

Universidad Tecnológica Nacional

Facultad Regional Córdoba

Cátedra de Ingeniería y Calidad de Software

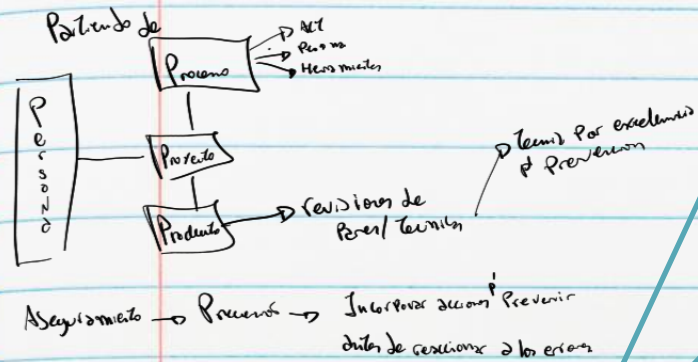
Docentes: Judith Meles & Laura Covaro

TESTING DE SOFTWARE

0

PRUEBA DE SOFTWARE

COBERTURA DE TEMAS



Testing de Software

Definición

Principios

Mitos

Defectos vs Errores

Casos de Prueba

Ciclos de Prueba

Conceptos importantes

Niveles de Prueba

Ambientes

Proceso de Pruebas

Estrategias de Prueba

Métodos de Prueba

Tipos de Prueba

TDD

Familiar →

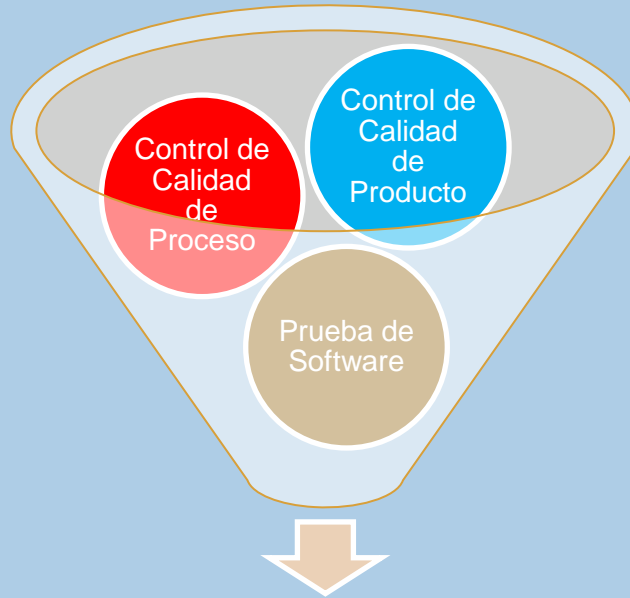
Waley Artefacto que se construye se puede revisar

Informal → Wall Through

Prueba de Software en Contexto...

Administración
de
Configuración

*Esto dentro
de la prueba
protección*



**Aseguramiento de
Calidad de Software**

¿QUÉ ES EL TESTING?



DEFINICIÓN DE PRUEBA DE SOFTWARE

No Asegura la calidad

Lo único q hace es Descubir defectos

Visión más apropiada del Testing:

- Proceso DESTRUCTIVO de tratar de encontrar defectos (cuya presencia se asume!) en el código. o más tradicional
- Se debe ir con una actitud negativa para demostrar que algo es incorrecto.
- Testing exitoso \Rightarrow es el que encuentra defectos!
- Mundialmente: 30 a 50% del costo de un software confiable

TESTING EN EL CONTEXTO: No es una Gestión

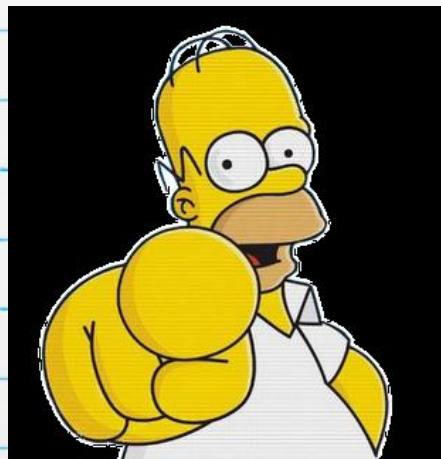
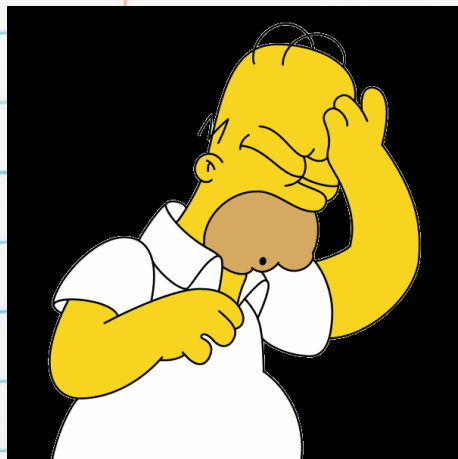
7

ASEGURAR LA CALIDAD VS CONTROLAR LA CALIDAD

- Una vez definidos los requerimientos de calidad, tengo que tener en cuenta que:
 - La calidad no puede “inyectarse” al final
 - La calidad del producto depende de tareas realizadas durante *todo* el proceso
 - Detectar errores en forma *temprana* ahorra esfuerzos, tiempo, recursos
 - La calidad no solamente abarca aspectos del producto sino también del proceso y como éstos se pueden mejorar, que a su vez evita defectos recurrentes.
 - El testing NO puede asegurar ni calidad en el software ni software de calidad

CONCEPTOS: ERROR VS DEFECTO

error cuando Pasa a otro Work Flow



el TESTING encuentra DEFECTOS

CONCEPTOS: DEFECTOS, SEVERIDAD Y PRIORIDAD

Falla \rightarrow depende la criticidad del Defecto

Botón VS Ingreso de String en
List

Grado general
el Defecto es

Severidad

- 1 - Bloqueante
- 2 - Crítico
- 3 - Mayor
- 4 - Menor
- 5 - Cosmético

mayor complejidad
de desarrollo

No se debe vincular 1/2

Prioridad

- 1 - Urgencia
- 2 - Alta
- 3 - Media
- 4 - Baja

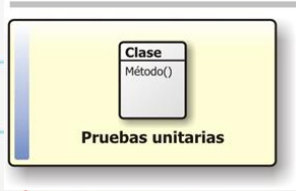
q se necesita
desarrollar, viene
del cliente del Ps
q en resp. lo necesita el
cliente.

NIVELES DE PRUEBA

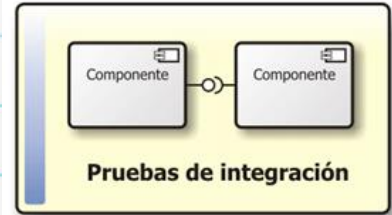
Unitate
↓
CY/CD & Testing Sense

Primer el Probar los
después Componente 2 Probar
TDD No es de Testing sino una mentalidad/
de desarrollo

