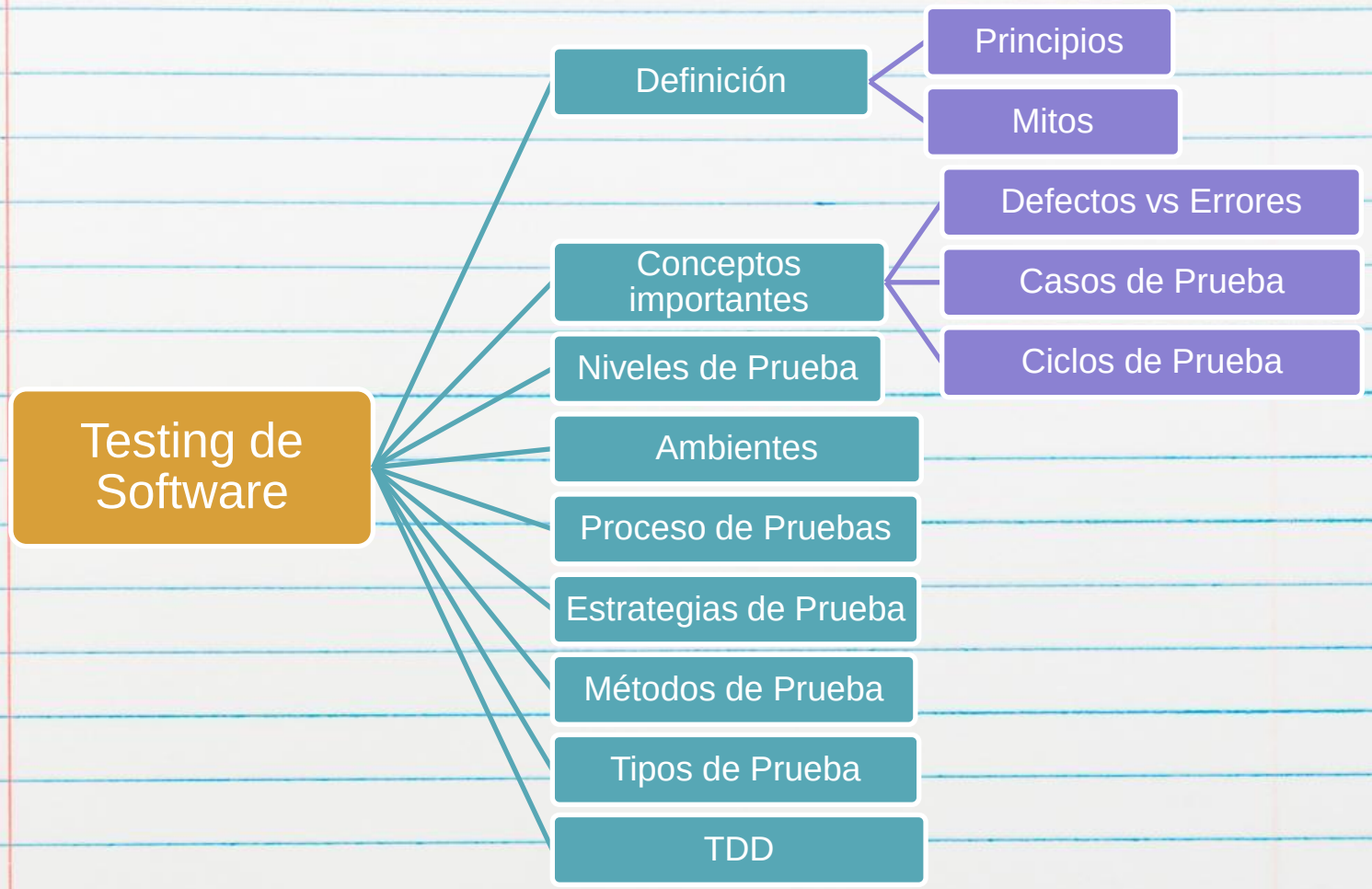
Universidad Tecnológica Nacional
Facultad Regional Córdoba
Cátedra de Ingeniería de Software
Docentes: Judith Meles & Laura Covaro

