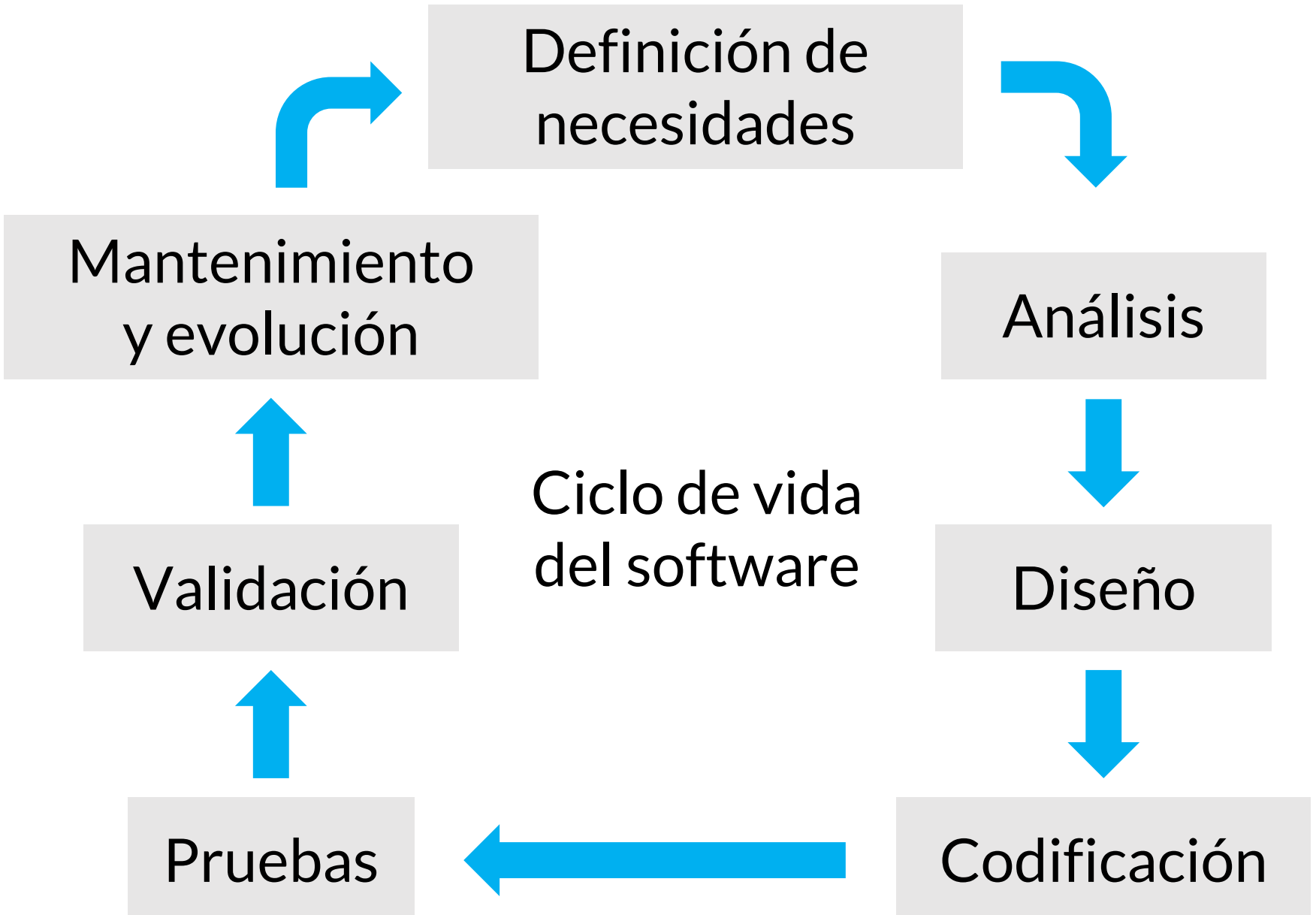
Construcción del Software

PRESUPUESTO

RECURSOS

TIEMPO

Actividades Clave de pruebas


Definición de necesidades
Análisis
Diseño
Codificación
Pruebas
Validación
Mantenimiento y evolución
Ciclo de vida del software

ESTRATEGIA DE PRUEBAS

¿Que problema tenemos actualmente?

¿O qué problemas debemos evitar?

