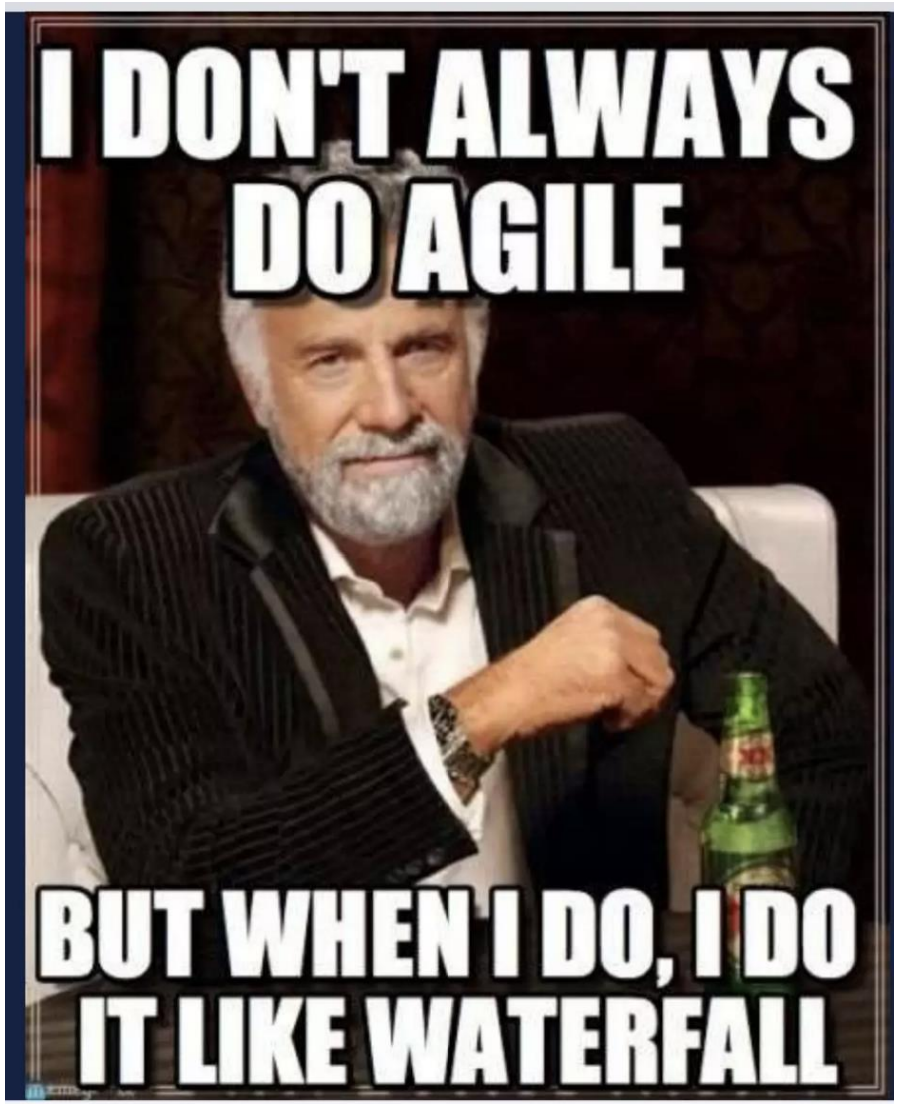
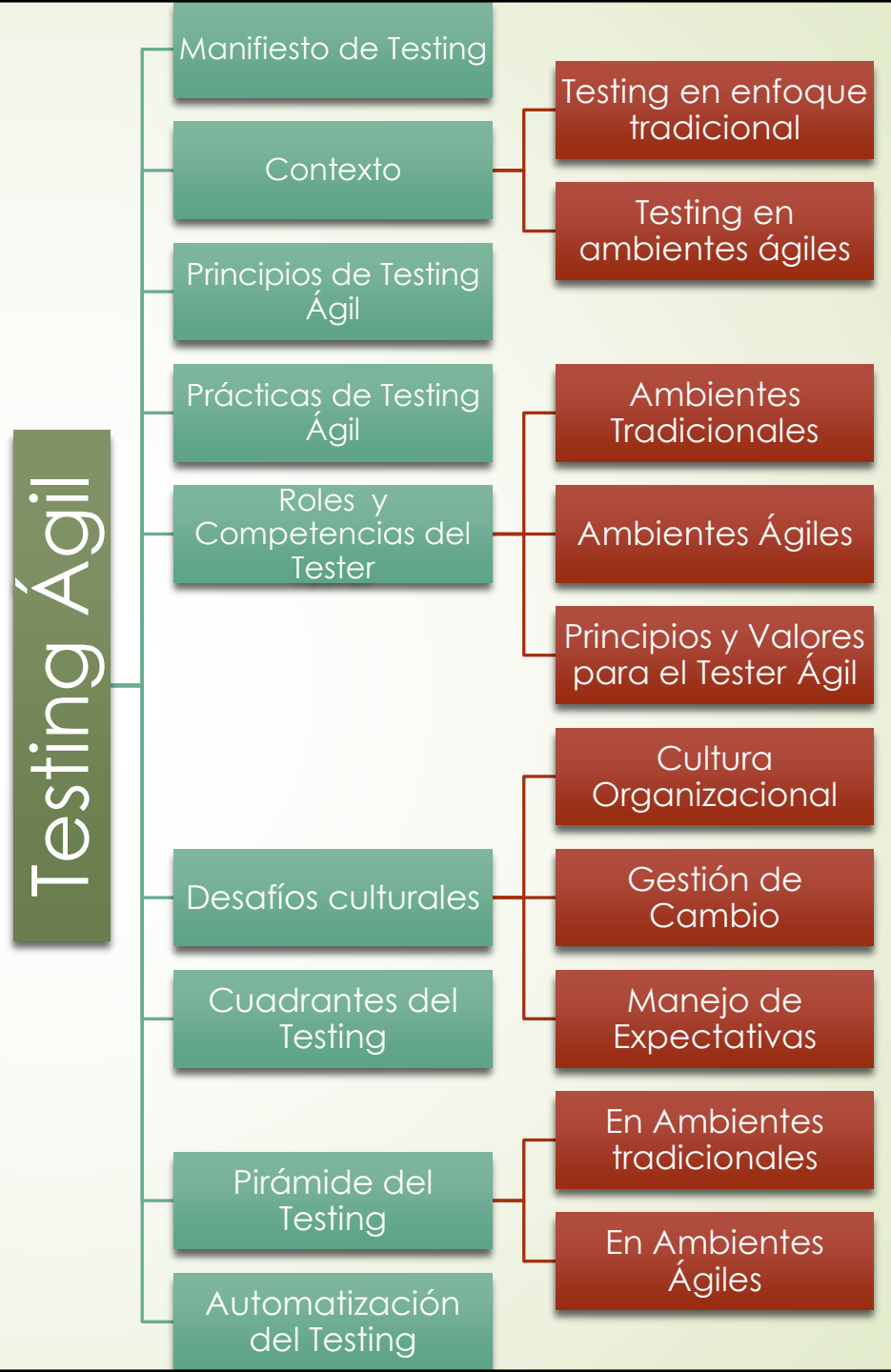
Universidad Tecnológica Nacional
Facultad Regional Córdoba
Cátedra de Ingeniería de Software
Docentes: Judith Meles & Laura Covaro