Pruebas unitarias



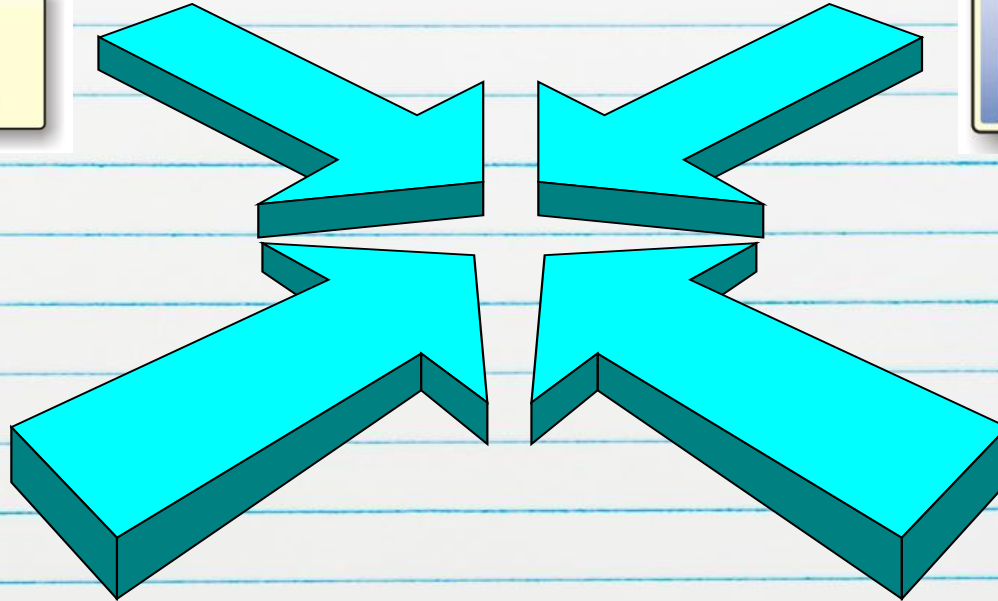
pruebas de integración



Pruebas de sistema



Pruebas de aceptación

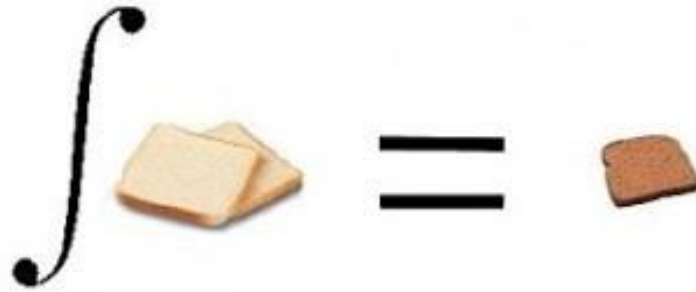


NIVELES DE TESTING: TESTING DE INTEGRACIÓN

Problema de Integración

→ Sitio Auto-motivado en CI

15

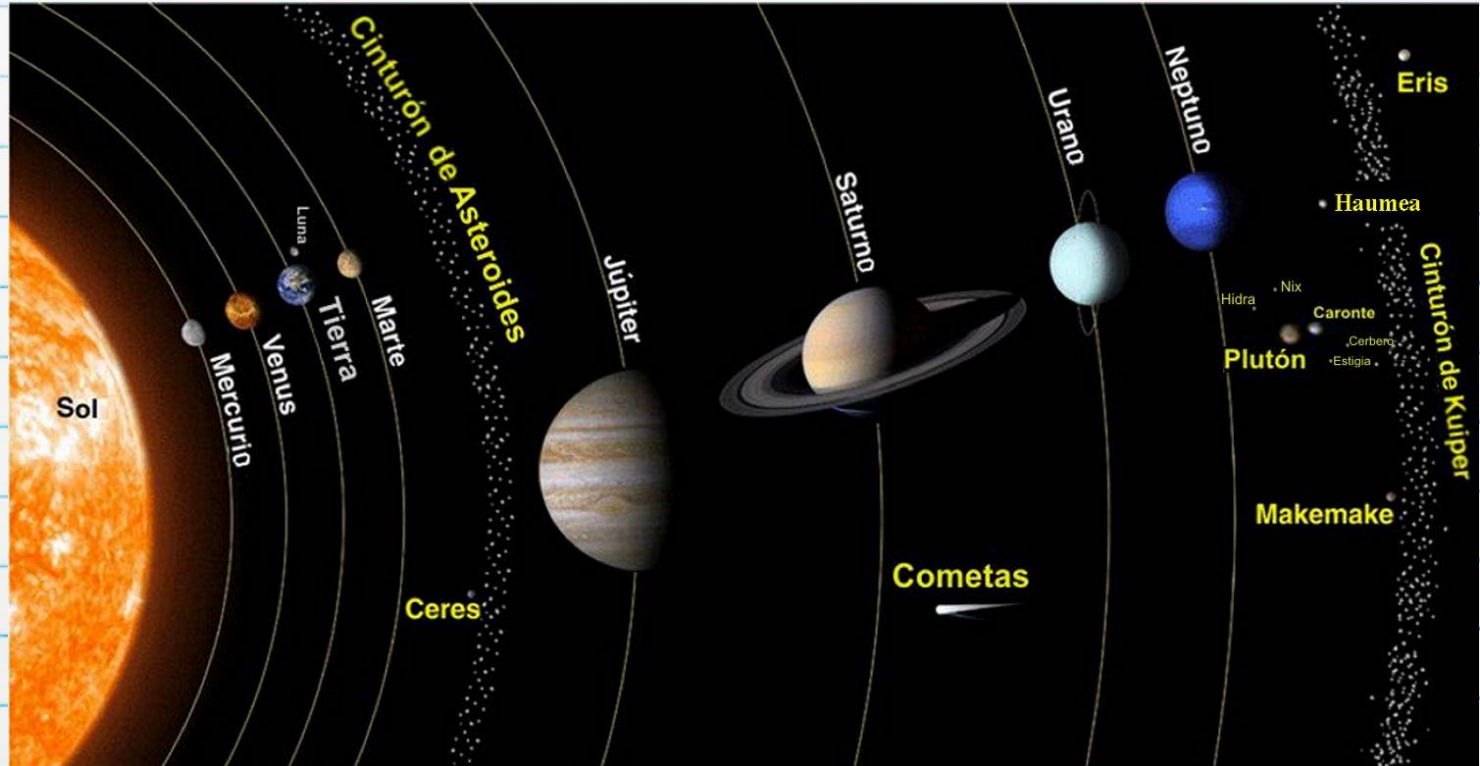


PAN INTEGRAL

Porque las ciencias se preocupan por tu regularidad

NIVELES DE TESTING: TESTING DE SISTEMA

17



NIVELES DE TESTING: TESTING DE SISTEMA o de Versión (en ciclo iterativo)

- Es la prueba realizada cuando una aplicación esta funcionando como un todo (Prueba de la construcción Final).
- Trata de determinar si el sistema en su globalidad opera satisfactoriamente (recuperación de fallas, seguridad y protección, stress, performance, etc.)
- El entorno de prueba debe corresponder al entorno de producción tanto como sea posible para reducir al mínimo el riesgo de incidentes debidos al ambiente específicamente y que no se encontraron en las pruebas.
- Deben investigar tanto requerimientos funcionales y no funcionales del sistema.

NIVELES DE TESTING:

TESTING DE ACEPTACIÓN

19



NIVELES DE TESTING:

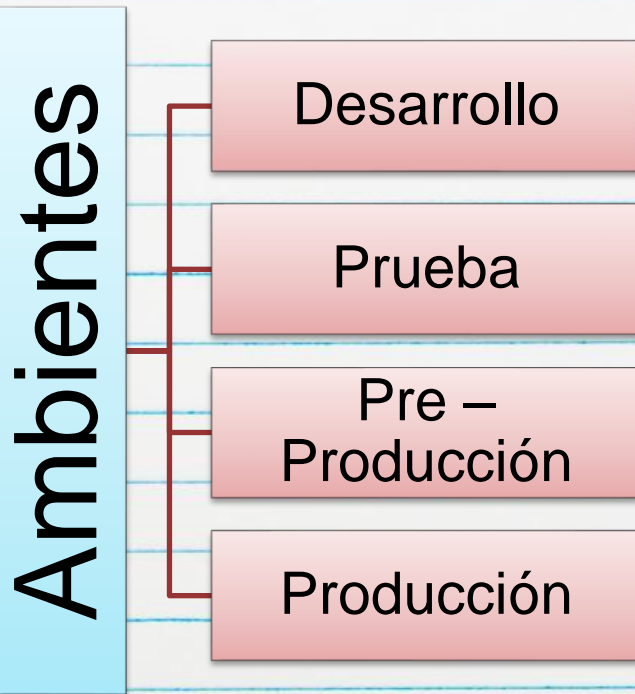
TESTING DE ACEPTACIÓN

UAT User Acceptance Testing
or UAT Review

20

- Es la prueba realizada por el usuario para determinar si la aplicación se ajusta a sus necesidades.
- La meta en las pruebas de aceptación es el de establecer confianza en el sistema, las partes del sistema o las características específicas y no funcionales del sistema.
- Encontrar defectos no es el foco principal en las pruebas de aceptación.
- Comprende tanto la prueba realizada por el usuario en ambiente de laboratorio (pruebas alfa), como la prueba en ambientes de trabajo reales (pruebas beta).