Escenarios y Contextos

- Seguridad
- Arquitectura
- Performance
- Usabilidad
- Escalabilidad

TESTING EN DESARROLLO DE SOFTWARE

VS

TESTING **CHECKING**

ESTRATEGIAS DE CHECKING

- Solo se ejecutan si...
- Se ejecutan cada que...
- Se ejecutan de manera programada

ERRORES COMUNES DURANTE LA EJECUCIÓN

- **Pruebas duplicadas**
- **Pruebas similares**
- **Pruebas sin valor agregado**
- **Pruebas caducadas**

La automatización de pruebas consiste en el uso de **software especial** para controlar la ejecución de pruebas y la comparación entre los resultados obtenidos y los resultados esperados. Sin embargo, se trata de un **checking repetitivo y automatizado**.

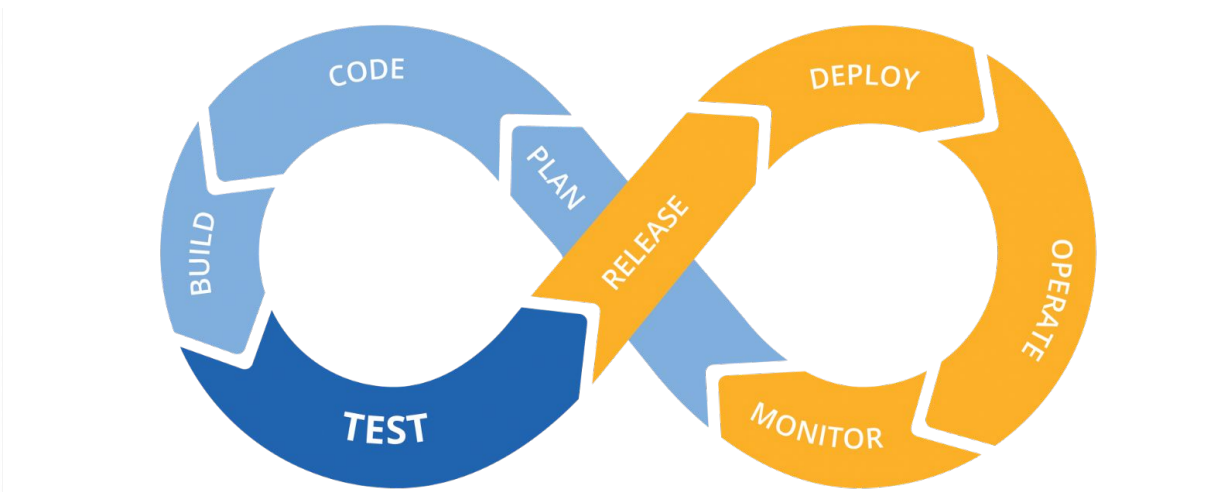
DESVENTAJAS CHECKING MAL EMPLEADO

- Pobre cobertura de pruebas
- Falta de actualización
- Mal manejo de versiones

VENTAJAS CHECKING BIEN EMPLEADO

- Correr pruebas en paralelo o en múltiples plataformas
- Reducción de error humano
- Probar grandes cantidades de datos

Por otro lado, cuando ya queremos hablar de **Integración continua y Liberación Continua**, entonces la **automatización** es la solución definitiva para la eficiencia del equipo de desarrollo digital y equipos DevOps



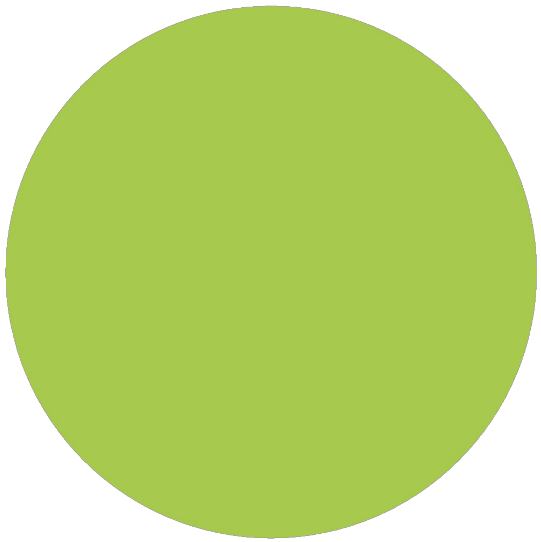
TESTING ÁGIL

Testing ágil involucra a todos los miembros de un equipo ágil multifuncional, en el cual el rol del tester es el de un experto multifuncional, que garantiza se entregue el valor de negocio deseado al cliente a un ritmo sostenible y continuo.