Testing en Ambientes Ágiles

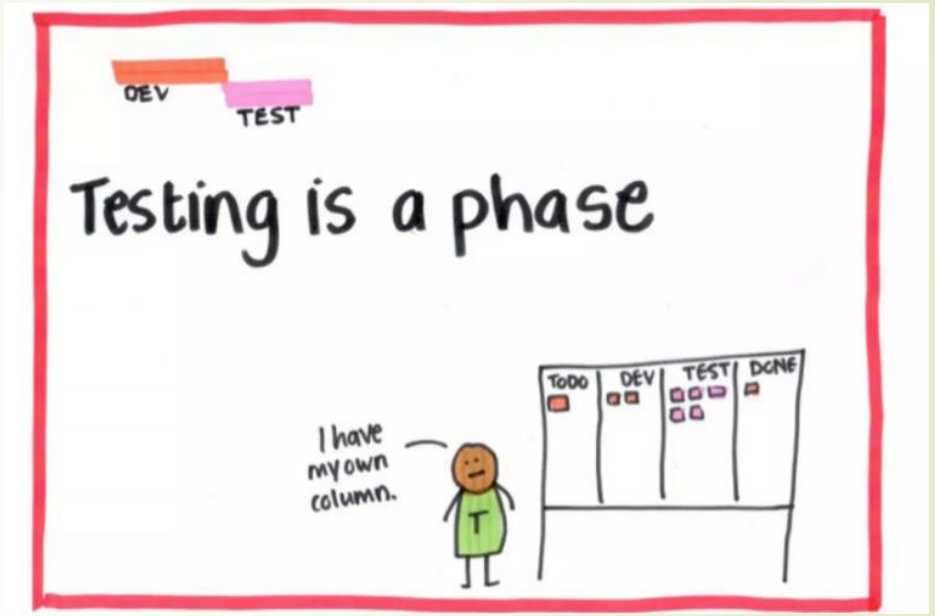
Judith Meles – Laura Covaro


I DON'T ALWAYS
DO AGILE
BUT WHEN I DO, I DO
IT LIKE WATERFALL

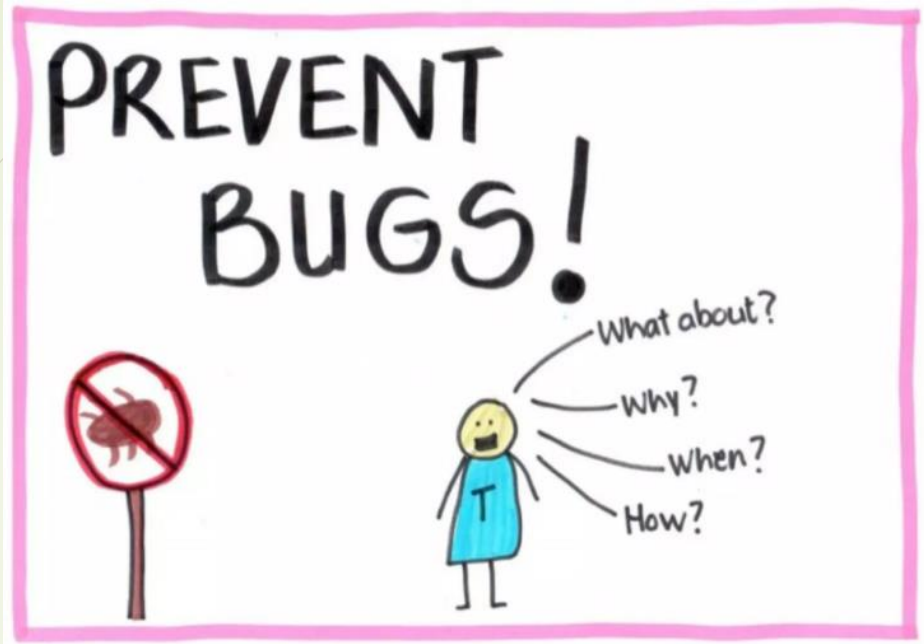
Testing Ágil: Agenda



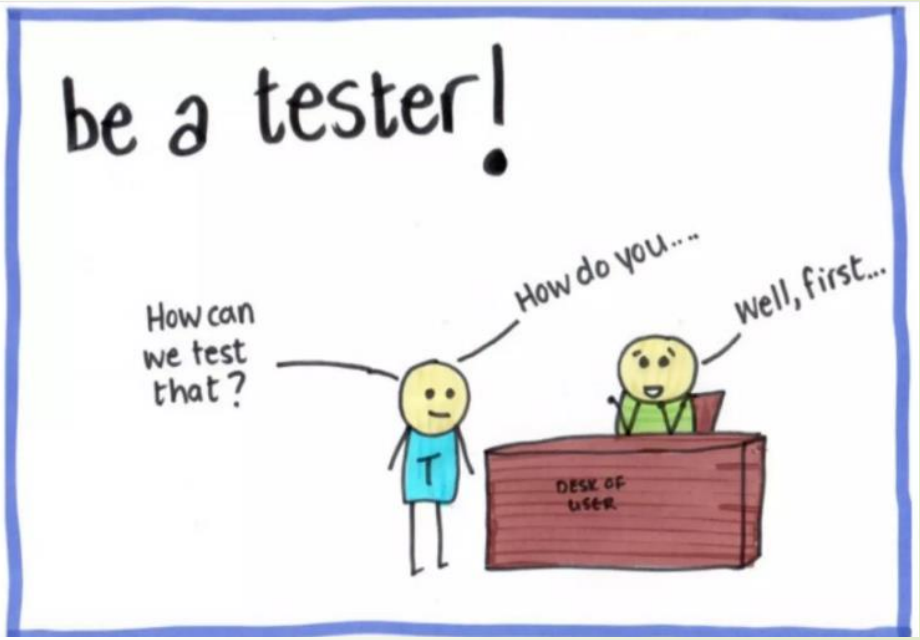
Testing Manifesto