AMBIENTES PARA CONSTRUCCIÓN DEL SOFTWARE

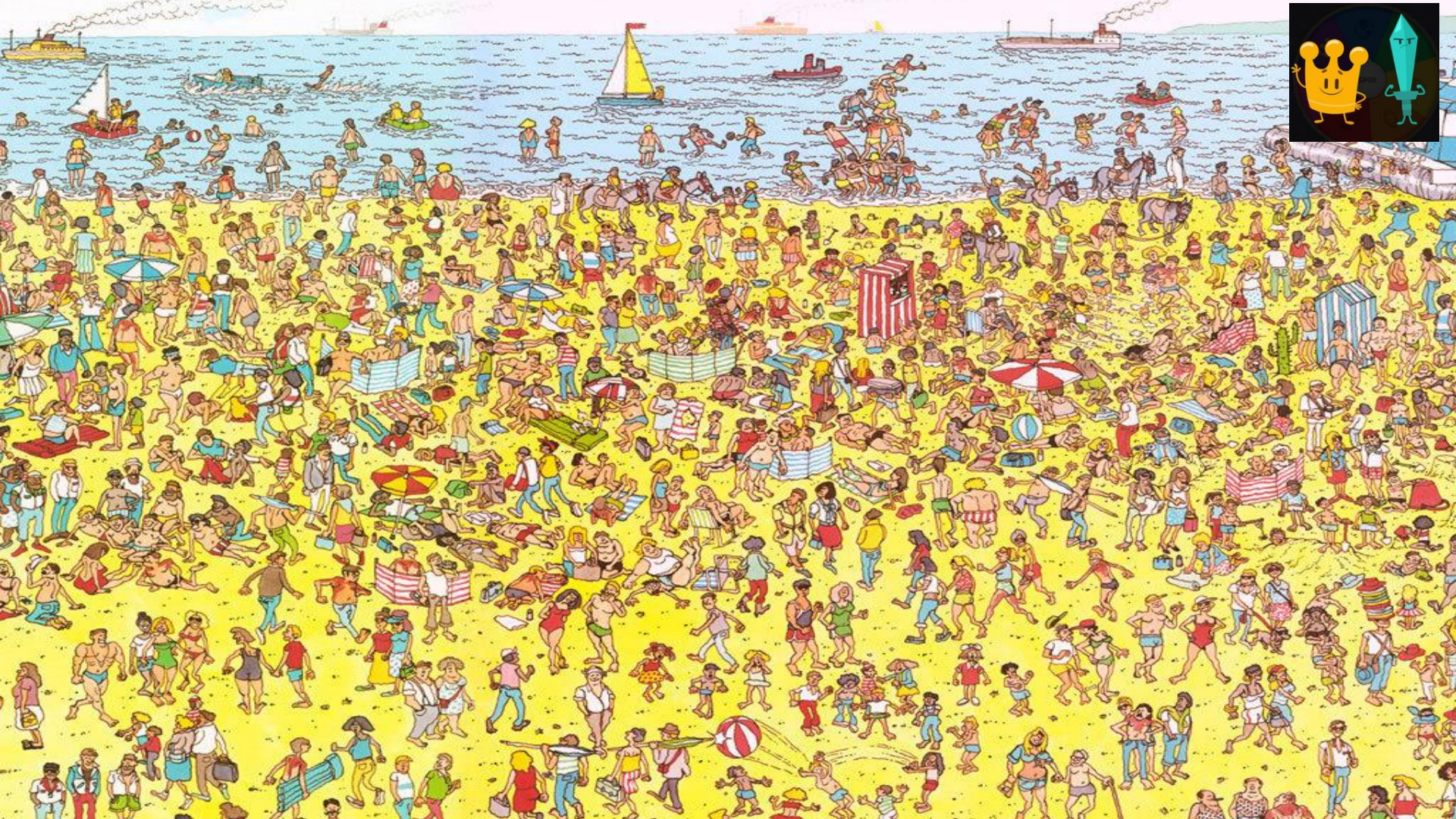


VAT Lebens. Ser Als Pero Por G, 7, 7, 7
est en Producción

¿CONOCEN A WALLY??

22





CONCEPTOS: CASO DE PRUEBA

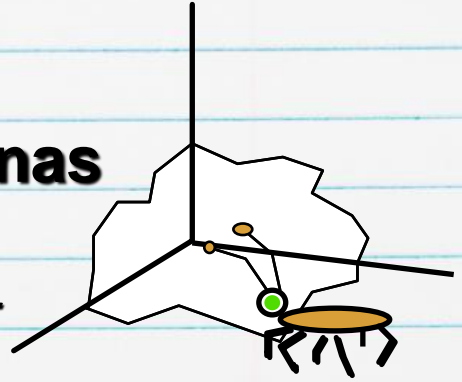
- Set de condiciones o variables bajo las cuales un tester determinará si el software está funcionando correctamente o no.
- Buena definición de casos de prueba nos ayuda a REPRODUCIR defectos

↳ Si el defecto no se puede reproducir
no es defecto

CASOS DE PRUEBA

“Los bugs se esconden en las esquinas y se congregan en los límites ...”

Boris Beizer



OBJETIVO

descubrir errores

CRITERIO

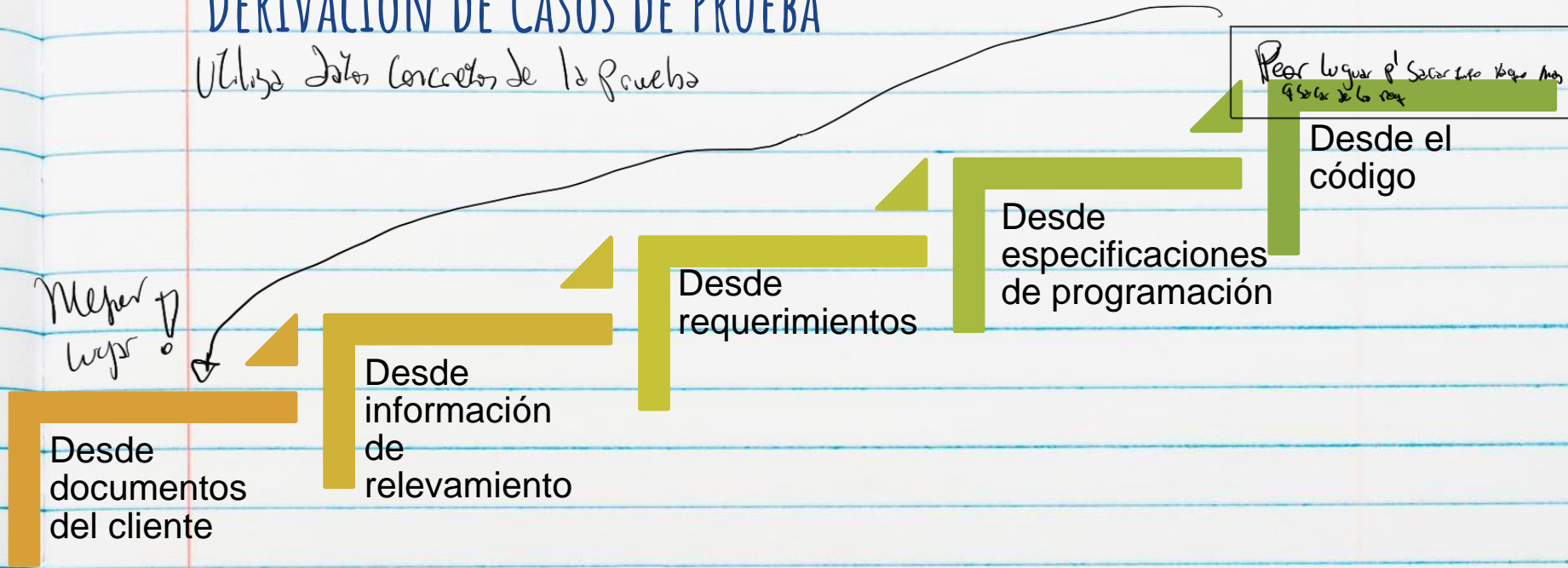
en forma completa

RESTRICCIÓN

con el mínimo de esfuerzo y tiempo

DERIVACIÓN DE CASOS DE PRUEBA

Utiliza datos concretos de la prueba



CONCLUSIONES SOBRE LA GENERACIÓN DE CASOS

- Ninguna técnica es completa
- Las técnicas atacan distintos problemas
- Lo mejor es combinar varias de estas técnicas para complementar las ventajas de cada una
- Depende del código a testear
- Sin requerimientos todo es mucho más difícil
- Tener en cuenta la conjetura de defectos
- Ser sistemático y documentar las suposiciones sobre el comportamiento o el modelo de fallas

CONDICIONES DE PRUEBA *Seteo / Contexto*

- Esta es la reacción esperada de un sistema frente a un estímulo particular, este estímulo está constituido por las distintas entradas.
- Una condición de prueba debe ser probada por al menos un caso de prueba

CASOS DE PRUEBA

Un caso de prueba es la unidad de la actividad de la prueba.