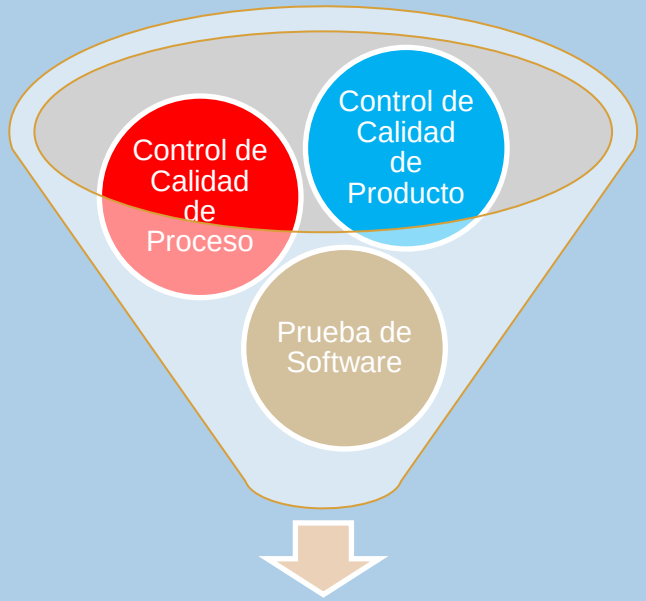
TESTING DE SOFTWARE

Cobertura de temas



Prueba de Software en Contexto…

Administración de Configuración



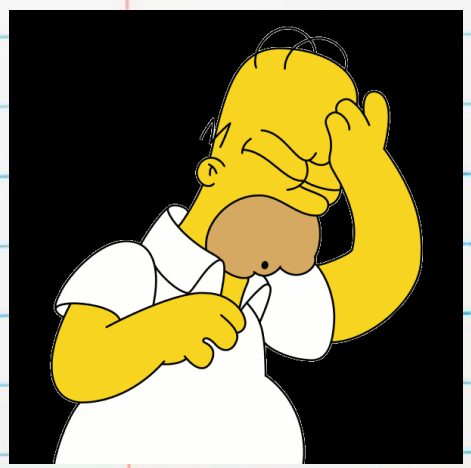
¿Qué es el testing?

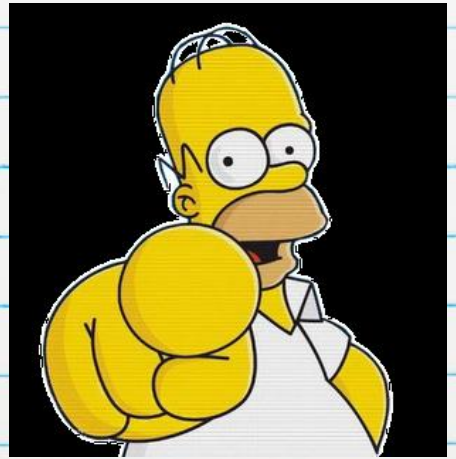
Definición de Prueba de software

Visión más apropiada del Testing:

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

Conceptos: Error vs Defecto



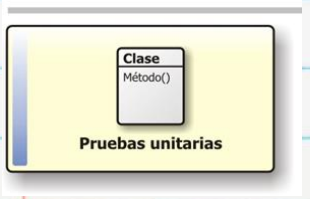


Conceptos: Defectos, severidad y prioridad

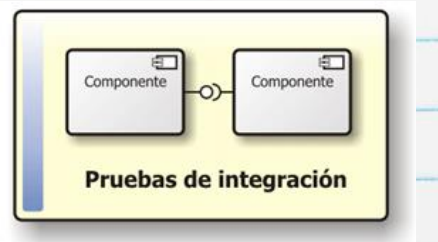
Severidad	Prioridad
1 – Bloqueante	1 – Urgencia
2 – Crítico	2 – Alta
3 – Mayor	3 – Media
4 – Menor	4 – Baja
5 - Cosmético	