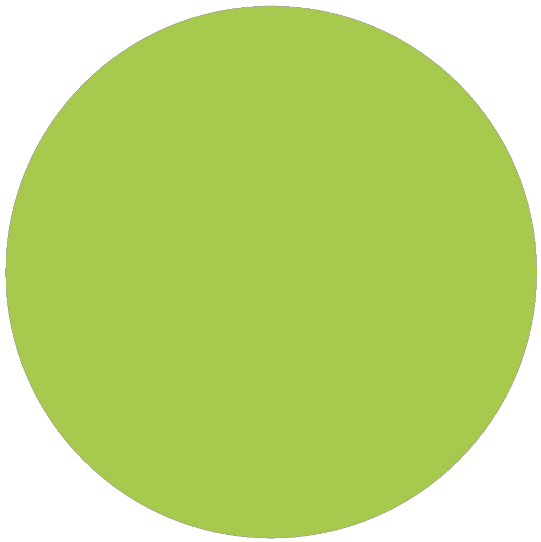
ESTRATEGIAS ÁGILES

- El Testing es de “todo el equipo”
- El Testing puede ser independiente (opcional)
- Integración continua
- Testing guiado por pruebas (Test Driven Development – TDD)
- Desarrollo guiado por comportamiento (Behaviour Driven Development – BDD)
- Desarrollo guiado por pruebas de aceptación (Acceptance Test Driven Development – ATDD)

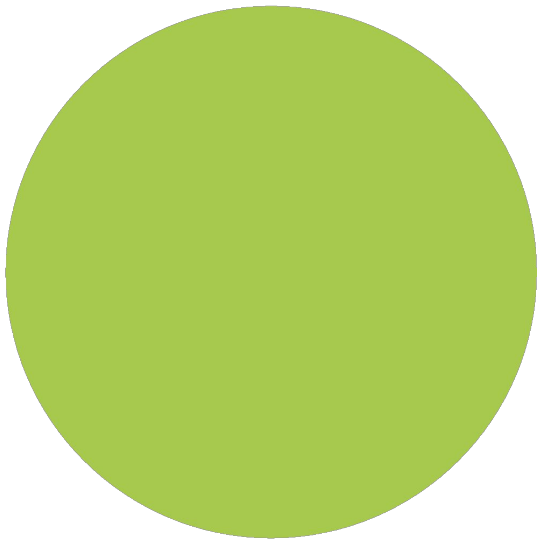
NIVELES DE PRUEBAS DE SOFTWARE



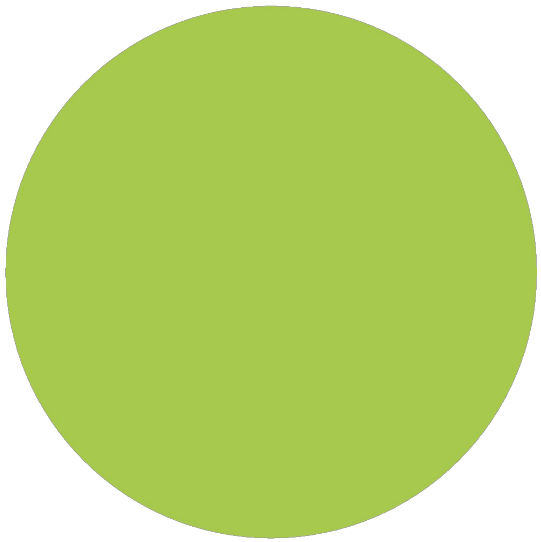
Prueba de Componentes



Prueba de Integración



Prueba de Sistema



Prueba de Aceptación

TIPOS DE PRUEBAS DE SOFTWARE

1. PRUEBAS FUNCIONALES

Se entiende como las Funcionalidades del Sistema cómo “lo que el sistema hace”.

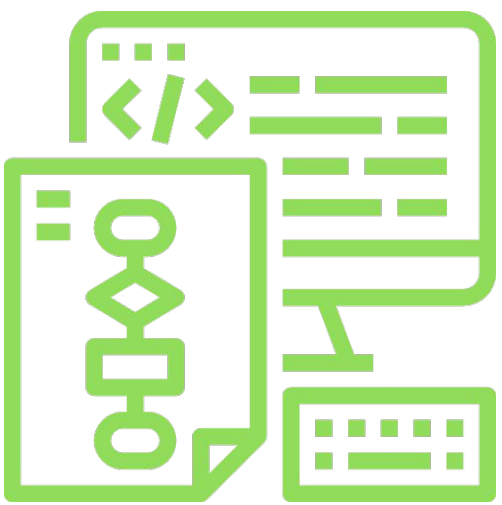
2. PRUEBAS NO-FUNCIONALES

El objetivo de esta es probar “como funciona el sistema”.

