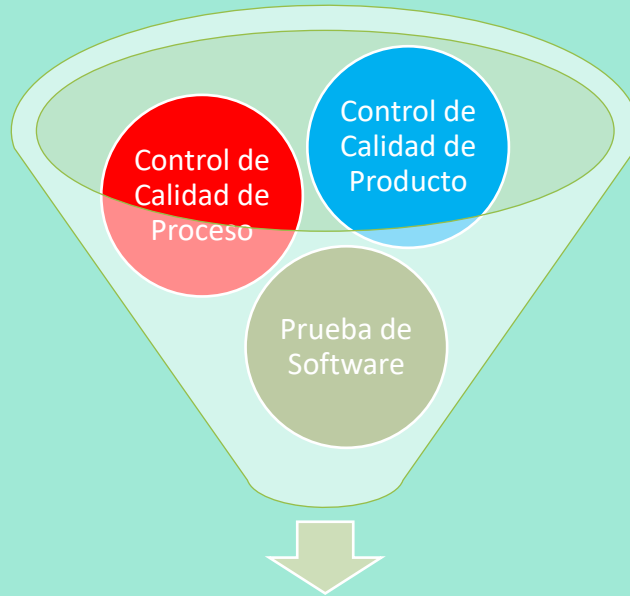


Universidad Tecnológica Nacional  
Facultad Regional Córdoba  
Cátedra de Ingeniería de Software  
Docentes: Judith Meles & Laura Covaro

# TESTING DE SOFTWARE

# Prueba de Software en Contexto...

Administración  
de  
Configuración



Aseguramiento de Calidad de  
Software

# Testing en el contexto:

## 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

# 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.

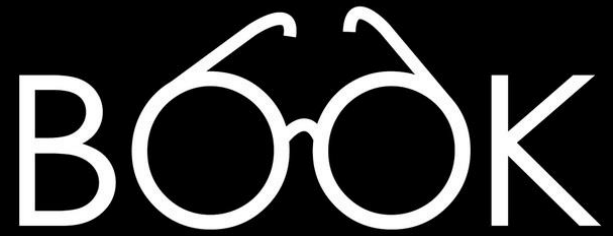


# Definición de Prueba de software

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  $\Rightarrow$  **es el que encuentra defectos!**
- Mundialmente: **30 a 50% del costo de un software confiable**

# ¿Qué es el testing?



- "The process consisting of all life cycle activities, both static and dynamic, concerned with planning, preparation and evaluation of software products and related work products to determine that they satisfy specified requirements, to demonstrate that they are fit for purpose and to detect defects." (ISTQB)
- "The process of operating a system or component under specified conditions, observing or recording the results, and making an evaluation of some aspect of the system or component." (IEEE Std 610-1990)
- "The process of analyzing a software item to detect the differences between existing and required conditions (that is, bugs) and to evaluate the features of the software items." (IEEE Std 829-1998)

# ¿Qué es el testing?



# 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

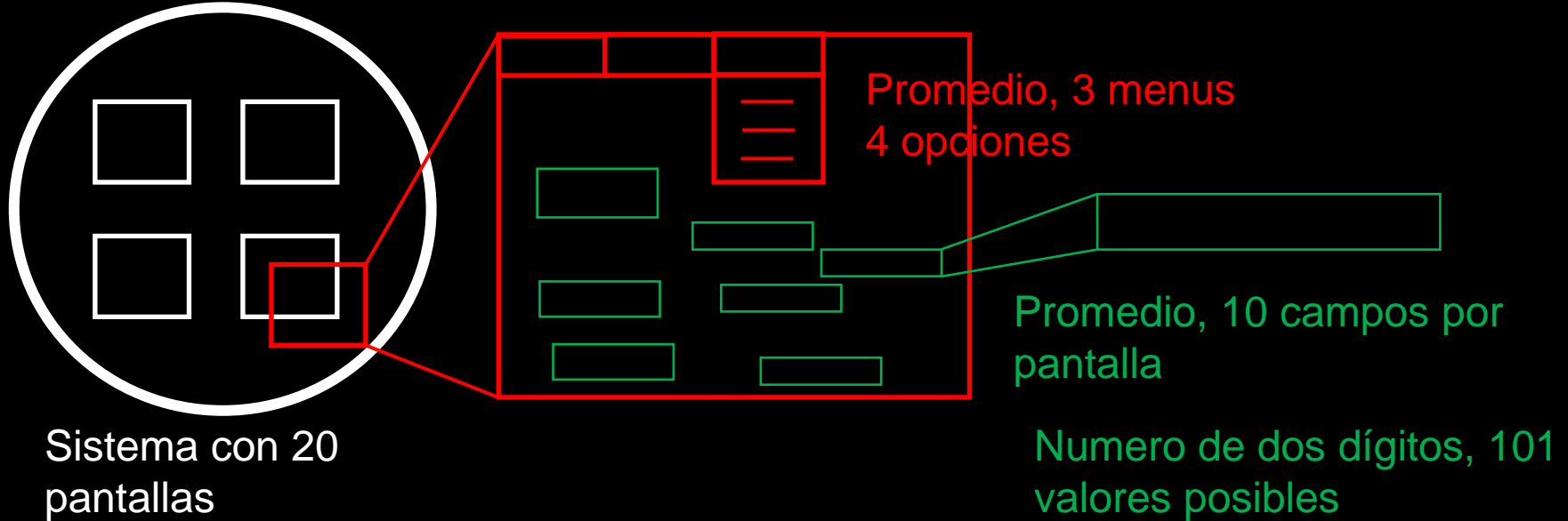




# Principios del Testing

- 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.

# ¿Cuánto testing es suficiente?



Sistema con 20  
pantallas

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?

- 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?