NIVELES DE PRUEBA

Pruebas unitarias



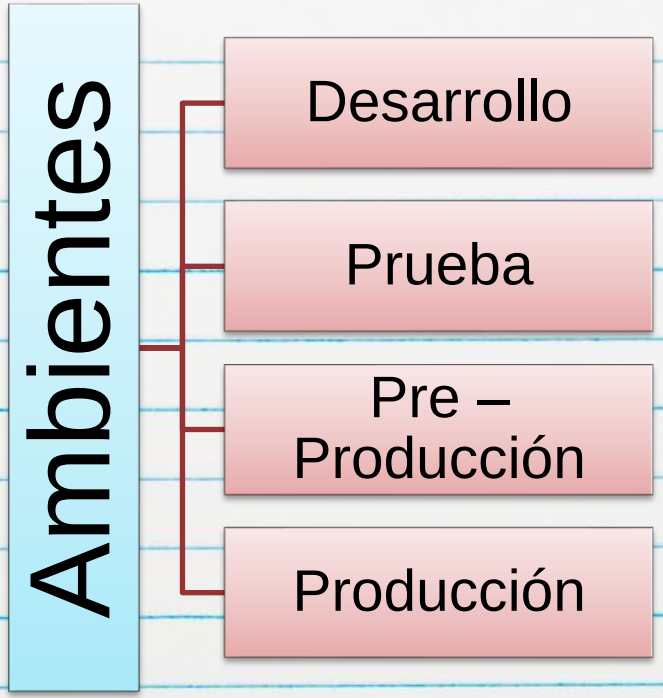
pruebas de integración



Pruebas de sistema

Pruebas de aceptación

Ambientes para construcción del Software



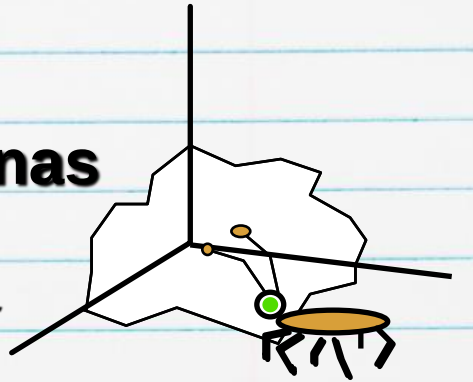
Conceptos: Caso de Prueba

- 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

CASOS DE PRUEBA

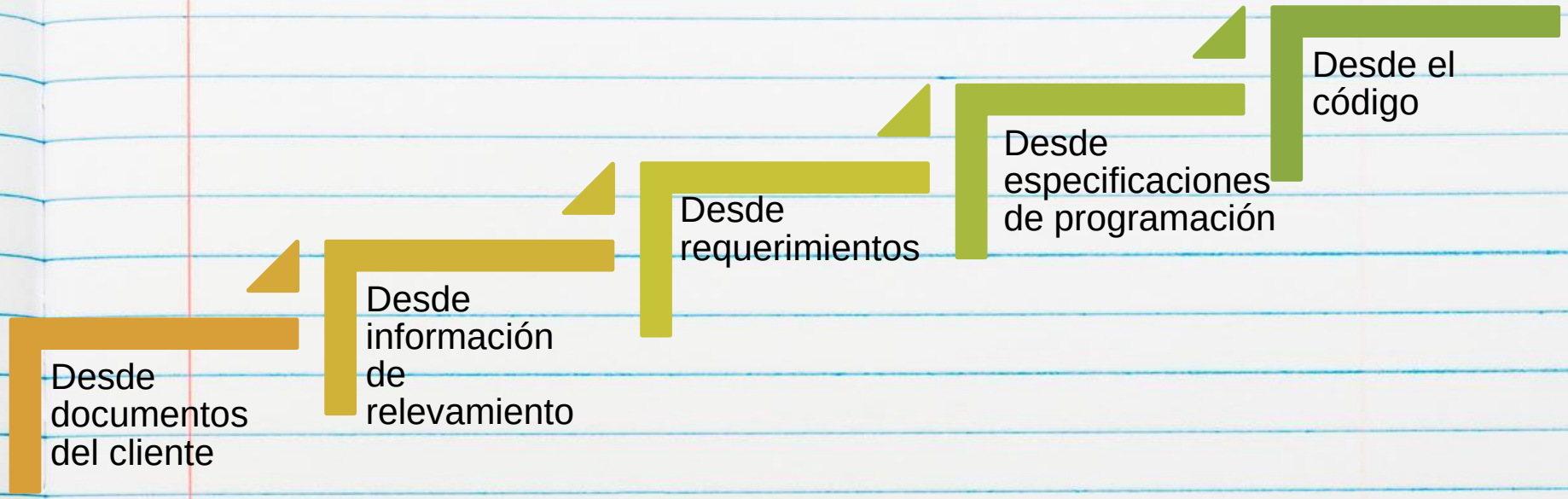
"Los bugs se esconden en las esquinas y se congregan en los límites ..."

Boris Beizer



OBJETIVO	**descubrir errores**
CRITERIO	**en forma completa**
RESTRICCIÓN	**con el mínimo de esfuerzo y tiempo**

Derivación de Casos de Prueba



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

- 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

Estrategias

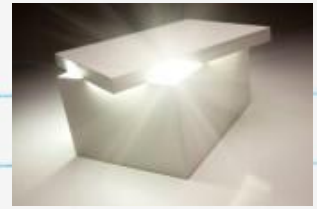
Métodos

- 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

Caja Negra

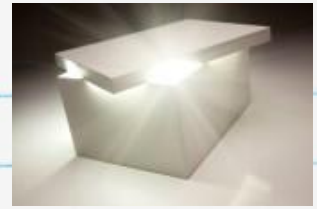
- Basado en especificaciones
  - Partición de Equivalencias
  - Análisis de valores límites
  - Etc.
- Basados en la experiencia
  - Adivinanza de Defectos
  - Testing Exploratorio