3. PRUEBAS ESTRUCTURALES

Para poder llevar a cabo estas pruebas, normalmente el tester debe tener conocimientos acerca de la tecnología y el stack que se está empleando

4. PRUEBAS DE MANEJO DE CAMBIOS



Es probar nuevamente un componente ya probado para verificar que no ha sido impactado por actualizaciones.

PRUEBAS ESTÁTICAS Y DINÁMICAS

“

Las pruebas dinámicas se enfocan principalmente en comportamientos externos visibles durante la ejecución del software

”

“

Las pruebas estáticas se basan en la examinación manual de los elementos que conforman la construcción del software

”

¿Qué son los elementos?

- Contratos, planes y calendario del proyecto, así como su presupuesto.
- El análisis de requerimientos
- Especificaciones o reglas de negocio
  - Técnicos
  - Seguridad

¿Qué son los elementos?

- Las definiciones de
  - Historias del usuario
  - Criterios de Aceptación
  - Mockups
- El diseño de la arquitectura
- Las pruebas (Testware), puntos de verificación CI
- Guías de usuario
- Evaluación/revisión del código

Beneficios

- Detectar y corregir defectos de manera más eficiente
- Identificar y priorizar la ejecución de pruebas en etapas posteriores
- Prevenir defectos
  - Que no son fácilmente detectables durante las pruebas dinámicas
  - Durante la etapa de análisis y diseño

Beneficios

- Cubrir aspectos como:
  - Inconsistencias, ambigüedades, contradicciones, definiciones inexactas, requerimientos redundantes
- Reducir el retrabajo e incrementar la productividad
- Reducir el costo y el tiempo
- Mejorando la comunicación entre todo los miembros del equipo

DEFINICIÓN Y DISEÑO DE PRUEBAS

¿Qué hace un tester?

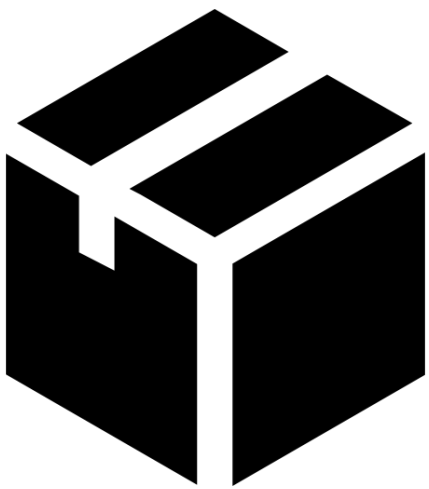
1. Encontrar problemas
2. Documentar problemas
3. Comunicar problemas

Si no encuentra problemas antes de que el producto sea entregado el cliente, entonces su testing es ineficiente

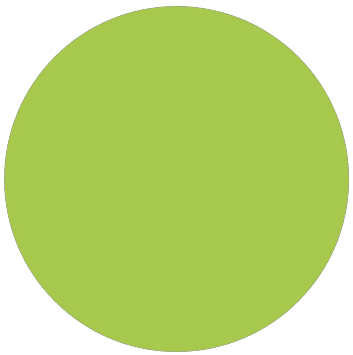
Si cuando encuentra problemas no sabe documentar y reproducir los pasos correctos su testing genera retrabajo y sube el costo

Si como representante de la calidad del producto no sabe argumentar y proteger los intereses del negocio o los clientes, entonces su testing no agrega valor.

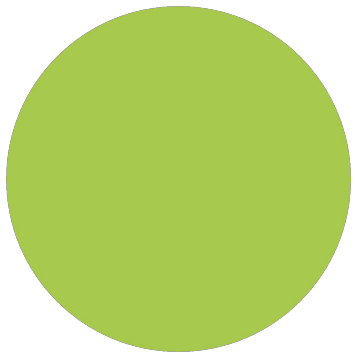
PRUEBAS DE CAJA:



- Negra
- Blanca
- Gris



CAJA NEGRA



CAJA BLANCA

TÉCNICAS DE PRUEBAS

CAJA NEGRA

- Partición de equivalencia
- Valores límite
- Tabla de decisiones
- Transición de estados
- Casos de uso

CAJA GRIS

- Casos de negocio
- Pruebas End to End
- Pruebas de integración

CAJA BLANCA

- Cobertura de declaración
- Cobertura de decisiones

EJERCICIOS:

Diseñar y ejecutar pruebas correspondiente a :

a. un sitio web
b. una aplicación móvil
c. una API
d. un caso de uso

GESTIÓN DE MONITOREO Y CONTROL

GESTIÓN DE PRUEBAS

- Planeación de pruebas
- Monitoreo y control de pruebas
- Análisis de pruebas
- Diseño de pruebas
- Implementación de pruebas
- Ejecución de pruebas
- Finalización de las pruebas

Planeación de pruebas

“Definición de Objetivos de las pruebas, alcance de las mismas, las técnicas de pruebas que se llevarán a cabo, junto con la estimación y definición de fechas de entrega, así como los criterios de salida”

Monitoreo y Control de pruebas

“Durante el Monitoreo se va midiendo y comparando los resultados de las métricas, y entonces durante el CONTROL se toman acciones para alcanzar el objetivo del plan y los criterios de salida.”

Análisis de pruebas

“Cuándo estamos analizando las pruebas para nuestro proyecto, necesitamos determinar qué debemos probar, obviamente basados en las prioridades de cobertura”

Diseño de pruebas

- Diseño de casos de alto nivel
- Diseñar y priorizar las pruebas
- Identificar los datos de pruebas
- Identificar el entorno de pruebas - infraestructura y herramientas
- Hacer una trazabilidad entre pruebas y sus condiciones

Implementación de pruebas

“Para poder prepararnos para hacer las pruebas, primero tenemos que asegurarnos que tenemos todo lo necesario para ello.”

Ejecución de pruebas

“Durante esta etapa, las suites de pruebas se ejecutan de acuerdo con el programa de ejecución de las pruebas.”

Finalización de pruebas

- Defectos con el estatus correcto
- Reporte para comunicar los resultados de las pruebas
- Finalizar y archivar ambiente de pruebas y sus datos
- Entregar el Testware al equipo de mantenimiento de pruebas
- Analizar lecciones aprendidas para futuras versiones
- Recopilar la información para ayudar a mejorar la madurez del proceso de prueba.

GESTIÓN DE PRUEBAS

- Planeación de pruebas
- Monitoreo y control de pruebas
- Análisis de pruebas
- Diseño de pruebas
- Implementación de pruebas
- Ejecución de pruebas
- Finalización de las pruebas

ROLES Y RESPONSABILIDADES

- Especialista en pruebas manuales
- Especialista en pruebas técnicas
- Líder del equipo de pruebas
- Ingeniero de calidad

1. Es el encargado de hacer coaching de calidad en toda la empresa

   A) ingeniero de calidad
   B) tester técnico

2. Es el responsable de emplear herramientas que automaticen o aceleren las ejecuciones de las pruebas

   A) ingeniero de calidad
   B) tester técnico

3. Es el responsable de asegurarse de la cobertura de pruebas y nuevos escenarios

   A) líder de pruebas
   B) tester manual

4. Es el facilitador del equipo que se asegura de implementar lo necesario para que se ejecuten las pruebas

   A) líder de pruebas
   B) tester manual

RETRABAJO

El dashboard puede ser una herramienta útil que mantiene informado a todo el equipo acerca del estatus de las pruebas

ACCIONES DE CONTROL

- Si identificamos un riesgo...
- Si identificamos falta de ambientes...
- Si el criterio de salida no se cumple...

Resultados de las pruebas

Desempeño del
equipo de testing

RETRABAJO

“

Esfuerzo adicional necesario para la corrección de una inconformidad en algún producto. El problema que surge con el retrabajo es obvio: es un esfuerzo adicional que no puede ser cobrado al cliente, pero que es necesario para que este quede conforme con lo que hemos hecho para él.

”

- Falta o mala documentación
- Falta de capacitación o dominio en las herramientas utilizadas
- Falta de capacitación o dominio en el software a desarrollar
- Falta de comunicación

GESTIÓN DE BUGS

Razones por las que aparecen defectos

- Hay presión de tiempo en la entrega del software
- Descuidos en el diseño
- Inexperiencia o falta de conocimiento
- Falta de comunicación en los requerimientos
- Diseño complejo de código
- Desconocimiento de las tecnologías usadas

¿Cómo crear un proceso de gestión de bugs?

- ¿Qué debe de hacer la persona que encuentre un defecto?
- ¿En qué herramienta debe documentar el defecto?
- ¿Cómo vamos a almacenar la información?
- ¿Qué información requiere el equipo de desarrolla para poder resolver un defecto?
- ¿Cuales son los estatus que se manejan para que fluya la resolución del defecto?
- ¿Cuales son los criterios de aceptación de cierre del defecto?