- 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

# 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: Error vs Defecto



# Conceptos: Defectos, severidad y prioridad

## Severidad

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

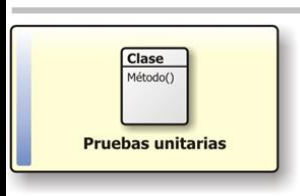
## Prioridad

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

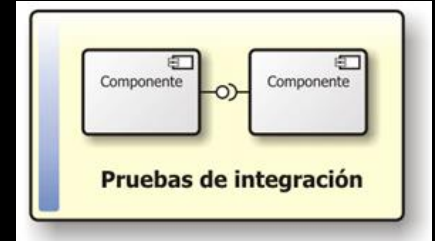


# NIVELES DE PRUEBA

## Pruebas unitarias



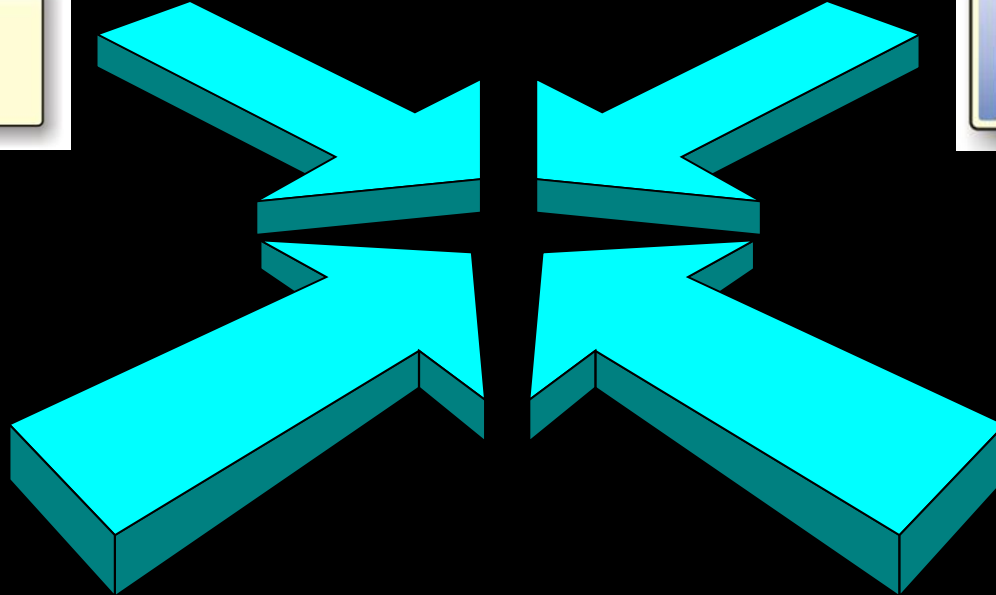
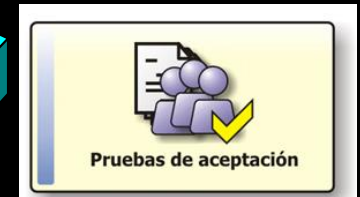
## pruebas de integración



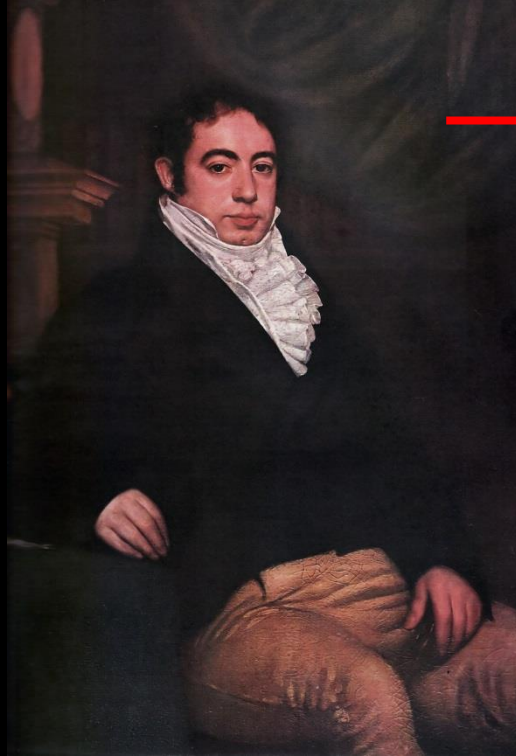
## Pruebas de sistema



## Pruebas de aceptación



# Niveles de Testing: Testing Unitario

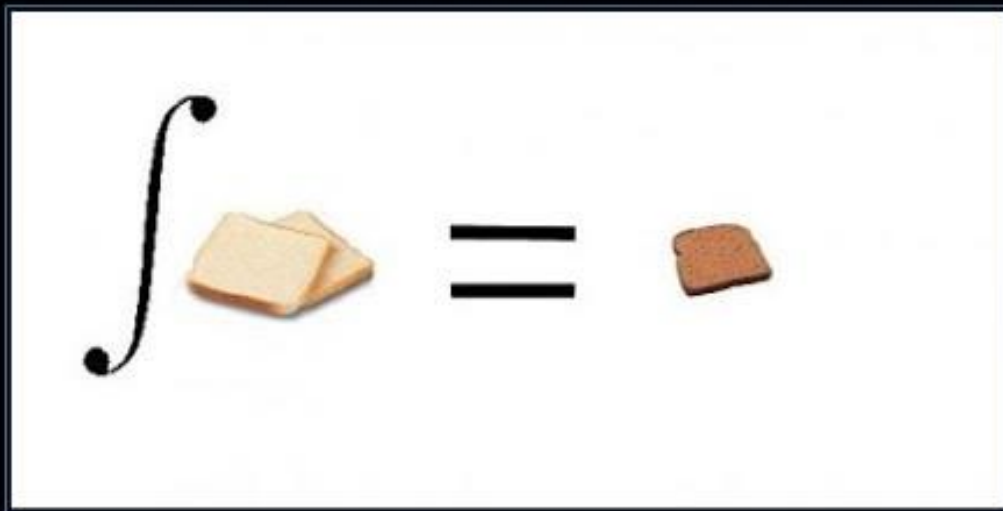


Bernardino Rivadavia

# Niveles de Testing: Testing Unitario

- Se prueba cada componente tras su realización/construcción.
- Solo se prueban componentes individuales.
- Cada componente es probado de forma independiente
- Se produce con acceso al código bajo pruebas y con el apoyo del entorno de desarrollo, tales como un framework de pruebas unitarias o herramientas de depuración.
- Los errores se suelen reparar tan pronto como se encuentran, sin constancia oficial de los incidentes.