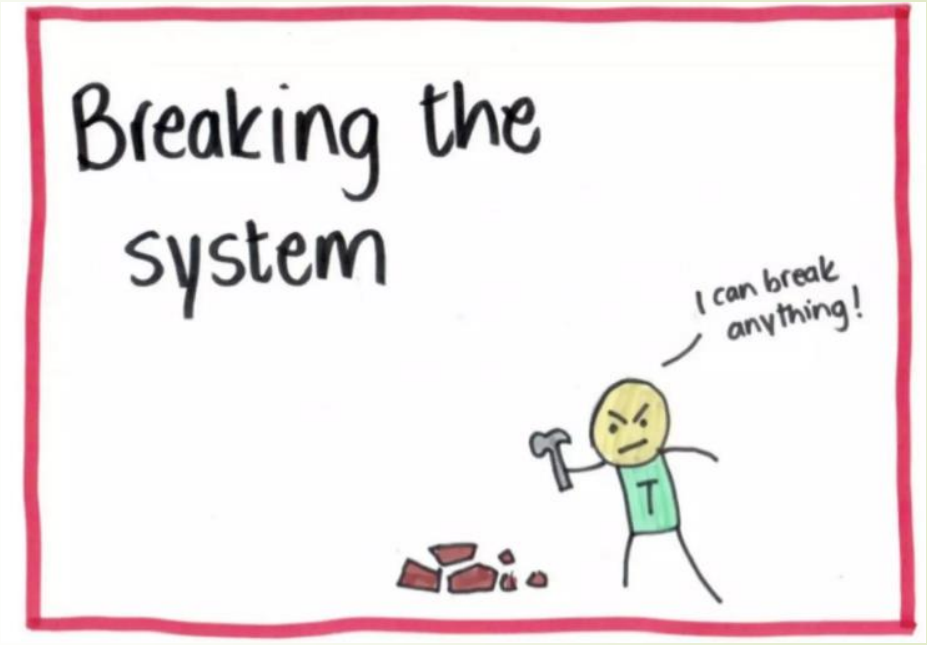
Valores Testing Manifiesto



Valores Testing Manifiesto



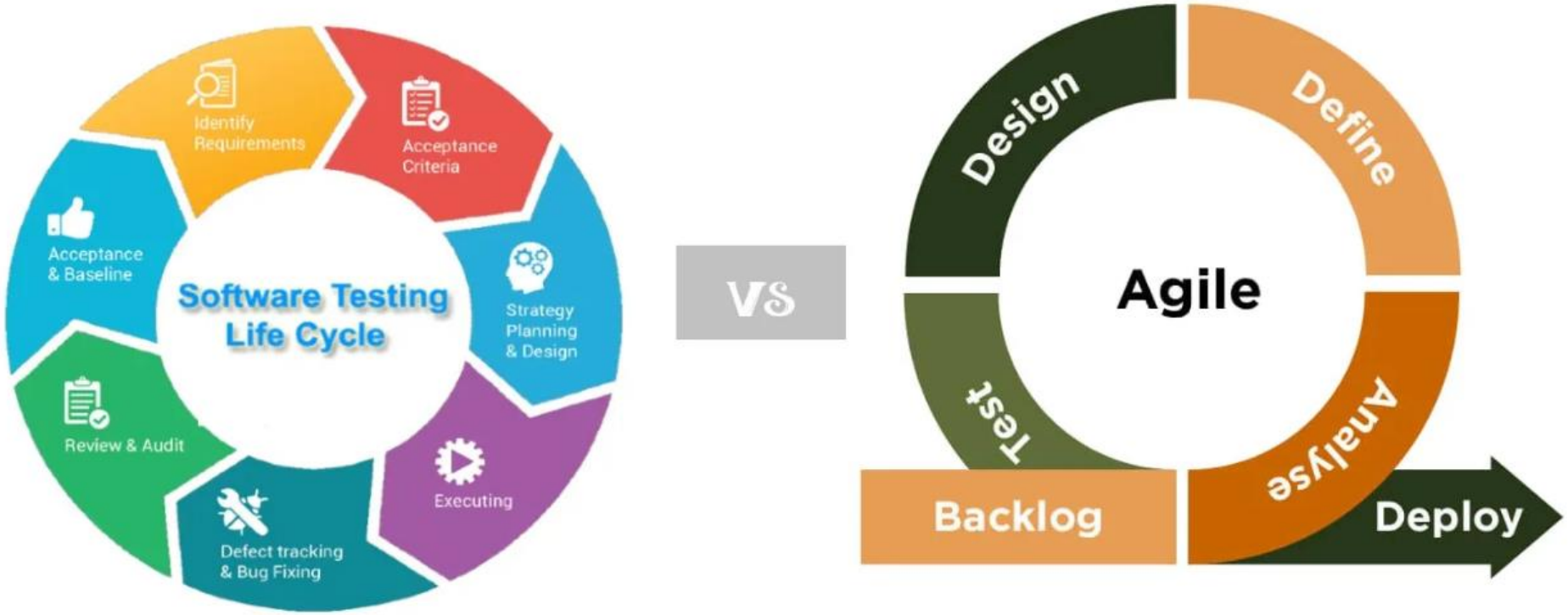
Valores Testing Manifiesto

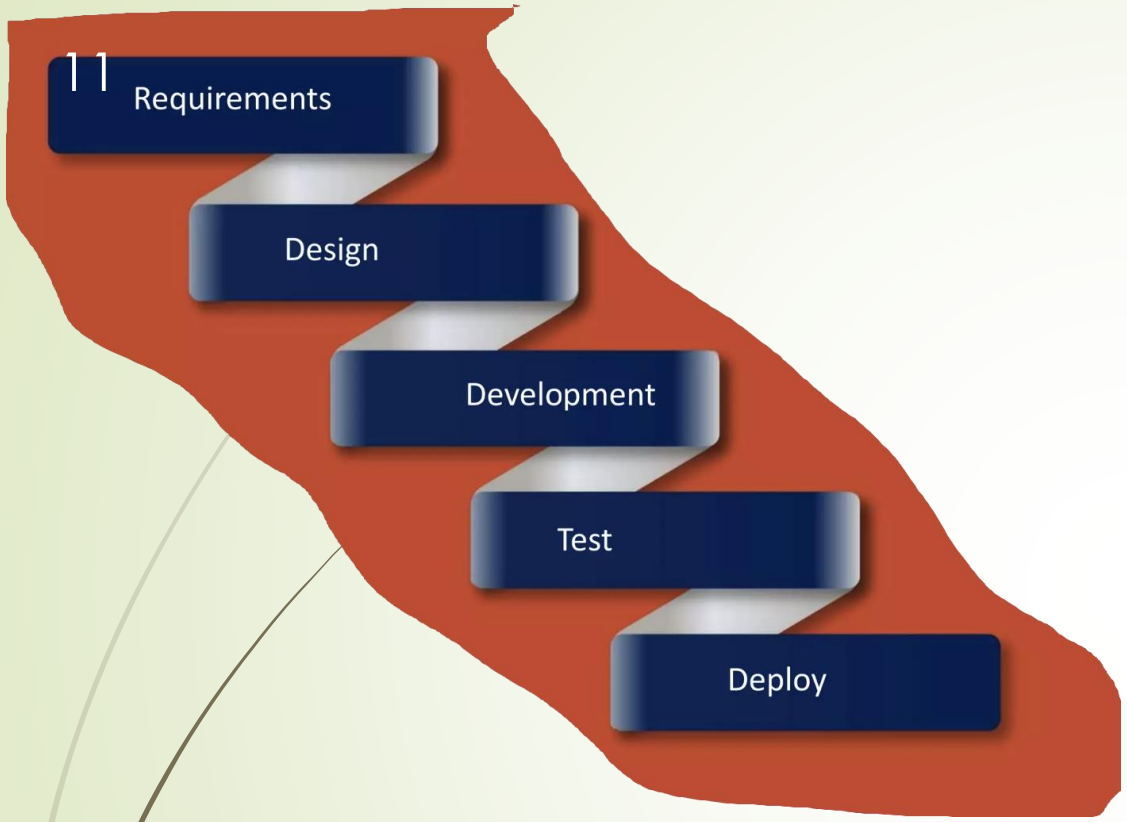


Valores Testing Manifiesto

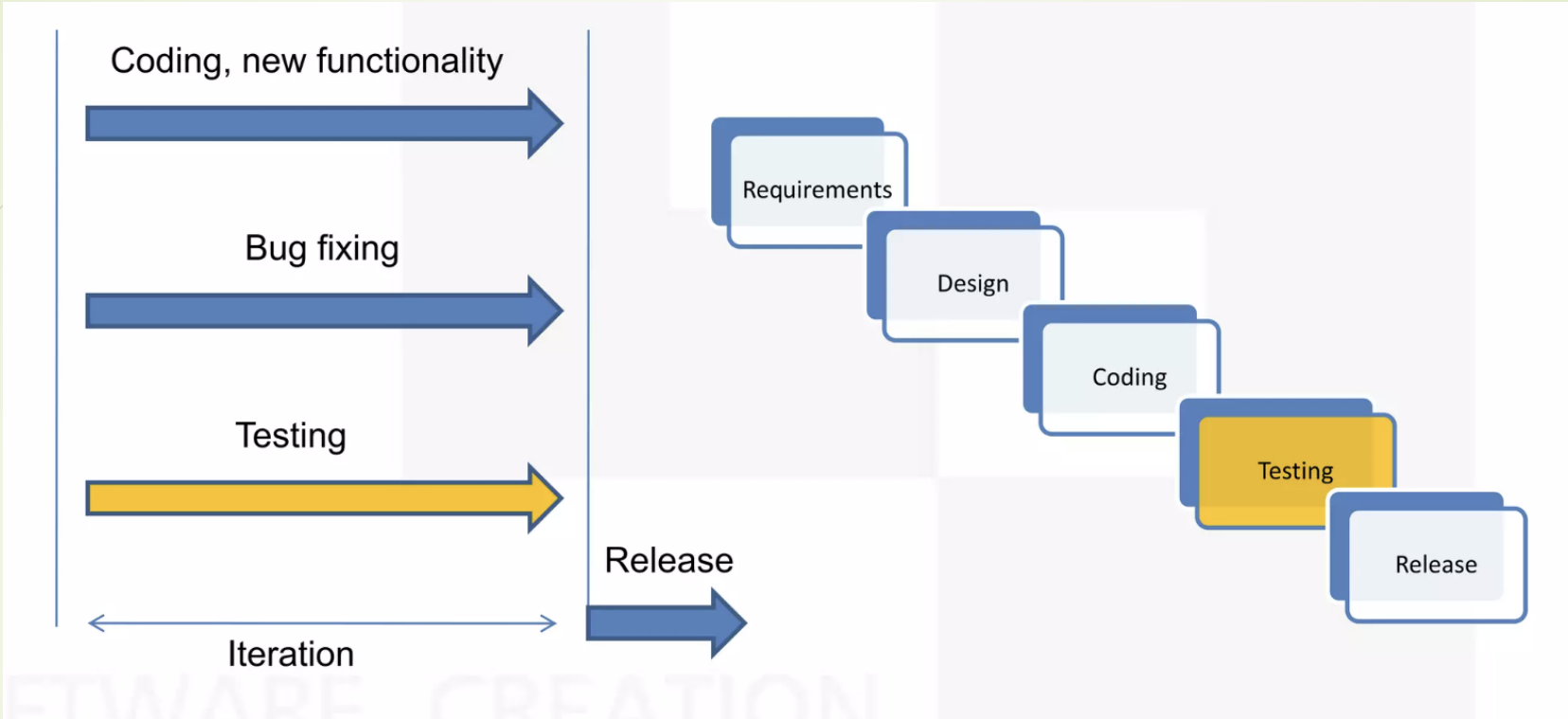
Valores Testing Manifiesto

Agile vs Software Testing Lifecycle





Enfoque Tradicional (cascada) vs. Agile

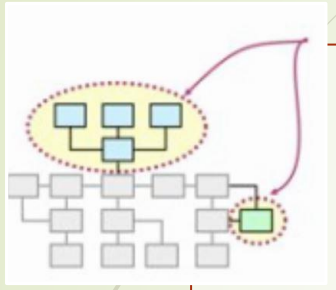


Enfoque Tradicional (cascada) vs. Agile

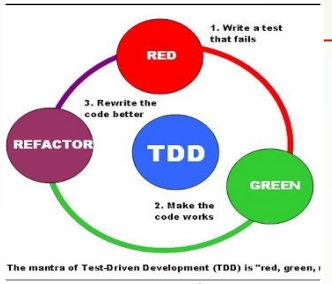
9 Principios para Testing en Proyectos Ágiles

- Testing se mueve hacia adelante en el proyecto
- Testing no es una fase
- Todos hacen testing
- Reducir la latencia del feedback
- Las pruebas representan expectativas
- Mantener el código limpio, corregir los defectos rápido
- Reducir la sobrecarga de documentación de las pruebas
- Las pruebas son parte del “done”
- De probar al final a Conducido por Pruebas