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

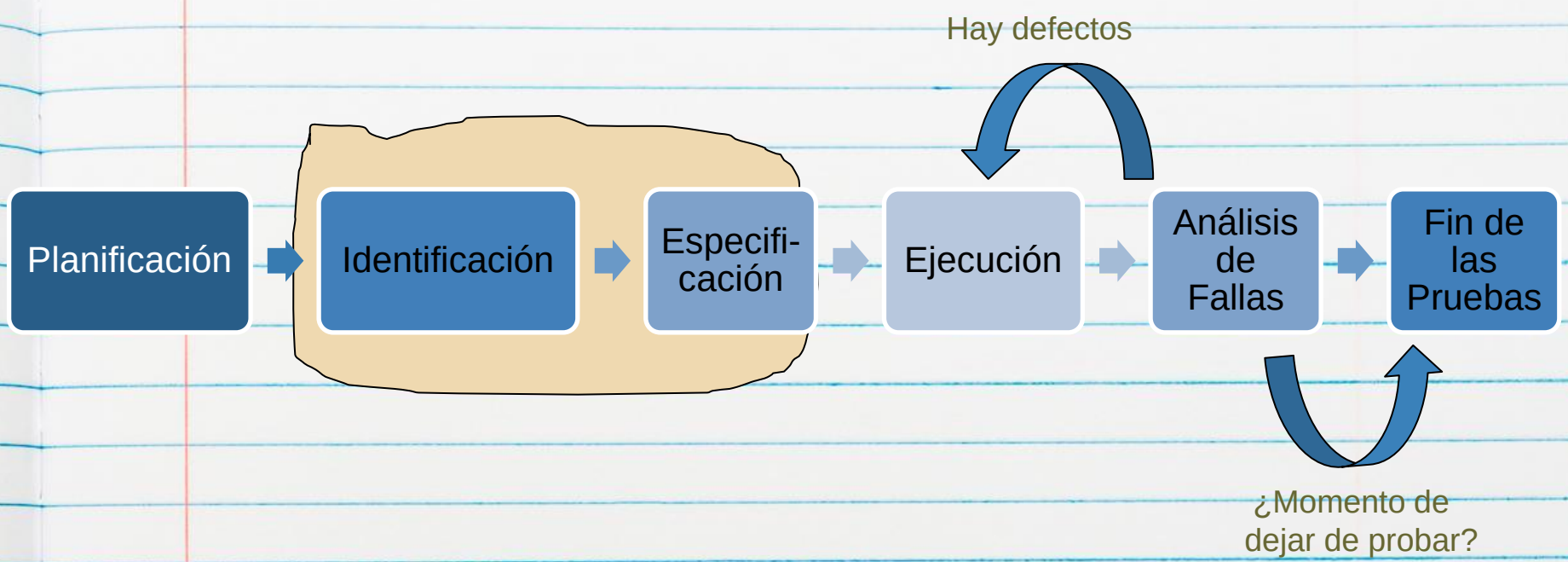
Conceptos: Ciclo de Test

- 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.

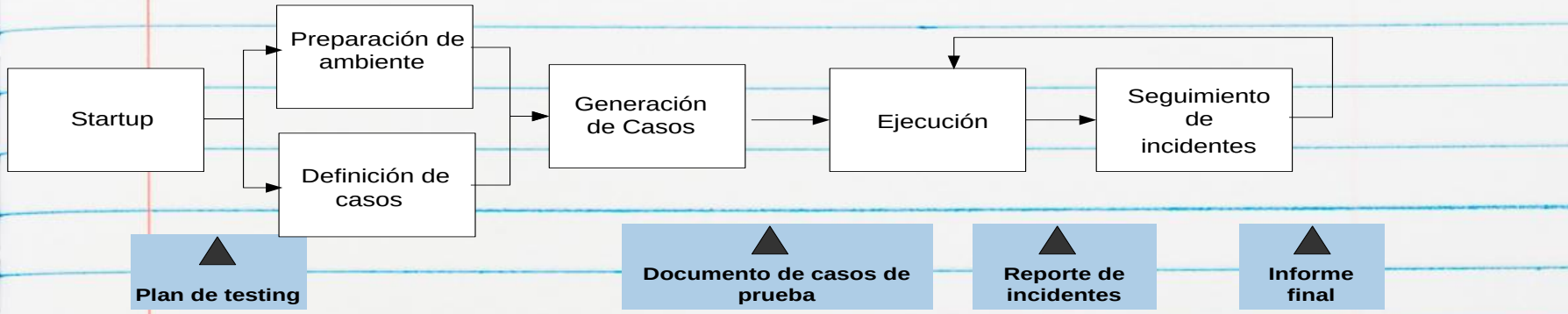
Conceptos: Regresión

- 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.

Proceso de Pruebas



Ejemplo de etapas/entregables de Testing



El Testing y el Ciclo de Vida

Verificación y Validación

VERIFICACION

¿Estamos construyendo el sistema correctamente?