# Niveles de Testing: Testing de Integración



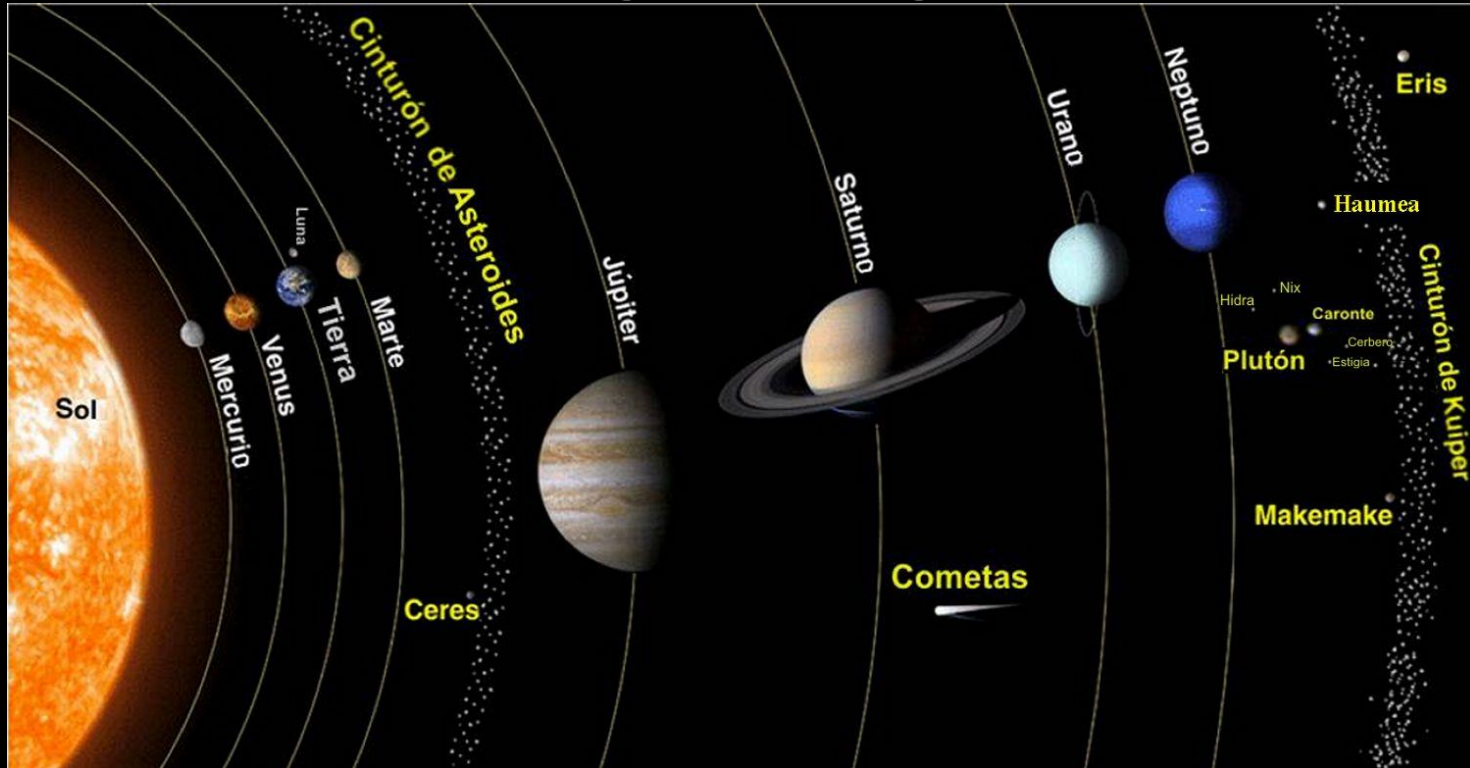
PAN INTEGRAL

Porque las ciencias se preocupan por tu regularidad

# Niveles de Testing: Testing de Integración

- Test orientado a verificar que las partes de un sistema que funcionan bien aisladamente, también lo hacen en conjunto
- Cualquier estrategia de prueba de versión o de integración debe ser incremental, para lo que existen dos esquemas principales:
  - Integración de arriba hacia abajo (top-down)
  - Integración de abajo hacia arriba (bottom-up).
- Lo ideal es una combinación de ambos esquemas.
- Tener en cuenta que los módulos críticos deben ser probados lo más tempranamente posible.
- Los puntos clave del test de integración son simples:
  - Conectar de a poco las partes más complejas
  - Minimizar la necesidad de programas auxiliares