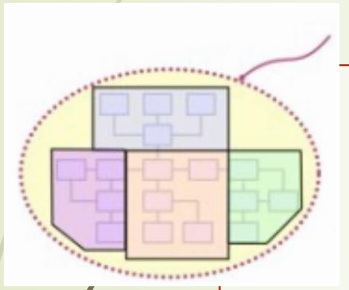
6 Prácticas Concretas



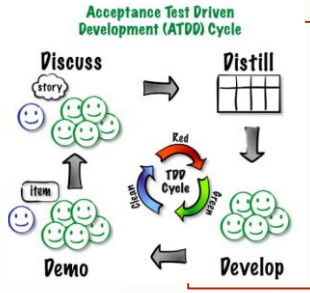
Pruebas de unidad e integración automatizadas



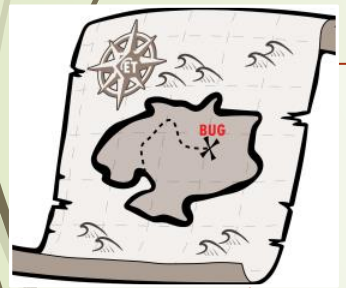
TDD (Test Driven Development)



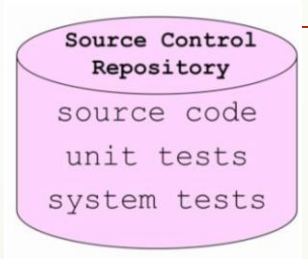
Pruebas de Regresión a nivel de sistema automatizadas



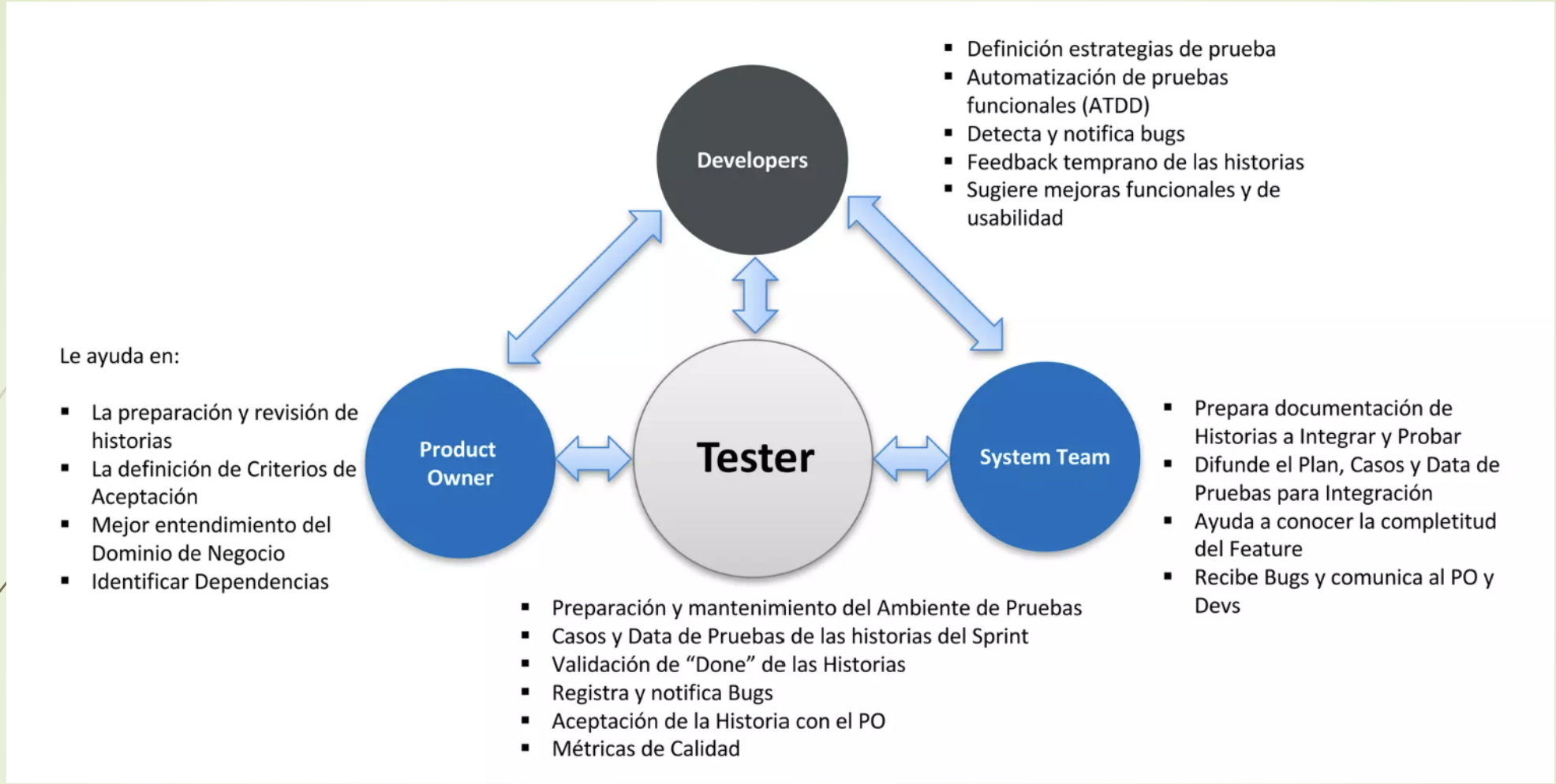
ATDD (Desarrollo conducido por Pruebas de aceptación