VALIDACION

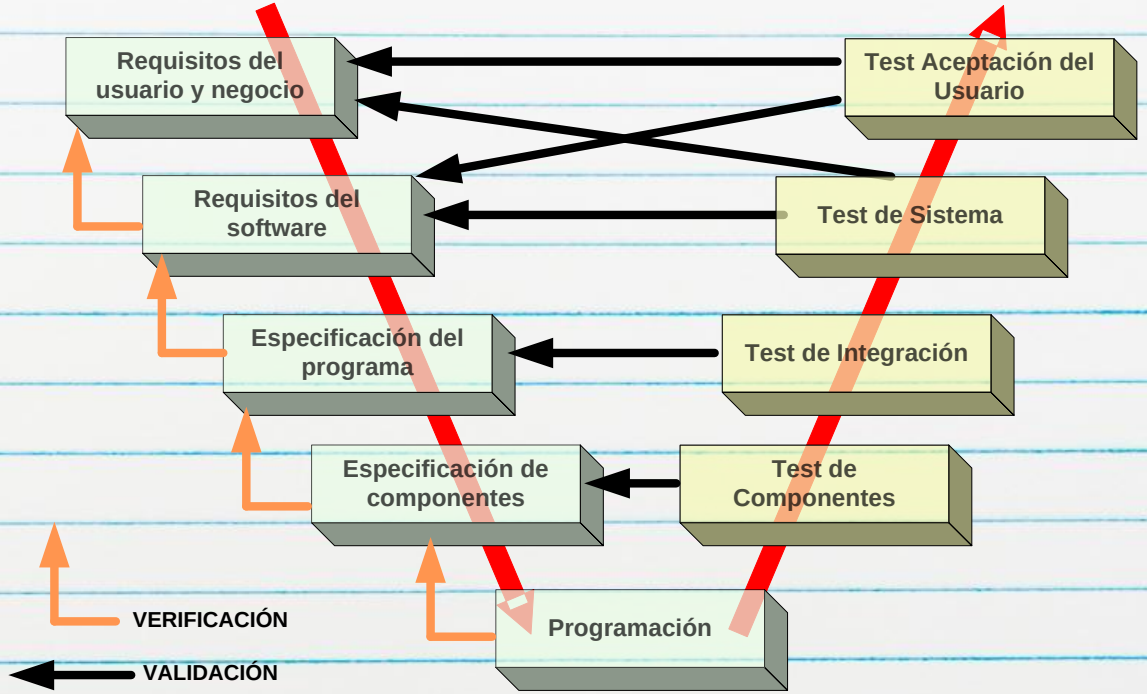
¿Estamos construyendo el sistema correcto?

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

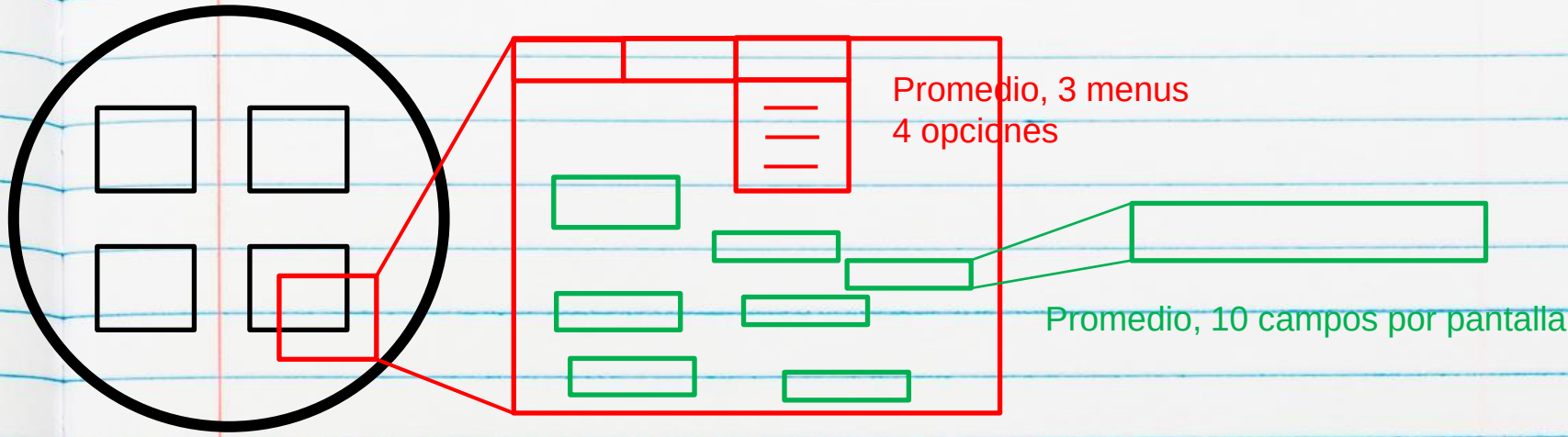
Modelo en V


Requisitos del usuario y negocio
Test Aceptación del Usuario
Requisitos del software
Test de Sistema
Especificación del programa
Test de Integración
Especificación de componentes
Test de Componentes
Programación
VERIFICACIÓN
VALIDACIÓN

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?