# Niveles de Testing: Testing de Sistema



# Niveles de Testing: Testing de Sistema

- 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

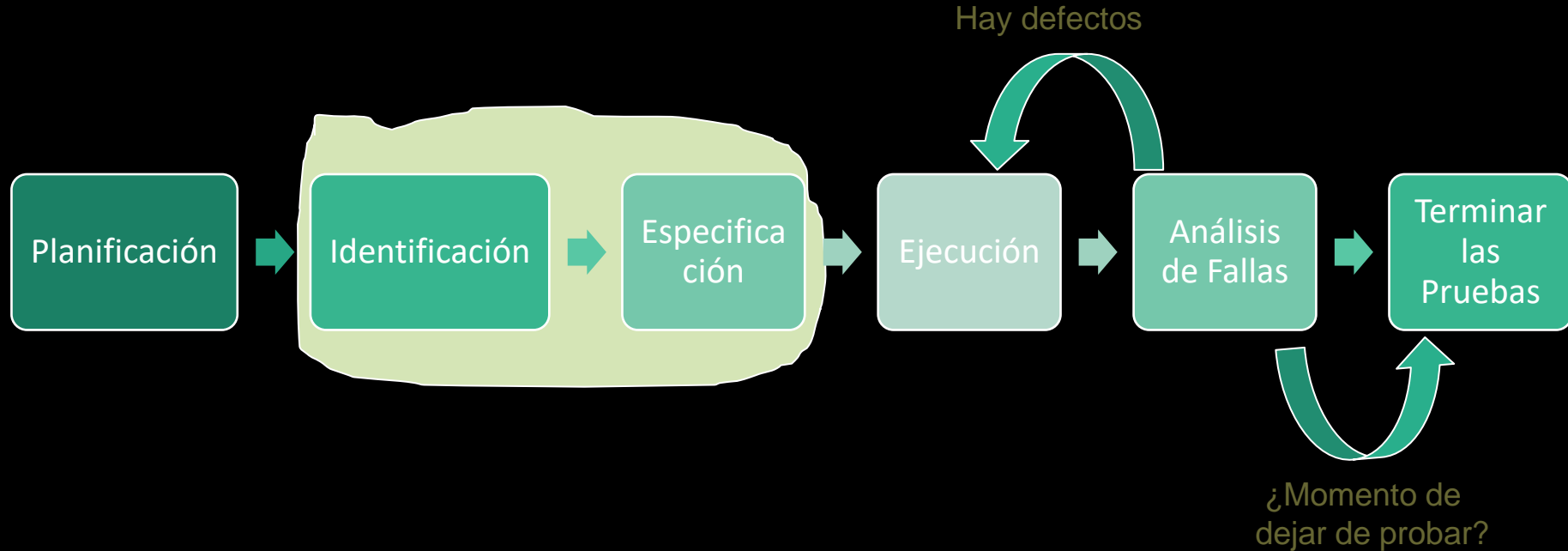




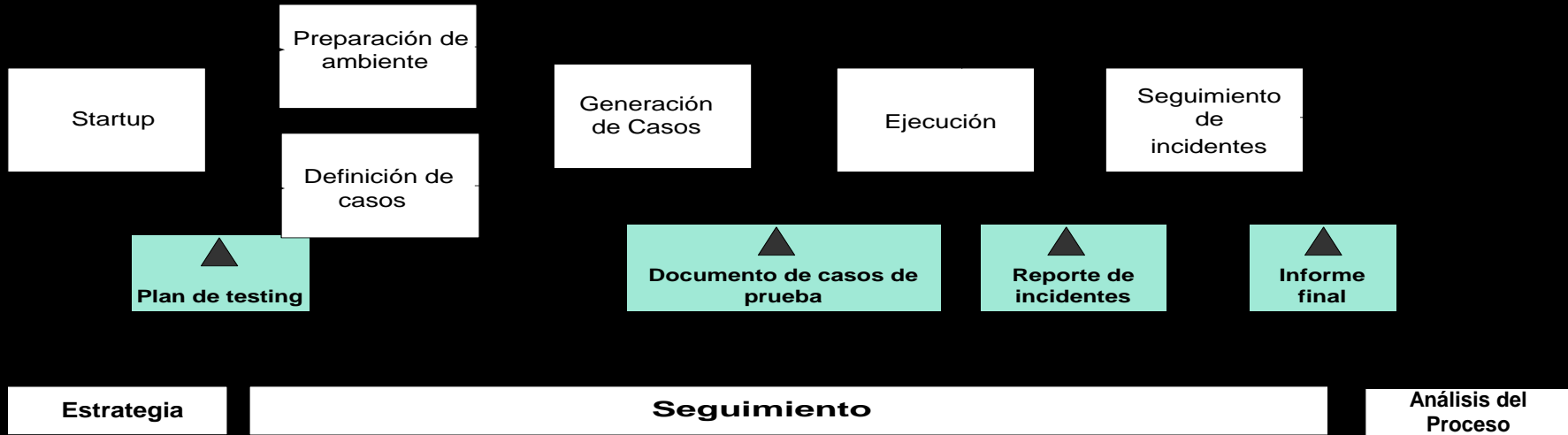
# Niveles de Testing: Testing de Aceptación

- 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).

# 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

Identificación y  
Especificación

- 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

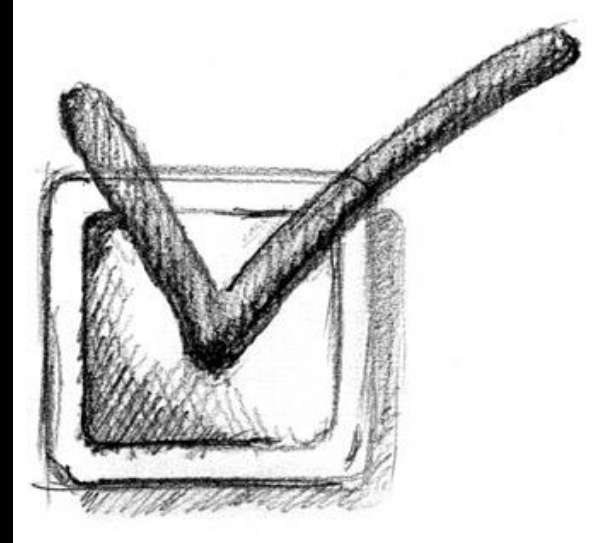
## Ejecución

- 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

## Evaluación y Reporte

- 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





# Proceso del Testing: Quick quiz

¿Y en Scrum? ¿Qué hacemos con este proceso?