Consta de tres partes:

- 1. Objetivo: la característica del sistema a comprobar**
- 2. Datos de entrada y de ambiente: datos a introducir al sistema que se encuentra en condiciones preestablecidas.**
- 3. Comportamiento esperado: La salida o la acción esperada en el sistema de acuerdo a los requerimientos del mismo.**

ESTRATEGIAS



MÉTODOS

- Para qué usarlos? El tiempo y el presupuesto es limitado
- Hay que pasar por la mayor cantidad de funcionalidades con la menor cantidad de pruebas

Sin Atorno y Codigos

CAJA NEGRA

Se efectuan Salidas, Se ingresan
dentro de la caja



- Basado en especificaciones
 - Partición de Equivalencias
 - Análisis de valores límites
 - Etc.
- Basados en la experiencia
 - Adivinanza de Defectos
 - Testing Exploratorio

CAJA BLANCA



- Se basan en el análisis de la estructura interna del software o un componente del software.
- Se puede garantizar el testing coverage

CAJA BLANCA



- Cobertura de enunciados o caminos básicos
- Cobertura de sentencias
- Cobertura de decisión
- Cobertura de condición
- Cobertura de decisión/condición
- Cobertura múltiple
- Etc

CONCEPTOS: CICLO DE TEST Por lo menos 2

Primer ciclo \rightarrow Porque es la base

- Un ciclo de pruebas abarca la ejecución de la totalidad de los casos de prueba establecidos aplicados a una misma versión del sistema a probar.



CONCEPTOS: REGRESIÓN

- Al concluir un ciclo de pruebas, y reemplazarse la versión del sistema sometido al mismo, debe realizarse una verificación total de la nueva versión, a fin de prevenir la introducción de nuevos defectos al intentar solucionar los detectados.



CONCEPTOS: REGRESIÓN

- Sin regresión

Caso de Prueba 1

Pasa

Caso de Prueba 2

Error

Arreglo

Error

Arreglo

Pasa

Caso de Prueba 3

Pasa

CONCEPTOS: REGRESIÓN

- Sin regresión

Tomo las listas de errores y
Pruebas solo eso

Caso de
Prueba
1

Pasa

???

Pasa

???