Sistema con 20 pantallas

Numero de dos dígitos, 101 valores posibles

Total de testing exhaustivo:

- 20 x 3 x 4 x 10 x 100 = 240.000

Suponiendo 1 seg por prueba:
4000 minutos -> 67 horas -> 8,5 días

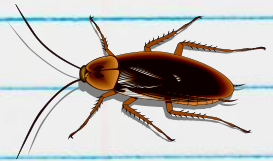
10 seg -> 17 semanas

1 min -> 1,4 años

10 min -> 13,7 años

Principios del Testing

- El testing muestra presencia de defecto.
- El testing exhaustivo es imposible
- Testing temprano
- Agrupamiento de Defectos
- Paradoja del Pesticida
- El testing es dependiente del contexto
- Falacia de la ausencia de errores

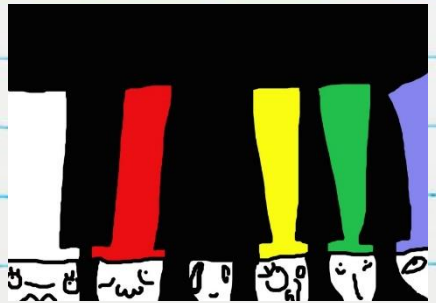


Principios del Testing

- 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

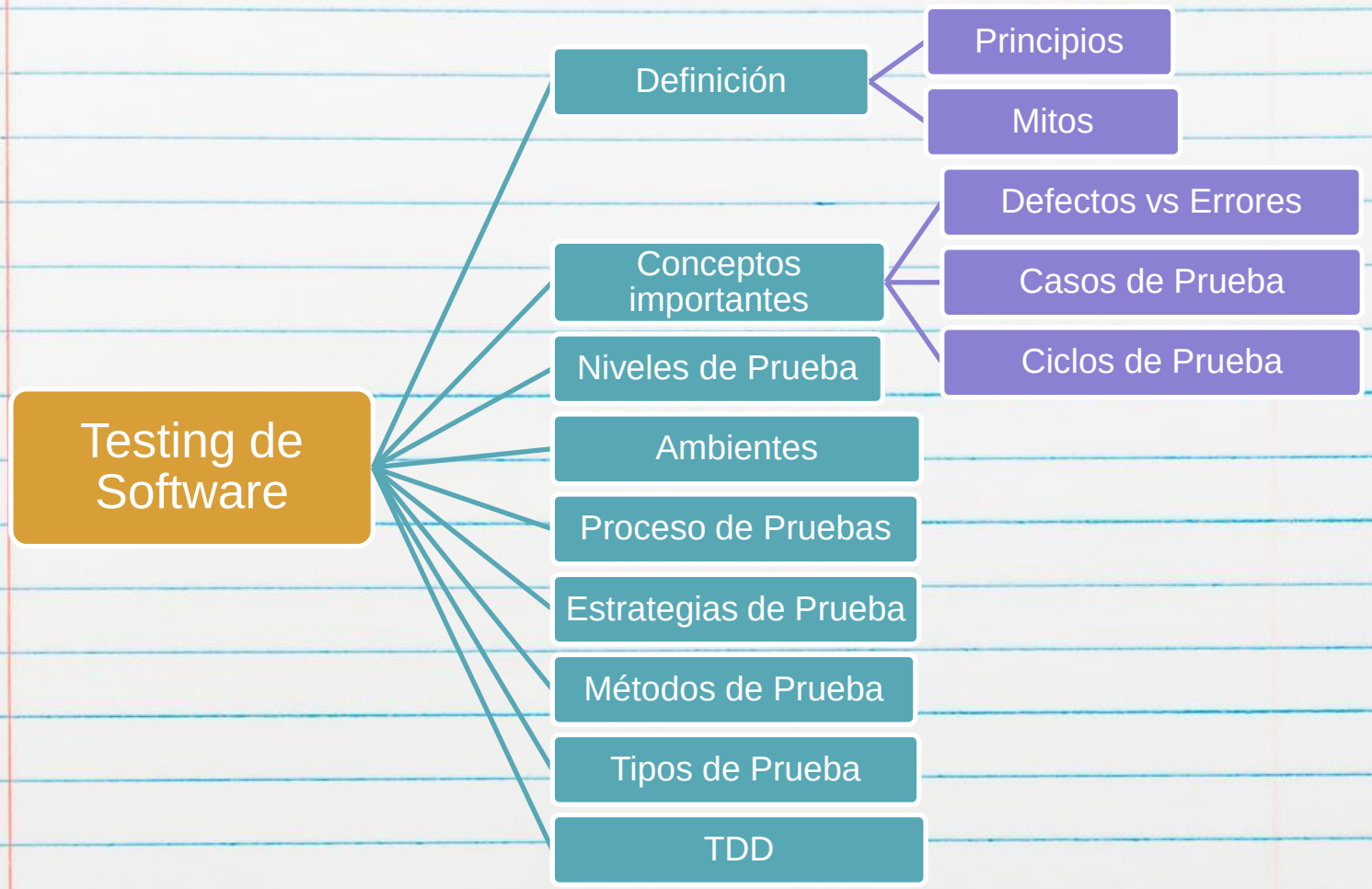


La Psicología del testing: Desarrolladores vs Testers

Conceptos: Smoke Test

- 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.