# 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

# 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

# CASOS DE PRUEBA

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

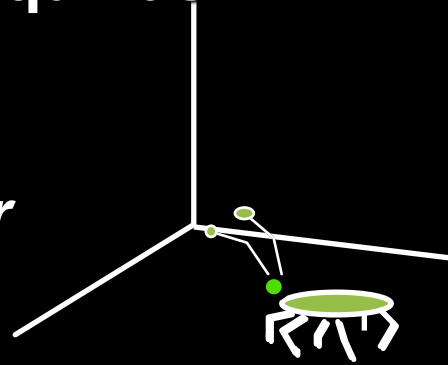
Consta de tres partes:

- 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.***

# 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

# 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.





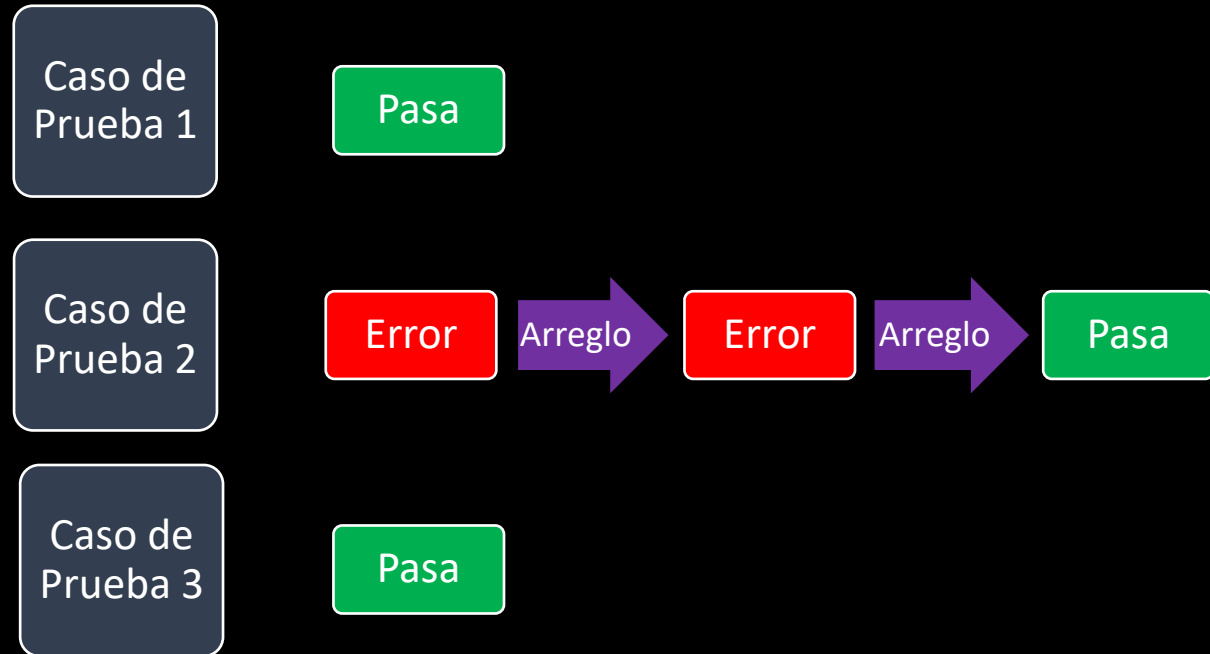
# Conceptos: Regresión

- 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.