Pasa

Si me hacen mano, no debo S.
resaltar errores

Caso de
Prueba
2

Error

Arreglo

Error

Arreglo

Pasa

Caso de
Prueba
3

Pasa

Problema
nuevo

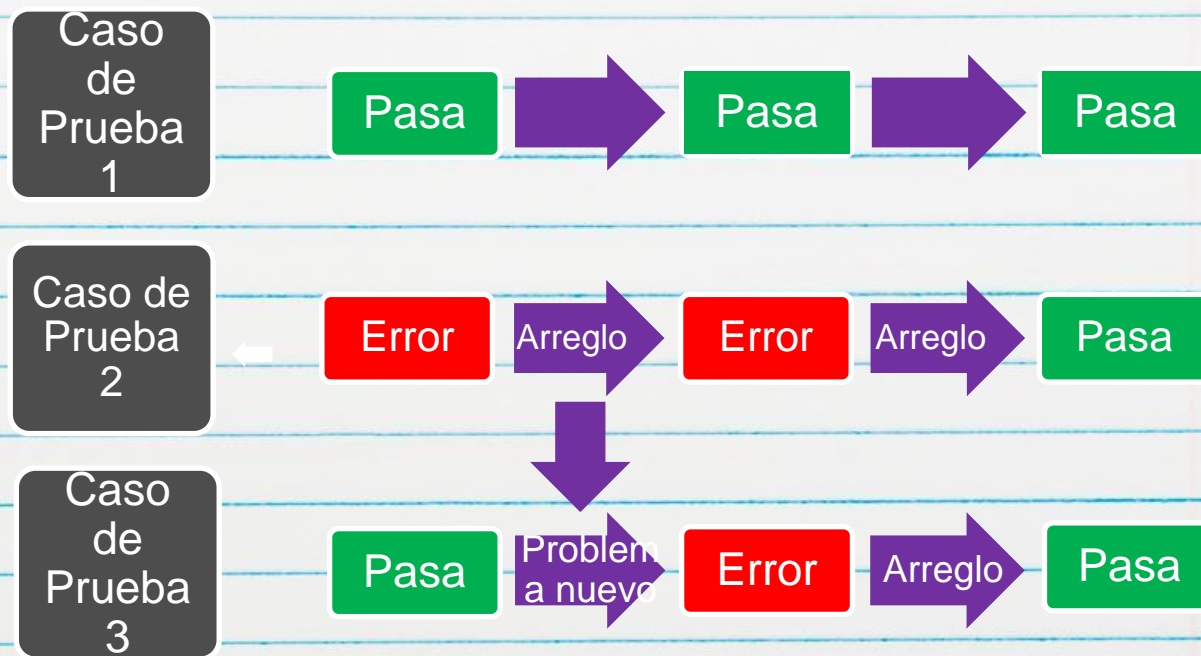
Error

Problema
escondido

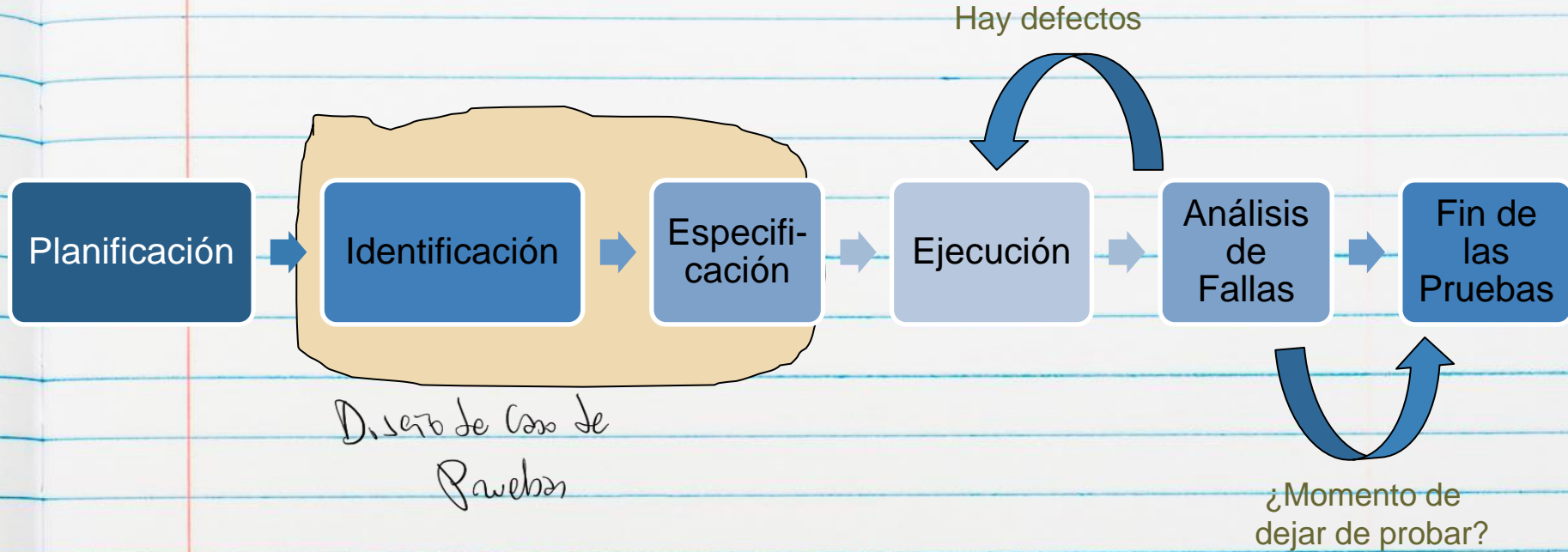
Error

CONCEPTOS: REGRESIÓN

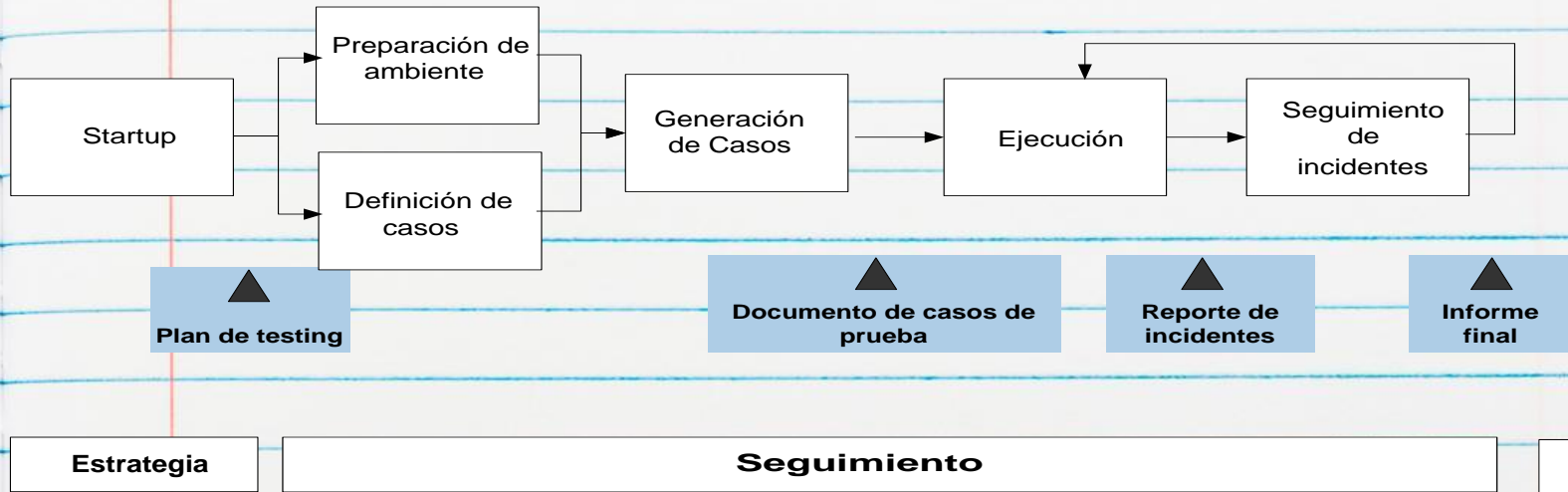
- Con regresión



PROCESO DE PRUEBAS



EJEMPLO DE ETAPAS/ENTREGABLES DE TESTING



PROCESO DEL TESTING

Planificación y Control

- La Planificación de las pruebas es la actividad de verificar que se entienden las metas y los objetivos del cliente, las partes interesadas (stakeholders), el proyecto, y los riesgos de las pruebas que se pretende abordar.
- Construcción del Test Plan:
 - Riesgos y Objetivos del Testing
 - Estrategia de Testing
 - Recursos
 - Criterio de Aceptación
- Controlar:
 - Revisar los resultados del testing
 - Test coverage y criterio de aceptación
 - Tomar decisiones



PROCESO DEL TESTING

- Revisión de la base de pruebas
- Verificación de las especificaciones para el software bajo pruebas
- Evaluar la testeabilidad de los requerimientos y el sistema
- Identificar los datos necesarios
- Diseño y priorización de los casos de las pruebas
- Diseño del entorno de prueba



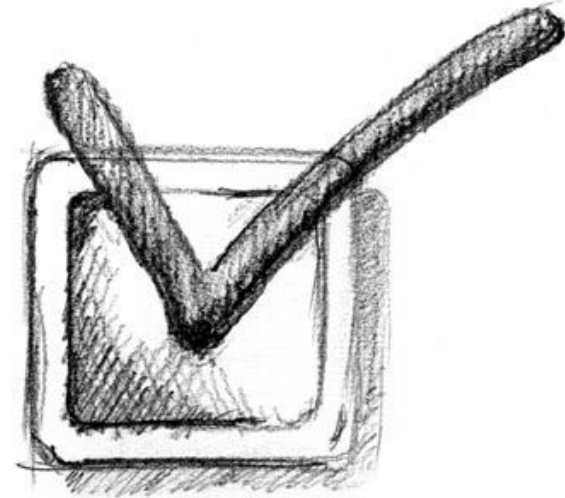
PROCESO DEL TESTING

Ejecución

- Desarrollar y dar prioridad a nuestros casos de prueba
- Crear los datos de prueba
- Automatizar lo que sea necesario
- Creación de conjuntos de pruebas de los casos de prueba para la ejecución de la prueba eficientemente.
- Implementar y verificar el ambiente.
- Ejecutar los casos de prueba
- Registrar el resultado de la ejecución de pruebas y registrar la identidad y las versiones del software en las herramientas de pruebas.
- Comparar los resultados reales con los resultados esperados.

PROCESO DEL TESTING

- Evaluar los criterios de Aceptación
- Reporte de los resultados de las pruebas para los stakeholders.
- Recolección de la información de las actividades de prueba completadas para consolidar.
- Verificación de los entregables y que los defectos hayan sido corregidos.
- Evaluación de cómo resultaron las actividades de testing y se analizan las lecciones aprendidas.





PROCESO DEL TESTING: QUICK QUIZ

¿Cuántas líneas de código necesito para empezar a hacer testing?

0

POR QUÉ EL TESTING ES NECESARIO? QUICK QUIZ



Porque la existencia de defectos en el software es inevitable



Para llenar el tiempo entre fin del desarrollo y el día del release



Para probar que no hay defectos



Porque el testing está incluido en el plan del proyecto



Porque debuggear mi código es rápido y simple



POR QUÉ EL TESTING ES NECESARIO? QUICK QUIZ



Para evitar ser demandado por mi cliente



Para reducir riesgos



Para construir la confianza en mi producto



Porque las fallas son muy costosas



Para verificar que el software se ajusta a los requerimientos y validar que las funciones se implementan correctamente.



EL TESTING Y EL CICLO DE VIDA



VERIFICACIÓN Y VALIDACIÓN

VERIFICACION

¿Estamos construyendo el sistema correctamente?

VALIDACION

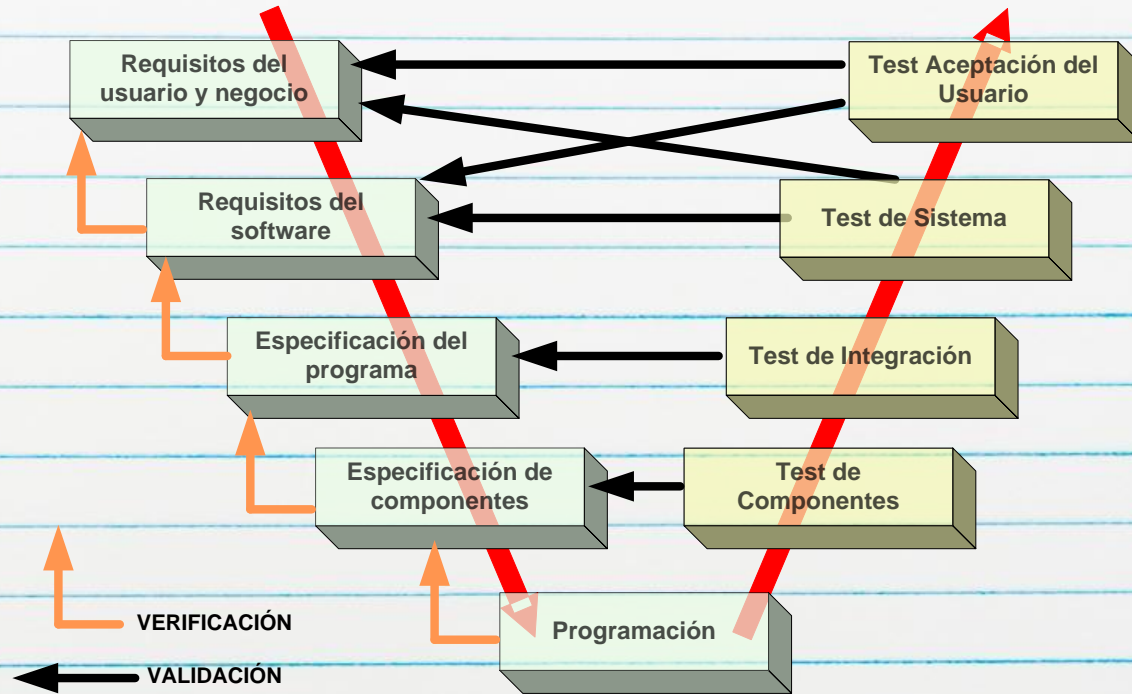
¿Estamos construyendo el sistema correcto?