Pruebas exploratorias



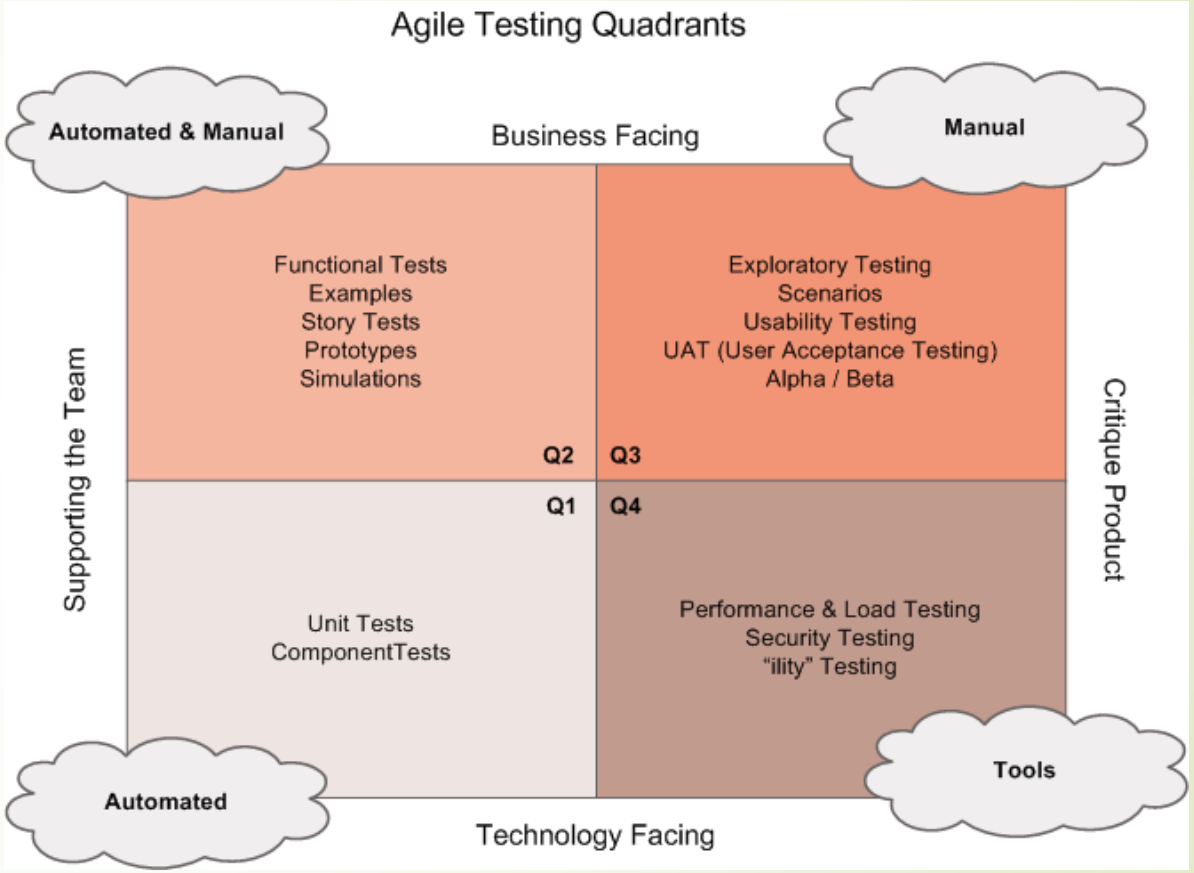
Control de versión de las pruebas con el código