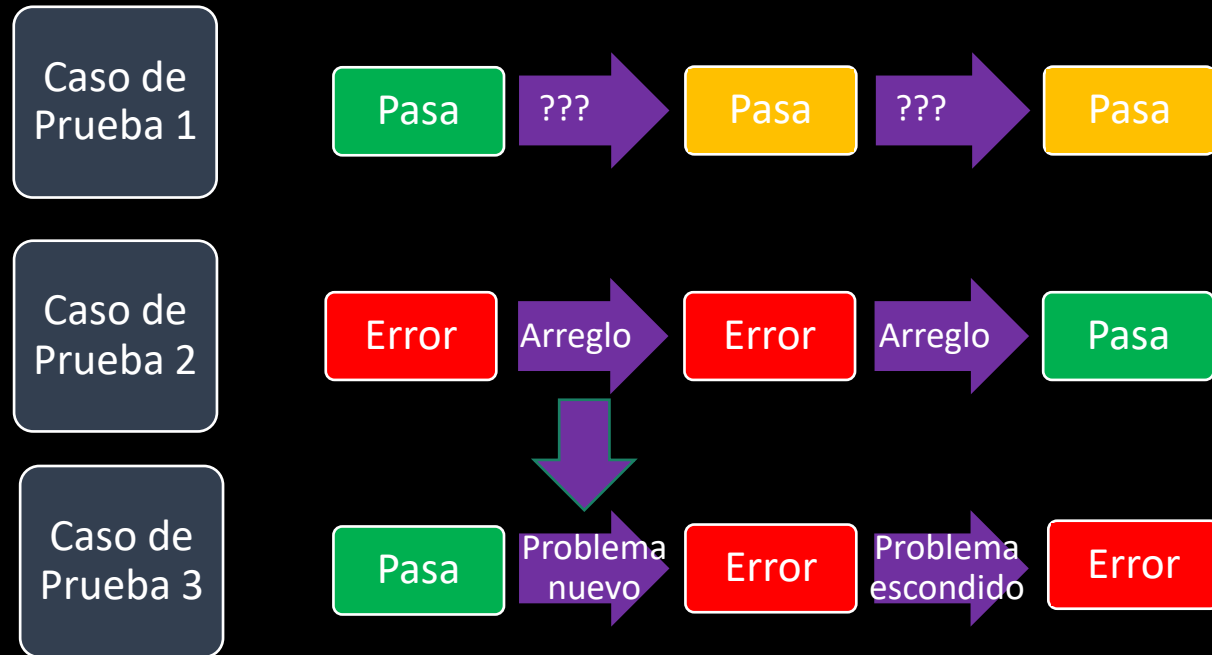
# Conceptos: Regresión

- Sin regresión



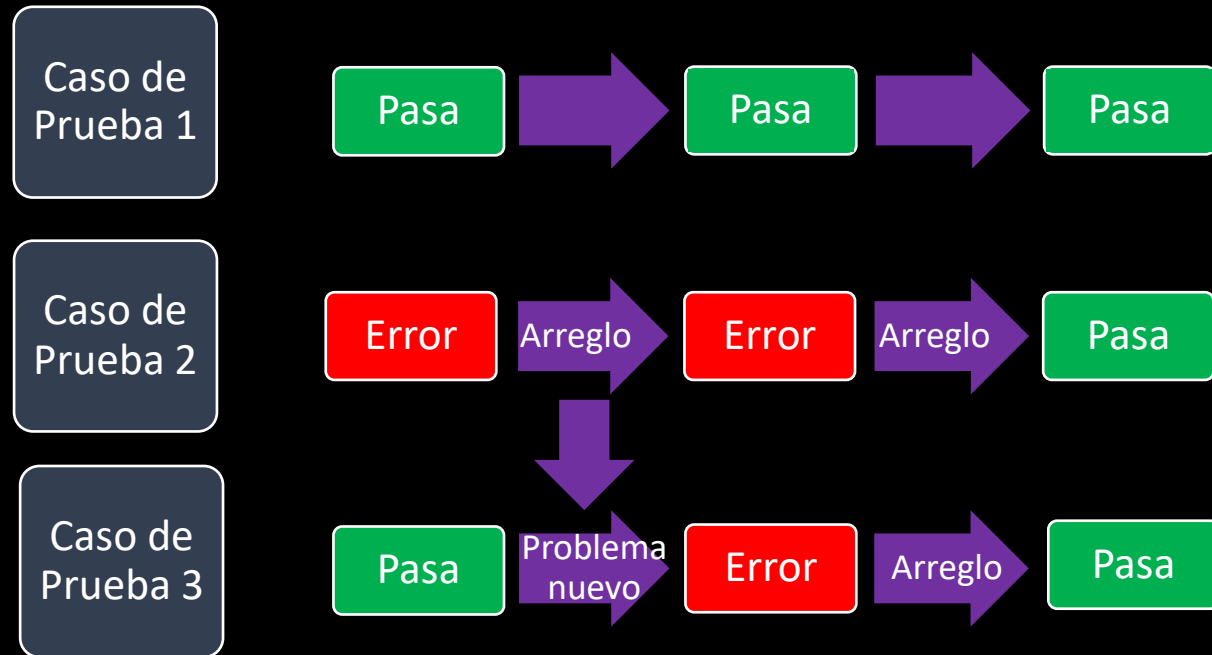
# Conceptos: Regresión

- Sin regresión



# Conceptos: Regresión

- Con regresión



# 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.



# 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

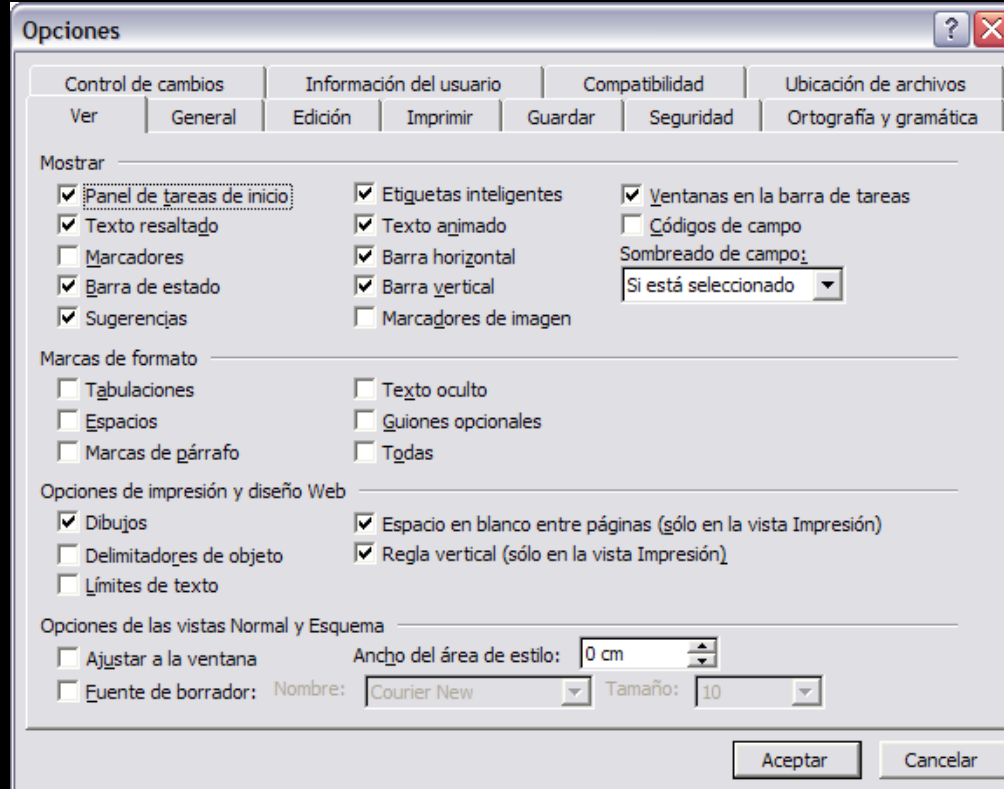
- Es la prueba de “cómo” funciona el sistema
- NO HAY QUE OLVIDARLAS!!!! Los requerimientos no funcionales son tan importantes como los funcionales
  - Performance Testing
  - Pruebas de Carga
  - Pruebas de Stress
  - Pruebas de usabilidad,
  - Pruebas de mantenimiento
  - Pruebas de fiabilidad
  -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