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

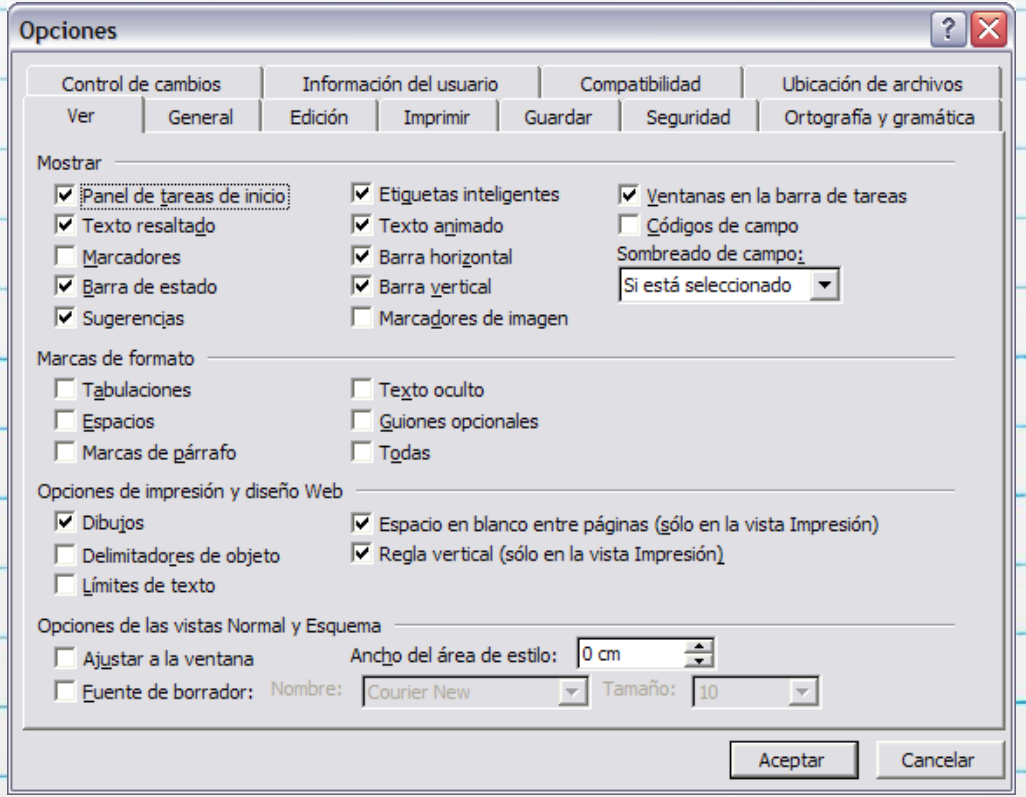
+

Muchas combinaciones

=

Más pruebas

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

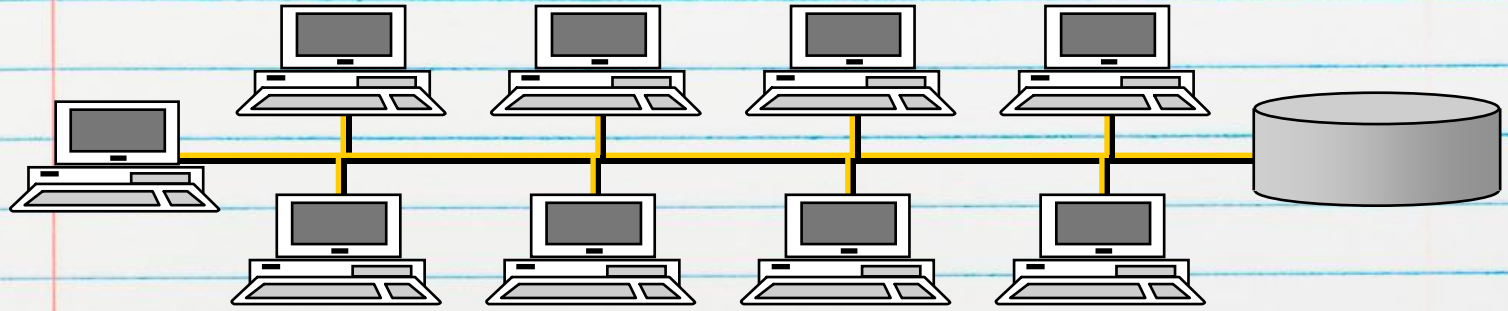


Las GUIs, son mucho más complejas que las interfaces basadas en caracteres

Prueba de Performance

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

- **Prueba de performance**
  - **Tiempo de respuesta**
  - **Concurrencia**

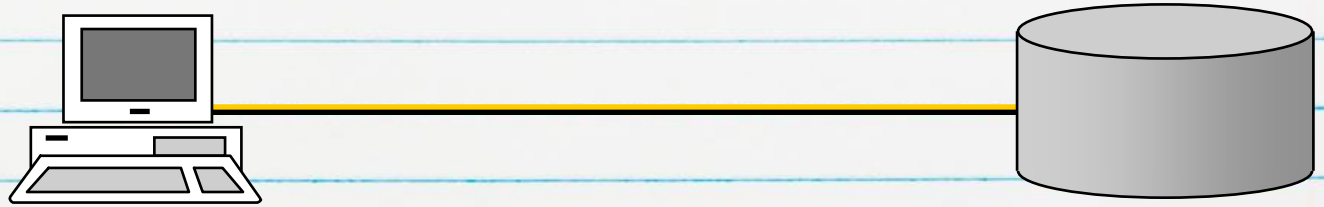


Prueba de Configuración

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

Aplicación Cliente

Servidor de Base de Datos



Cliente OS

TP monitor

Runtime

Mainframe conectividad

Server OS

Middleware