CICLO DE GESTIÓN

REPOSITORIO Y MONITOREO DE DEFECTOS

Una vez instaurado el proceso de gestión de bugs, también se debe precisar quién tiene acceso a los bugs y cuales son los permisos que tiene, por cuánto tiempo se almacenan, etc.

DEFECTOS Y SUGERENCIAS

DEFECTOS vs SUGERENCIAS

DEFECTOS

SUGERENCIAS

EJEMPLOS DE SUGERENCIAS

- **Ejemplo #1,** el mensaje de error no comunica adecuadamente
- **Ejemplo #2,** el color de la pantalla, no contrasta bien con el texto
- **Ejemplo #3,** no recibí un correo adicional de confirmación

“

Si la calidad la define el usuario final… sus sugerencias se vuelven defectos?

”

SUGERENCIAS CONVERTIDAS EN DEFECTOS / ACTUALIZACIONES DE SOFTWARE

- Hace lenta la operación
- Detiene parcial o totalmente el proceso
- El contenido o el flujo confunde al usuario
- Deja cometer muchos errores al usuario
- La traducción o el lenguaje empleado no es correcto
- No funciona sin internet

¿CÓMO REPORTAR UN DEFECTO/SUGERENCIA?

EJEMPLOS DE SUGERENCIAS

- **Ejemplo #1,** el mensaje de error no comunica adecuadamente
- **Ejemplo #2,** el color de la pantalla, no contrasta bien con el texto
- **Ejemplo #3,** no recibí un correo adicional de confirmación

SEGUIMIENTO Y CIERRE

DEPURACIÓN

“

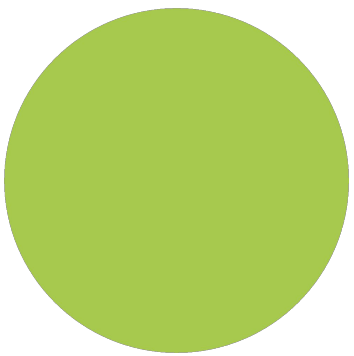
Uno de los principales problemas al desarrollar aplicaciones son los errores de ejecución

”

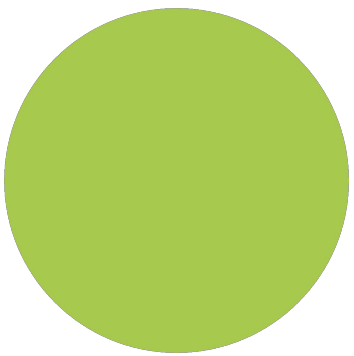
Depurando (Debugging)

Actividad que sirve para encontrar, analizar y arreglar defectos.

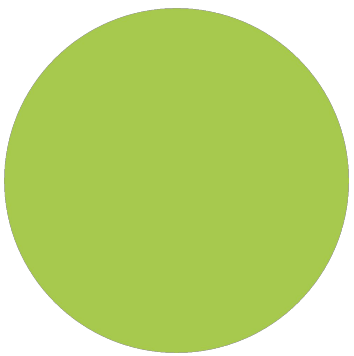
BENEFICIADOS



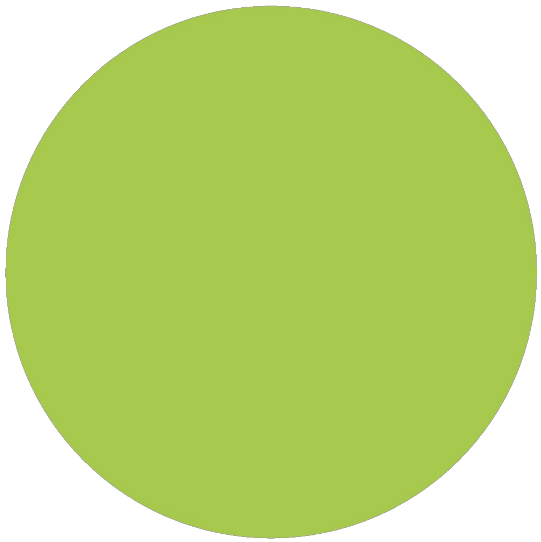
PROGRAMADOR



TESTER



ANALISTA / INVESTIGADOR



OBJETIVO DE LA DEPURACIÓN

ERRORES

OPORTUNIDADES PARA MEJORAR

SÍNTOMAS DE ERRORES

- Obtención de salida incorrecta
- Realización de operaciones fuera de lo normal
- No finalización del programa (ciclos infinitos, p. ej.)
- Caídas del programa