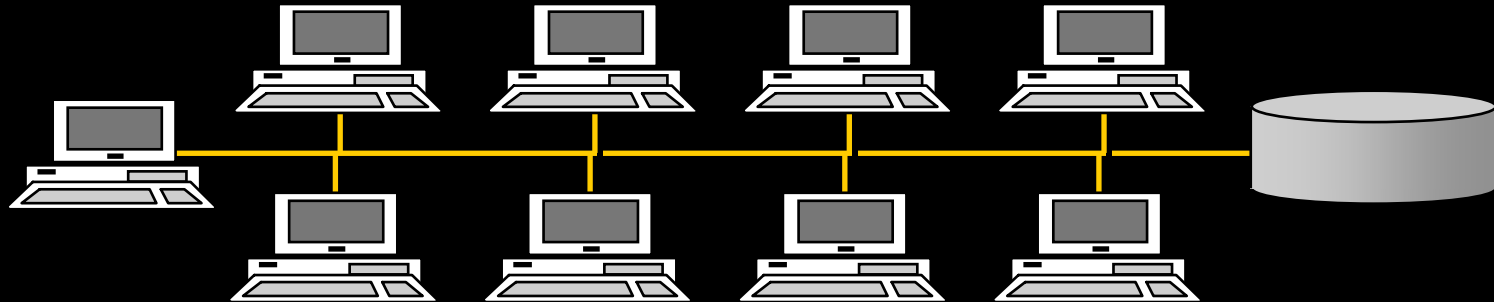
*Las GUIs,  
son mucho  
más  
complejas  
que las  
interfaces  
basadas en  
caracteres*



# Prueba de Performance

- Prueba de performance
  - Tiempo de respuesta
  - Concurrencia

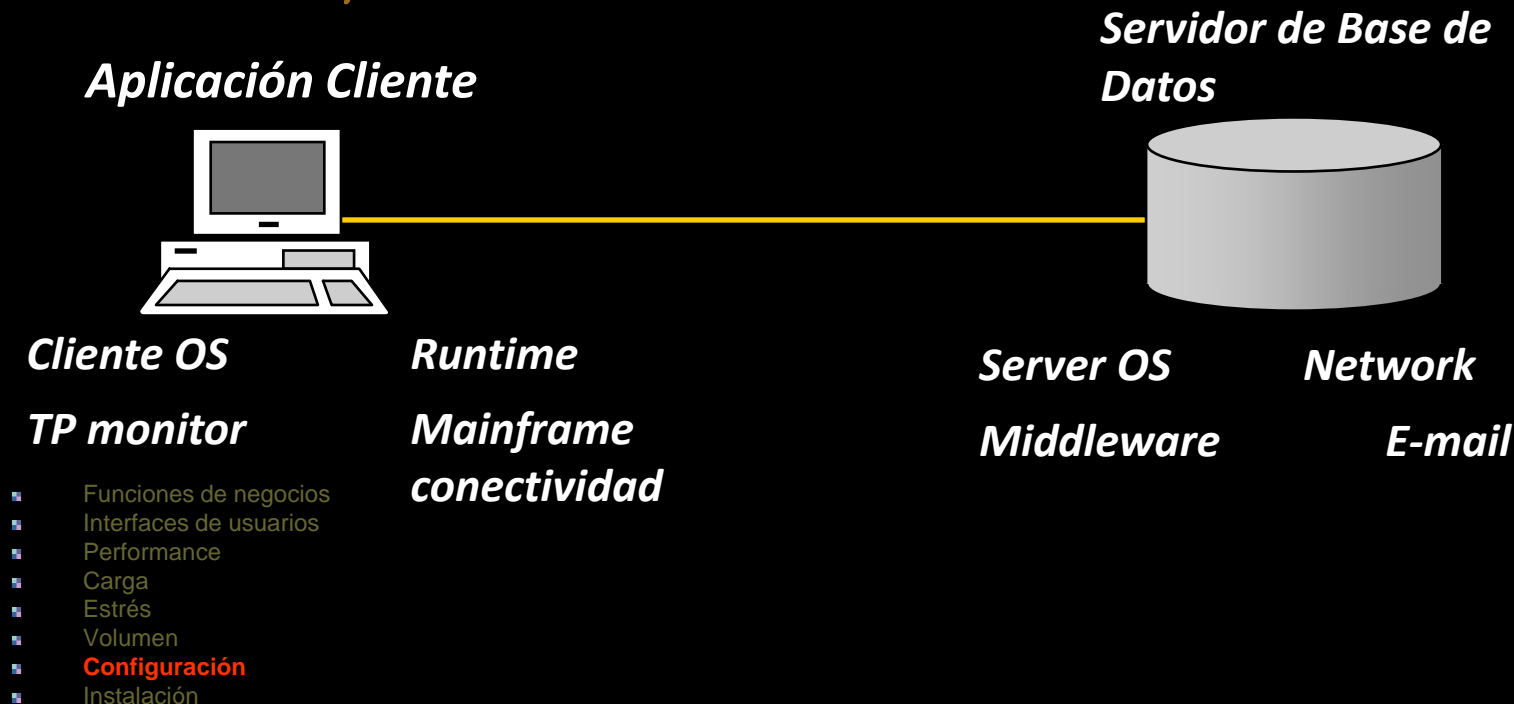
- Funciones de negocios
- Interfaces de usuarios
- **Performance**
- Carga
- Estrés
- Volumen
- Configuración
- Instalación



## ❑ *Prueba de Configuración*

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

# Prueba de Configuración



# El Testing y el Ciclo de Vida



# Verificación y Validación

## **VERIFICACION**

¿Estamos construyendo el sistema correctamente?