EL TESTING EN EL CICLO DE VIDA DEL SOFTWARE

Objetivos de involucrar las actividades de Testing de manera temprana:

- Dar visibilidad de manera temprana al equipo, de cómo se va a probar el producto.
- Disminuir los costos de correcciones de defectos

MODELO EN V



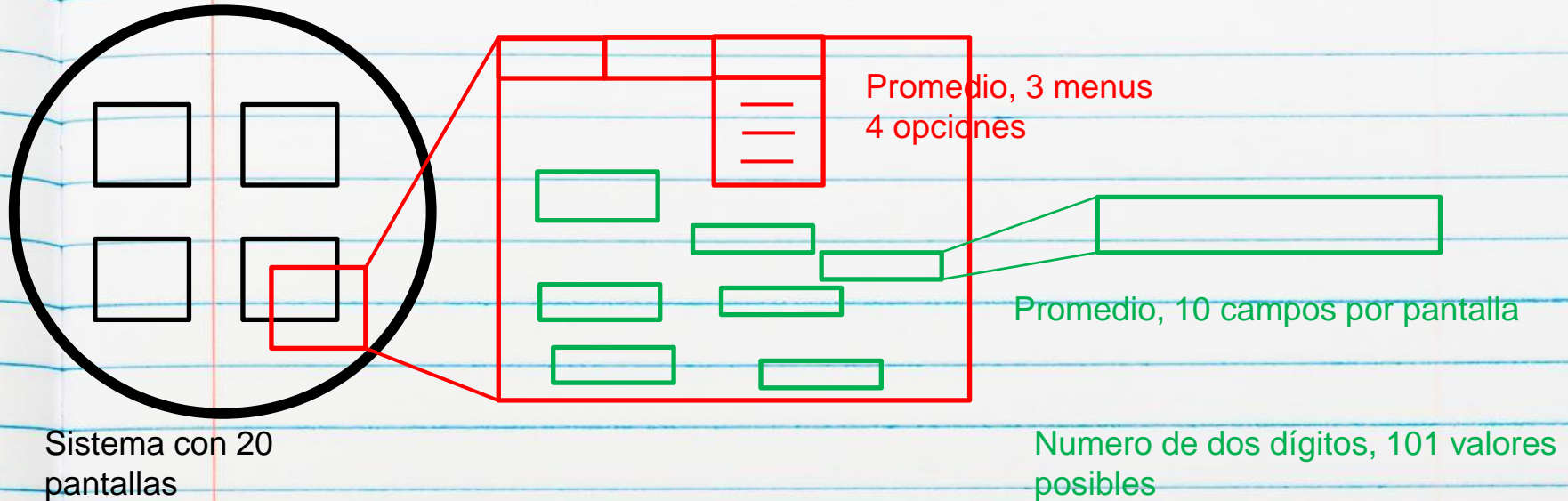
ROMPER MITOS

- El Testing es una etapa que comienza al terminar de codificar.
- El Testing es probar que el software funciona.
- **TESTING = CALIDAD de PRODUCTO**
- **TESTING = CALIDAD de PROCESO**
- El tester es el enemigo del programador.



¿CUÁNTO TESTING ES SUFICIENTE?

57



Total de testing exhaustivo:

- $20 \times 3 \times 4 \times 10 \times 100 = 240.000$

Suponiendo 1 seg por prueba:

4000 minutos -> 67 horas -> 8,5 días

10 seg -> 17 semanas

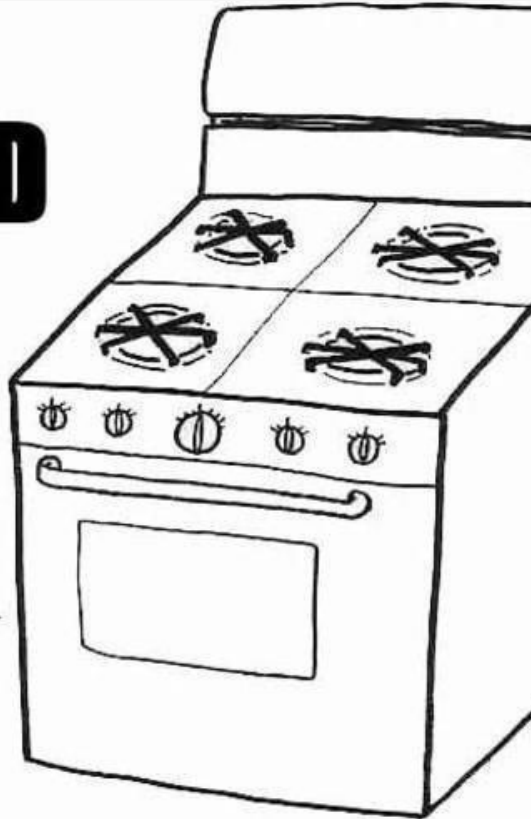
1 min -> 1,4 años

10 min -> 13,7 años

¿CUÁNTO TESTING ES SUFICIENTE?

58

**USTED
ESTA
AQUI**



¿CUÁNTO TESTING ES SUFICIENTE? QUICK QUIZ



Cuando se terminó todo lo planificado



Nunca es suficiente



Cuando se ha probado que el sistema funciona correctamente



Cuando se tiene la confianza de que el sistema funciona correctamente



Depende del riesgo de tu sistema



¿CUÁNTO TESTING ES SUFICIENTE?

- El testing exhaustivo es imposible.
- Decidir cuánto testing es suficiente depende de:
 - Evaluación del nivel de riesgo
 - Costos asociados al proyecto
- Usamos los riesgos para de terminar:
 - Que testear primero
 - A qué dedicarle más esfuerzo de testing
 - Que no testear (por ahora)

¿CUÁNTO TESTING ES SUFICIENTE?

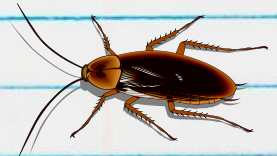
→ Variables!

- El **Criterio de Aceptación** es lo que comúnmente se usa para resolver el problema de determinar cuándo una determinada fase de testing ha sido completada.
- Puede ser definido en términos de:
 - Costos
 - % de tests corridos sin fallas
 - **Fallas predichas aún permanecen en el software**
 - No hay defectos de una determinada severidad en el software

PRINCIPIOS DEL TESTING



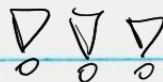
- El testing muestra presencia de defecto.
- El testing exhaustivo es imposible
- Testing temprano
- Agrupamiento de Defectos
- Paradoja del Pesticida → hay que cambiar Testers y casos de prueba
- El testing es dependiente del contexto
- Falacia de la ausencia de errores



PRINCIPIOS DEL TESTING

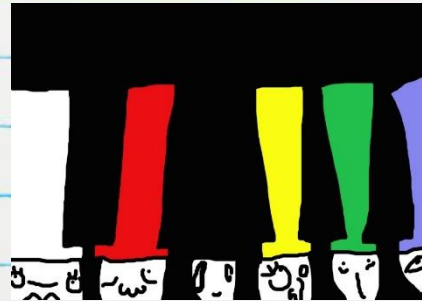
⤴ Solo no se cumple el 7.º

- Un programador debería evitar probar su propio código.
- Una unidad de programación no debería probar sus propios desarrollos.
- Examinar el software para probar que no hace lo que se supone que debería hacer es la mitad de la batalla, la otra mitad es ver que hace lo que no se supone que debería hacer.
- No planificar el esfuerzo de testing sobre la suposición de que no se van a encontrar defectos.



LA PSICOLOGÍA DEL TESTING

- La búsqueda de fallas puede ser visto como una crítica al producto y/o su autor
- La construcción del software requiere otra mentalidad a la de testear el software



LA PSICOLOGÍA DEL TESTING: DESARROLLADORES VS TESTERS

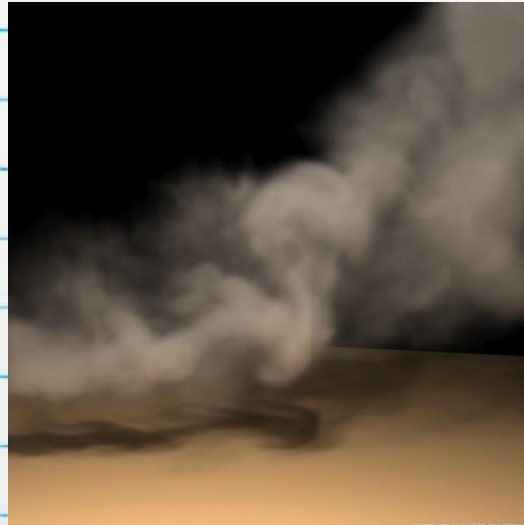
Algunos o particularmente testers



CONCEPTOS: SMOKE TEST

66

- Smoke Test:
 - Primer corrida de los tests de sistema que provee cierto aseguramiento de que el software que está siendo probado no provoca una falla catastrófica.