El depurador *(debugger)* permite:

- Ejecutar línea a línea
- Detener la ejecución temporalmente
  - En una línea de código concreta
  - Bajo determinadas condiciones
- Visualizar el contenido de las variables
- Cambiar el valor del entorno de ejecución para poder ver el efecto de una corrección en el programa

Algunos elementos no se acomodan correctamente en tu página web

Que ejecutas alguna acción pero NO recibes mensaje ni de error ni de confirmación

O al contrario, recibes mensajes de confirmación pero los datos no se actualizan

Tipos de herramientas:

- DEBUGGER
- MANUAL
- LOCAL / REMOTA

HERRAMIENTAS

- Mensajes de advertencia
- Estándares de compilación
- Verificación sintáctica y lógica

TÉCNICAS DE DEPURACIÓN

TÉCNICAS DE DEPURACIÓN

DEBUGGING

Observar valores de variables

Detener temporalmente aplicación

LOGS

Almacenar los valores

Rastreo de información

HISTORIAL

Capacidad de análisis forense

Comparar valores

Agrupar información

MONITOR REPORTES

Prevenir ataques o fallas

Observar anomalías

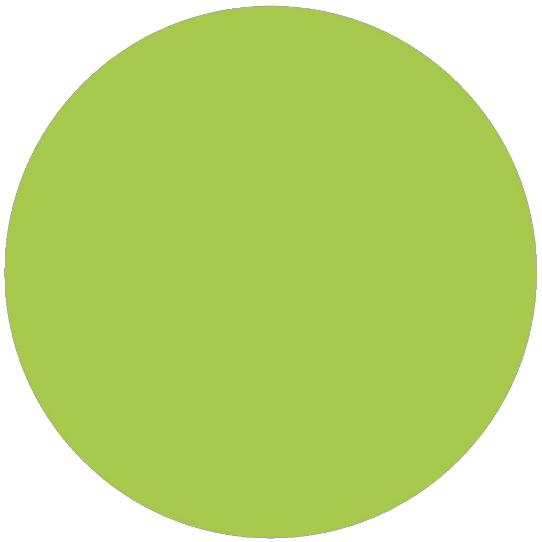
Acelerar tiempos de respuesta

DESVENTAJAS DE NO USAR LOGS

- Visibilidad nula de errores
- Metodología de trabajo no estandarizada.
- Accesos e información descentralizada
- Incremento del tiempo de respuesta

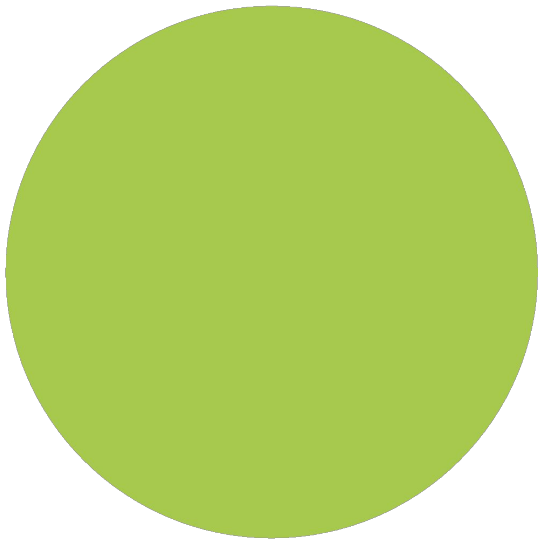
VENTAJAS DE GENERAR UN HISTORIAL / REPORTE

- Aplicar técnicas de Machine Learning
- Mejorar la gestión y el control de la información
- Detectar amenazas de red o virus
- Prevenir fugas de información, así como comportamientos inadecuados



FASE 1

ENCONTRAR ERROR



FASE 2

CORREGIR ERROR

FASE 1: PASOS PARA DEPURAR

1. Ir al módulo que falla
2. Establecer breakpoints
    a. En asignación de valores
    b. Procesamiento de valores
    c. Cambio de estados
3. Diseñar una mátrix de pruebas
4. Establecer los datos de prueba
5. Comenzar a depurar

PRUEBAS DE VERIFICACIÓN

PRUEBAS DE VERIFICACIÓN

- Tratan de reproducir el escenario fallido con los datos usados
- Se buscan nuevos escenarios donde se utilicen valores relativos siguiente flujos adicionales.
  - Otras Plataformas
  - Otros Sistemas Operativos
  - Otros exploradores
  - Otros dispositivos