## **VALIDACION**

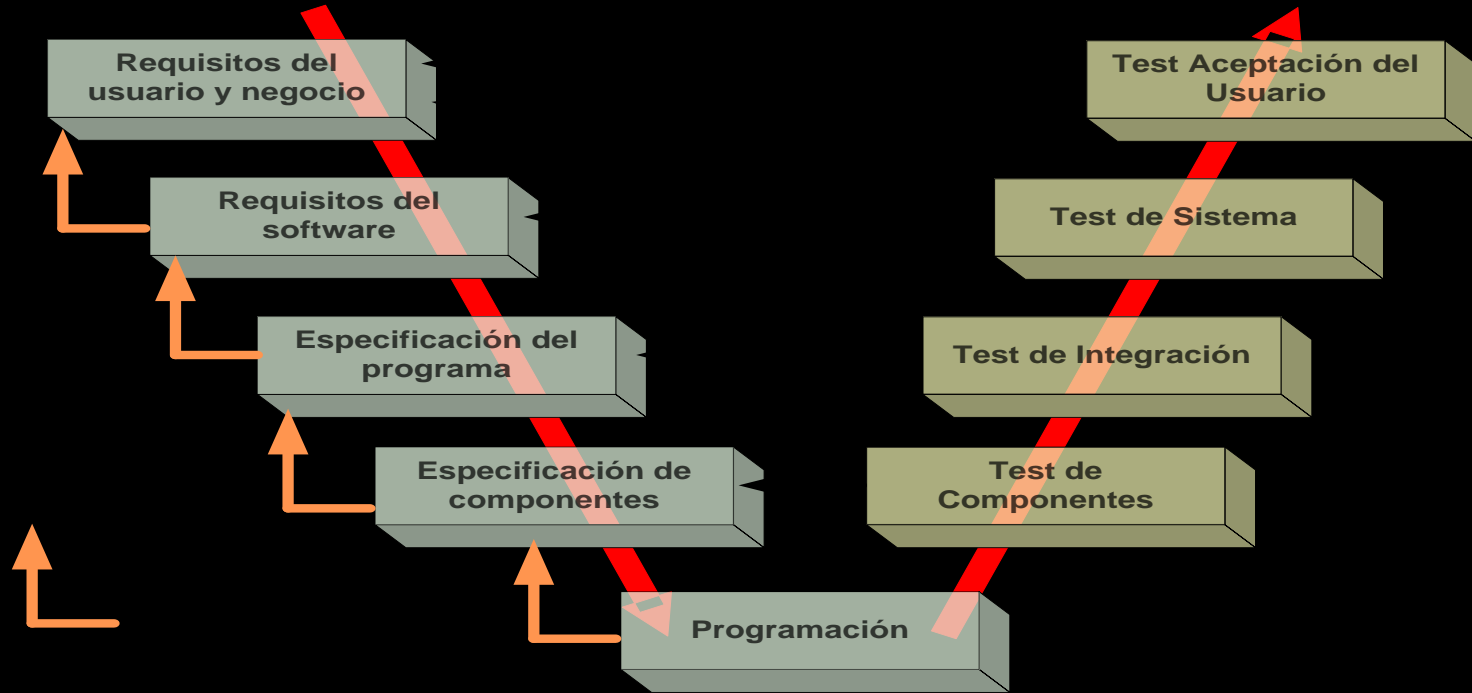
¿Estamos construyendo el sistema correcto?

# 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



# 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*

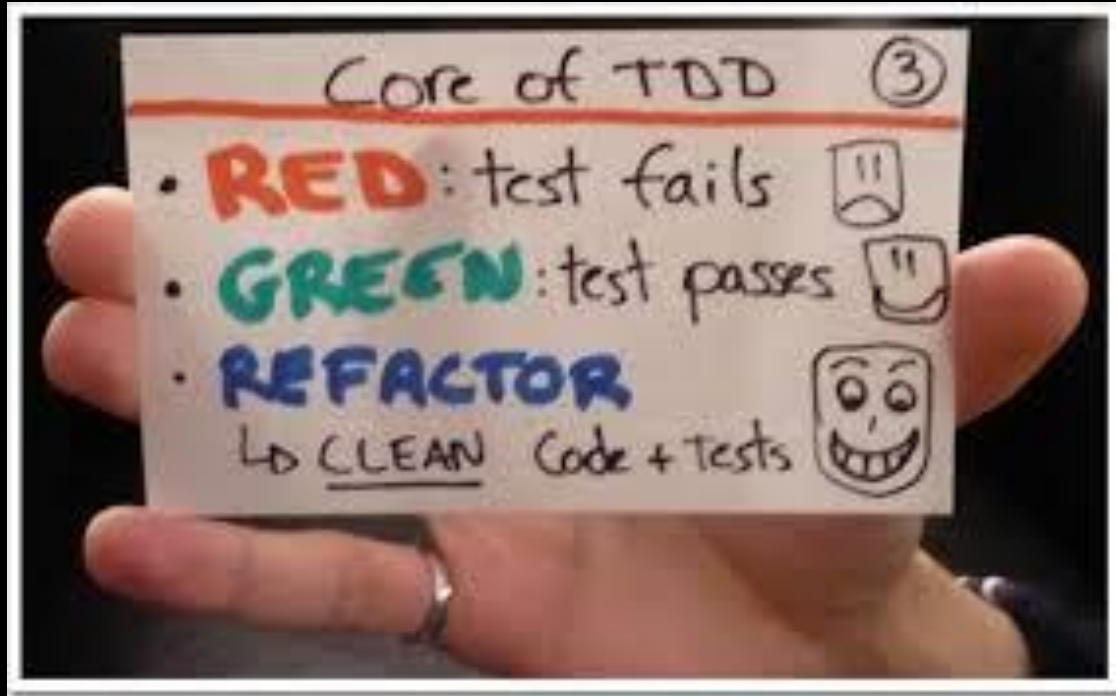
*“Test-Driven Development”: Kent Beck. XP*

# 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