Network

E-mail

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

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"

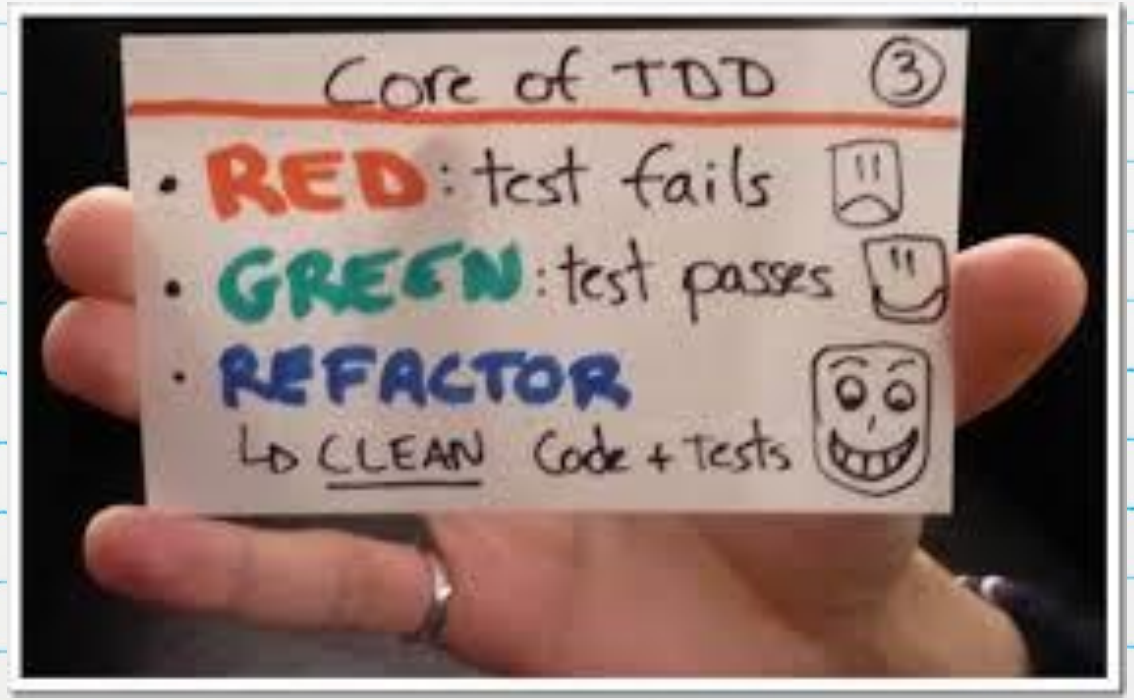
B. Beizer

"Test-Driven Development": Kent Beck. XP

TDD

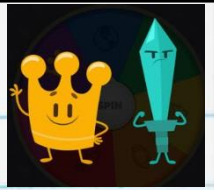
- Desarrollo guiado por pruebas de software, o Test-driven development (TDD)
- Es una técnica avanzada que involucra otras dos prácticas: *Escribir las pruebas primero (Test First Development)* y *Refactorización (Refactoring)*.
- Para escribir las pruebas generalmente se utilizan las pruebas unitarias

TDD



And repeat….

TDD: Quick quiz



¿Y si hacemos la clase “Multiplicación” usando TDD?

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