PRUEBAS DE REGRESIÓN

- La matriz de pruebas durante el debugging nos permite identificar módulos impactados que requieren regresión
- las pruebas de regresión ya fallaron la primera vez al no tener suficiente cobertura, debemos incorporar los nuevos datos de prueba
- Y si se puede otros más

DOCUMENTACIÓN

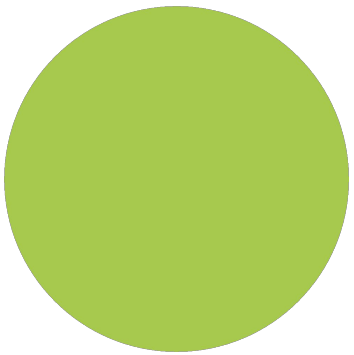
Se procura actualizar documentación:

- Comentarios en el código
- Documentación técnica
- Pruebas unitarias
- Pruebas específicas
- Matrices de pruebas
- Plan de pruebas

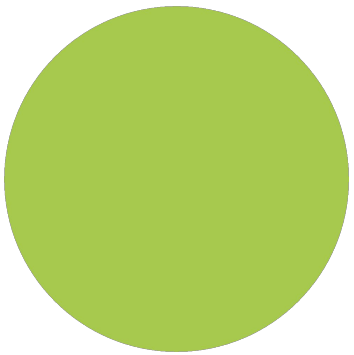
BASES DE LA AUTOMATIZACIÓN DE PRUEBAS

¿CUANDO ESTAMOS LISTOS PARA AUTOMATIZAR?

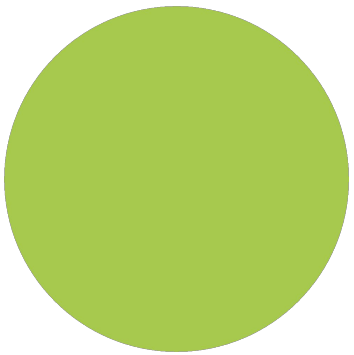
- TENEMOS PRUEBAS REPETITIVAS
- ✖ PERO NO ESTAN IDENTIFICADAS
- BUSCAMOS OPTIMIZAR LA EJECUCIÓN DE PRUEBAS
- ✖ SOLO ESCRIBIMOS SCRIPTS SIN AGRUPAR
- HEMOS DEFINIDO UN FRAMEWORK
- ✖ NO SE ESTANDARIZAN LAS PRUEBAS



PRUEBAS UNITARIAS



PRUEBAS DE INTEGRACIÓN



PRUEBAS FUNCIONALES O DE ACEPTACIÓN

TDD
(TEST DRIVEN DEVELOPMENT)

Es una técnica de diseño e implementación de software incluida dentro de la metodología XP (**Extreme Programming**).

Esta técnica nos permite obtener una c**obertura de pruebas muy alta**, aunque es importante destacar que este índice no indica que tengamos una buena calidad de tests. Por lo tanto, no debe ser un valor en el que fijarse únicamente.

Proceso TDD

1. Escribimos una prueba
2. Ejecutamos la prueba: Falla
3. Se escribe el código
4. Ejecutamos la prueba: Pasa

BDD
(BEHAVIOR DRIVEN DEVELOPMENT)

BDD es el desarrollo guiado por el comportamiento. Es un proceso que proviene de la evolución del TDD.

En BDD también se escriben pruebas antes del código, pero en vez de ser pruebas unitarias son pruebas que van a verificar que el comportamiento del código es correcto desde el punto de vista de negocio.

“

Entre más claros los casos de prueba, más eficiente la cobertura de pruebas. Entre menos errores o ambigüedad tengan los casos de pruebas, son más fácil de ejecutar o automatizar

”

GHERKIN

“

**Gherkin es un lenguaje de texto plano con estructura.
Esta diseñado para ser fácil de aprender y ser entendido por todos**

”

VENTAJAS DE GHERKIN

- Simple
- Palabras claves
- Estandariza los casos de uso
- Reduce el tiempo de diseño

Principales keywords usados en Gherkin

- Feature
- Scenario
- Given, When, Then, And, But (Steps)
- Background
- Scenario outline
- Examples

Feature: El usuario abre 1 puerta de perilla para salir

#comentarios

Scenario: El usuario tiene una puerta cerrada

Given girando la perilla

And empujando la puerta

When la puerta abre hacia afuera

Then la puerta queda abierta

And el usuario puede salir