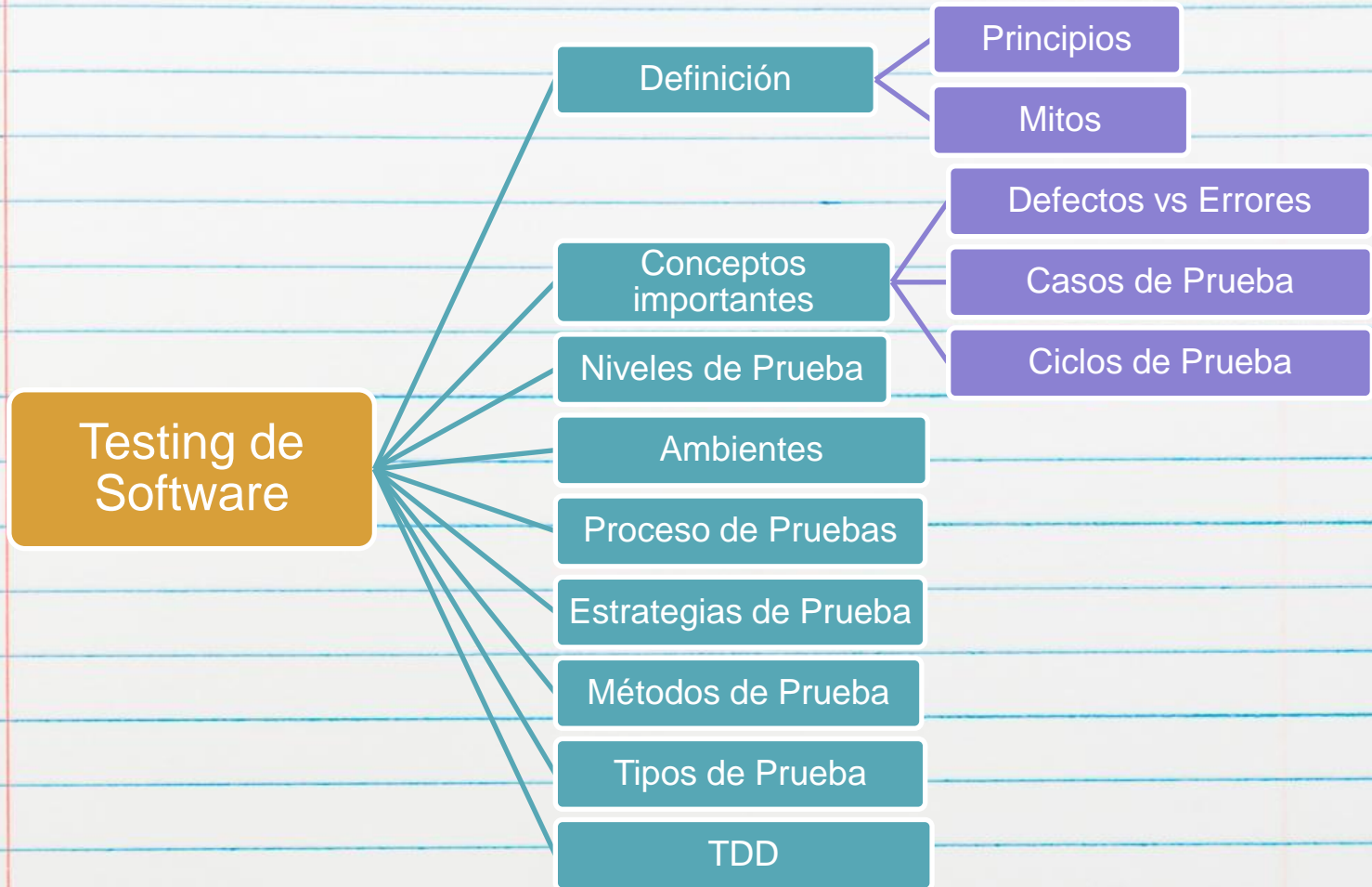
CONCEPTOS: SMOKE TEST: QUICK QUIZ

Por qué creen que se llama “Smoke test”?

1. Tiene su origen en las pruebas que hacían los ingenieros electrónicos, al comprobar si una placa empezaba a humear al aplicar
2. Porque hay que probar hasta que la computadora eche humo
3. Se refiere a ensayos físicos realizados con humo en sistemas cerrados de tuberías para detectar grietas/roturas
4. Ninguna es correcta



COBERTURA DE TEMAS



TIPOS DE PRUEBAS



- **Testing Funcional**
 - Las pruebas se basan en funciones y características (descripta en los documentos o entendidas por los testers) y su interoperabilidad con sistemas específicos
 - Basado en Requerimientos
 - Basado en los procesos de negocio

TIPOS DE PRUEBAS



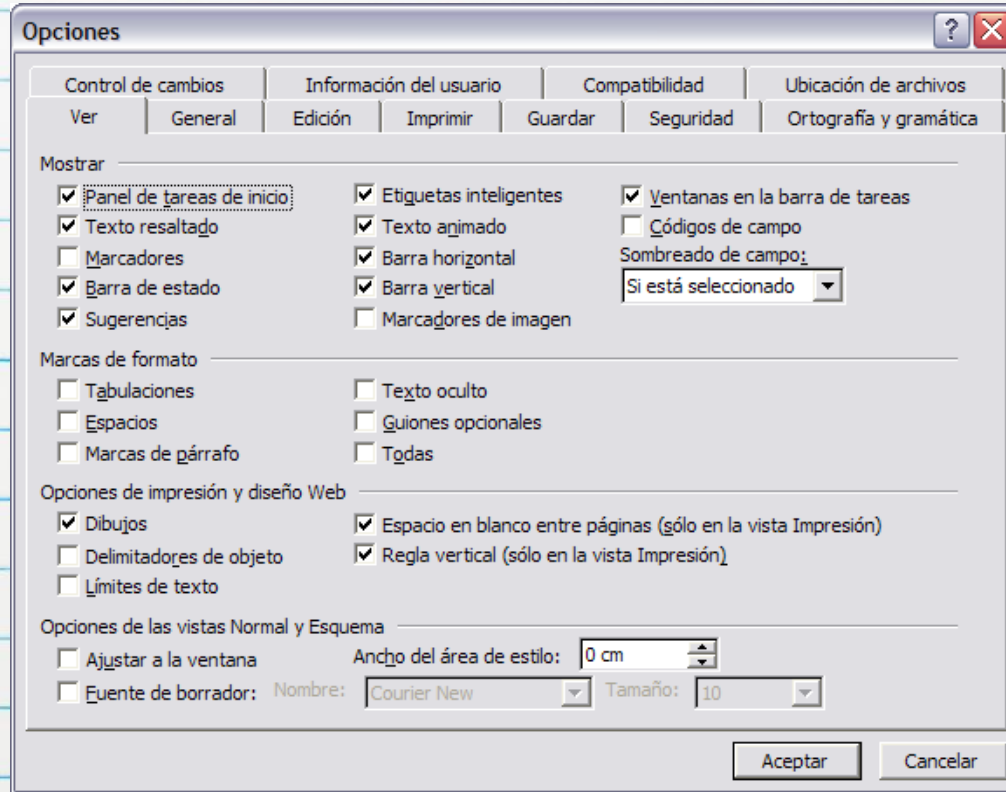
- ## Testing No Funcional

- Es la prueba de “cómo” funciona el sistema
- NO HAY QUE OLVIDARLAS!!!! Los requerimientos no funcionales son tan importantes como los funcionales
 - Performance Testing
 - Pruebas de Carga
 - Pruebas de Stress
 - Pruebas de usabilidad,
 - Pruebas de mantenimiento
 - Pruebas de fiabilidad
 - Pruebas de portabilidad

PRUEBAS DE INTERFACES DE USUARIOS

Usuario en control
+
Muchas combinaciones
=
Más pruebas

- Funciones de negocios
- **Interfaces de usuarios**
- Performance
- Carga
- Estrés
- Volumen
- Configuración
- Instalación

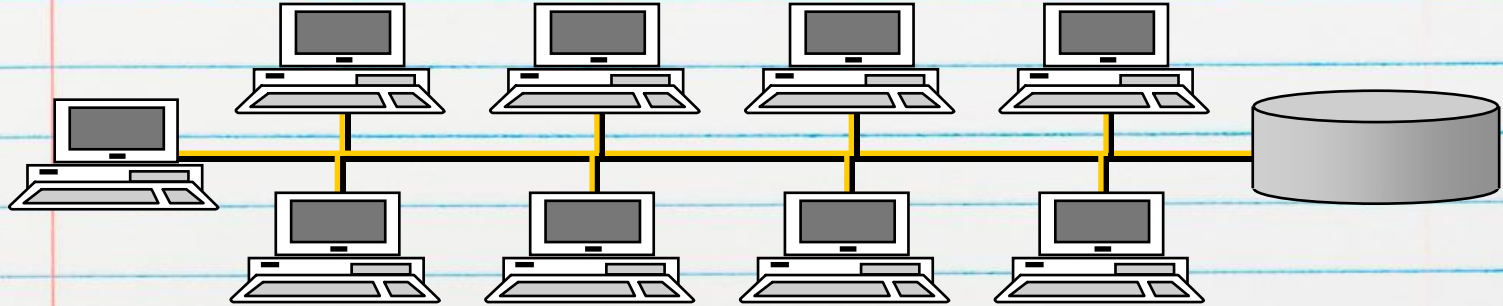


**Las GUIs,
son mucho
más
complejas
que las
interfaces
basadas en
caracteres**

PRUEBA DE PERFORMANCE

- Prueba de performance
 - Tiempo de respuesta
 - Concurrencia

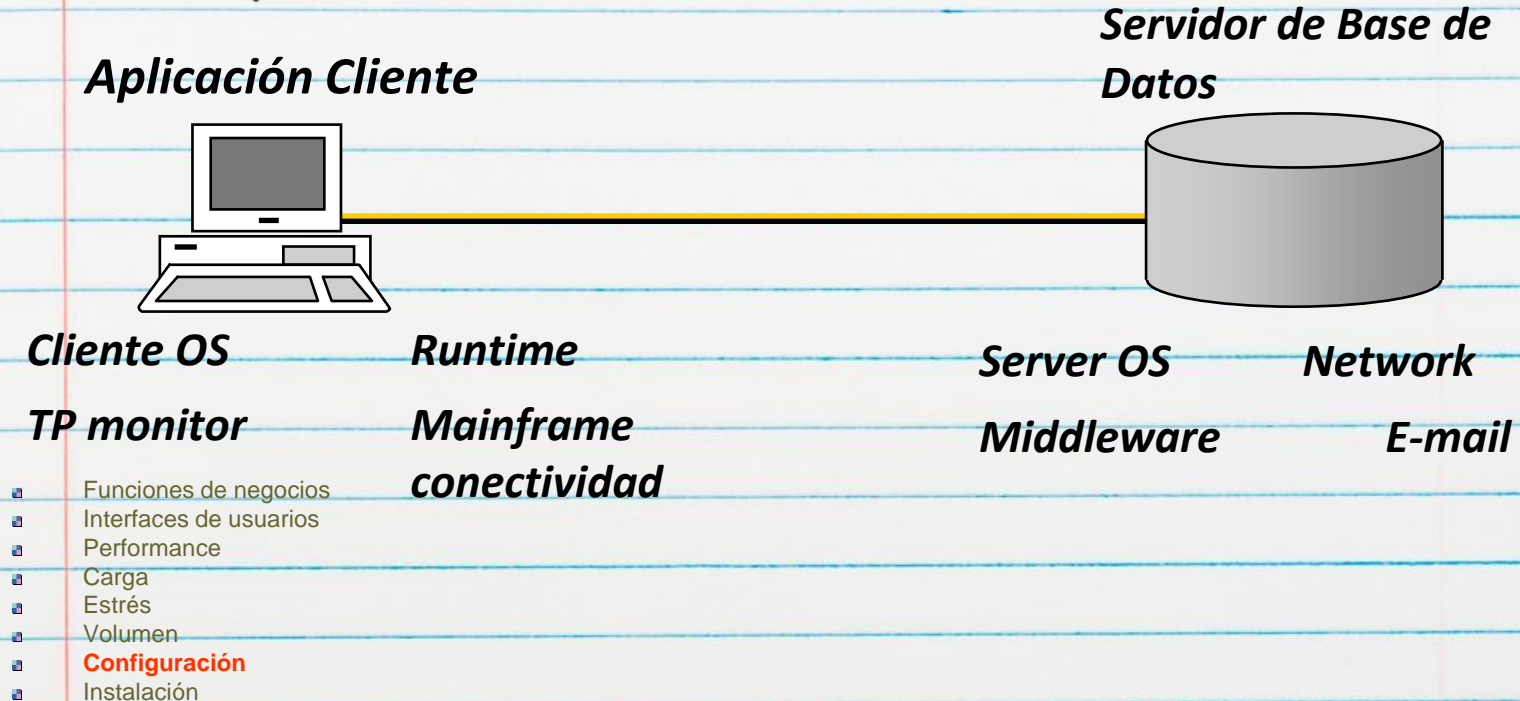
- Funciones de negocios
- Interfaces de usuarios
- **Performance**
- Carga
- Estrés
- Volumen
- Configuración
- Instalación



PRUEBA DE CONFIGURACIÓN

❑ Prueba de Configuración

- *Compatible con X video drivers*
- *Conexiones de red*
- *Software de terceras partes*



TDD

“El acto de diseñar tests es uno de los mecanismos conocidos más efectivos para prevenir errores...El proceso mental que debe desarrollarse para crear tests útiles puede descubrir y eliminar problemas en todas las etapas del desarrollo”

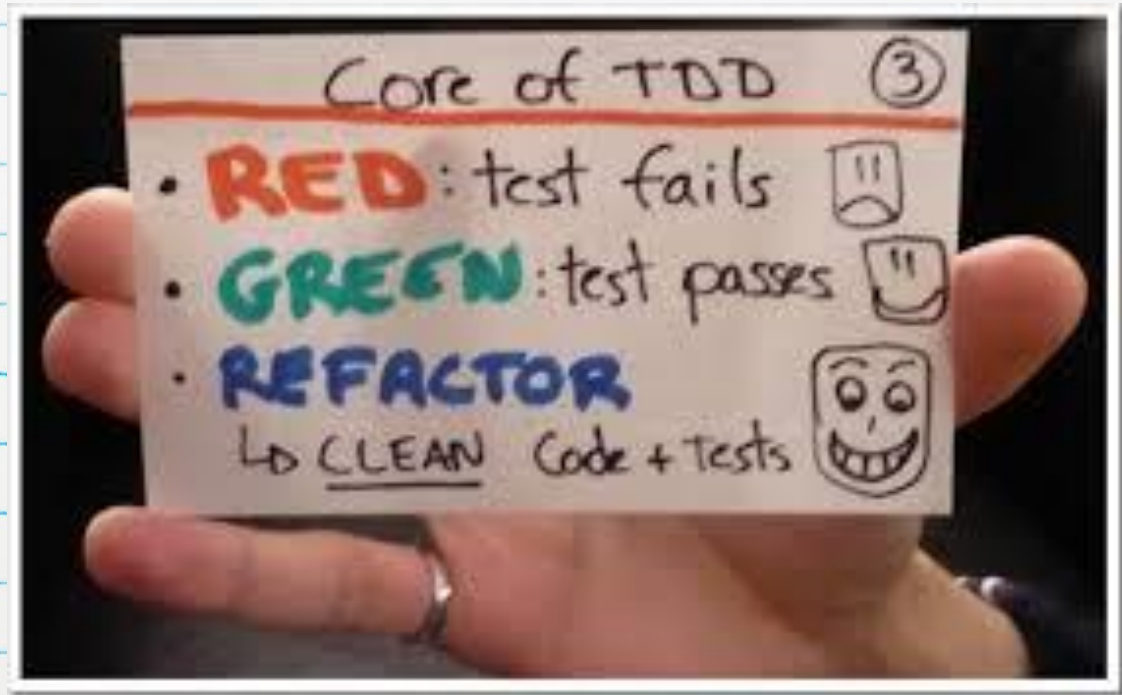
B. Beizer

“Test-Driven Development”: Kent Beck. XP

TDD

- Desarrollo guiado por pruebas de software, o Test-driven development (TDD)
- Es una técnica avanzada que involucra otras dos prácticas: *Escribir las pruebas primero (Test First Development)* y *Refactorización (Refactoring)*.
- Para escribir las pruebas generalmente se utilizan las pruebas unitarias

TDD



And repeat....

BIBLIOGRAFÍA

- “El Arte de Probar el Software”, G. Myers.
- IEEE Std. 610-1990
- IEEE Std. 829-1998 - Standard for Software Test Documentation
- ISTQB Foundation Level Syllabus
- “Test Driven Development: By Example”, Kent Beck
- “The Complete Guide to Software Testing” – Bill Hetzel
- “Software Testing Techniques, 2nd edition” – Boris Beizer
- “Agile Testing: A Practical Guide for Testers and Agile Teams” – L. Crispin