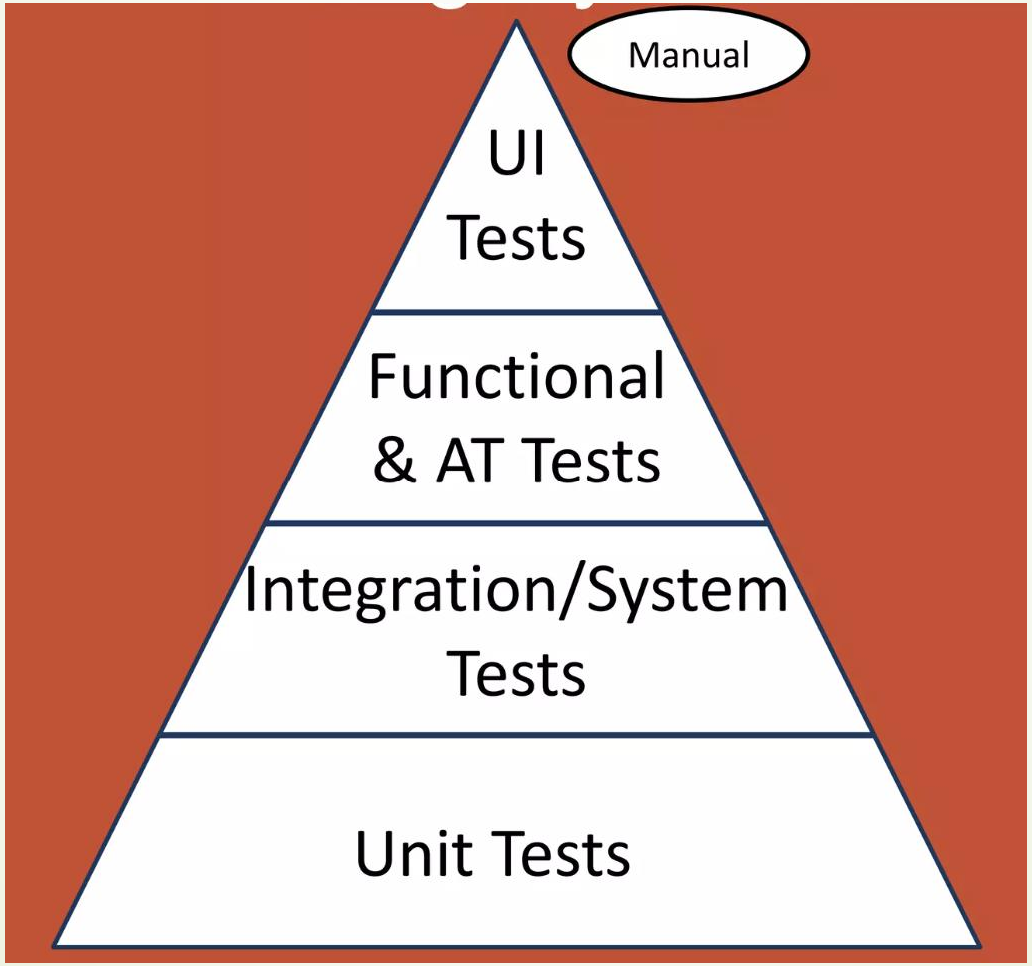
El rol del Tester en los equipos ágiles

Cambios culturales

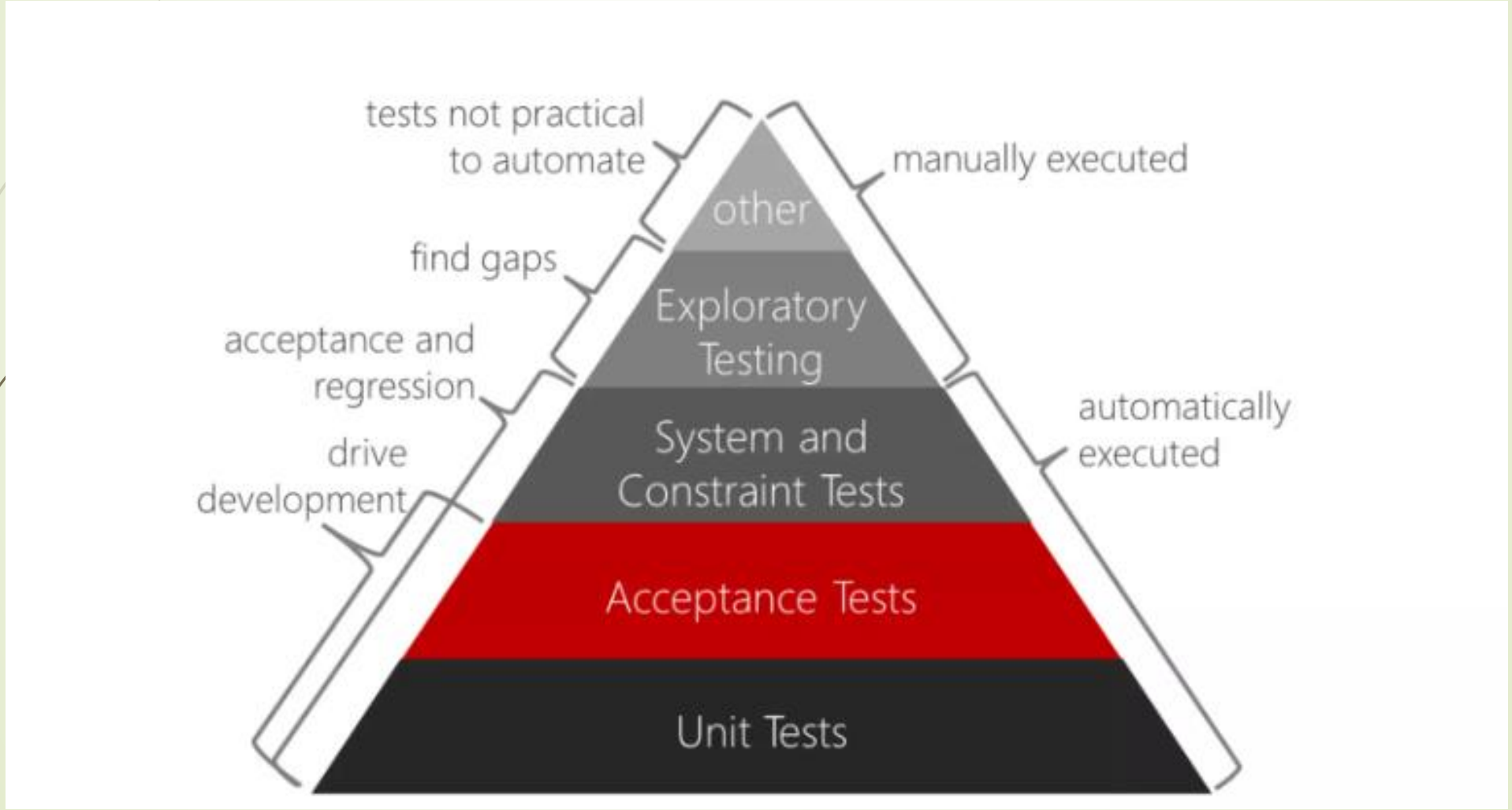
Cuadrantes de Testing Ágil



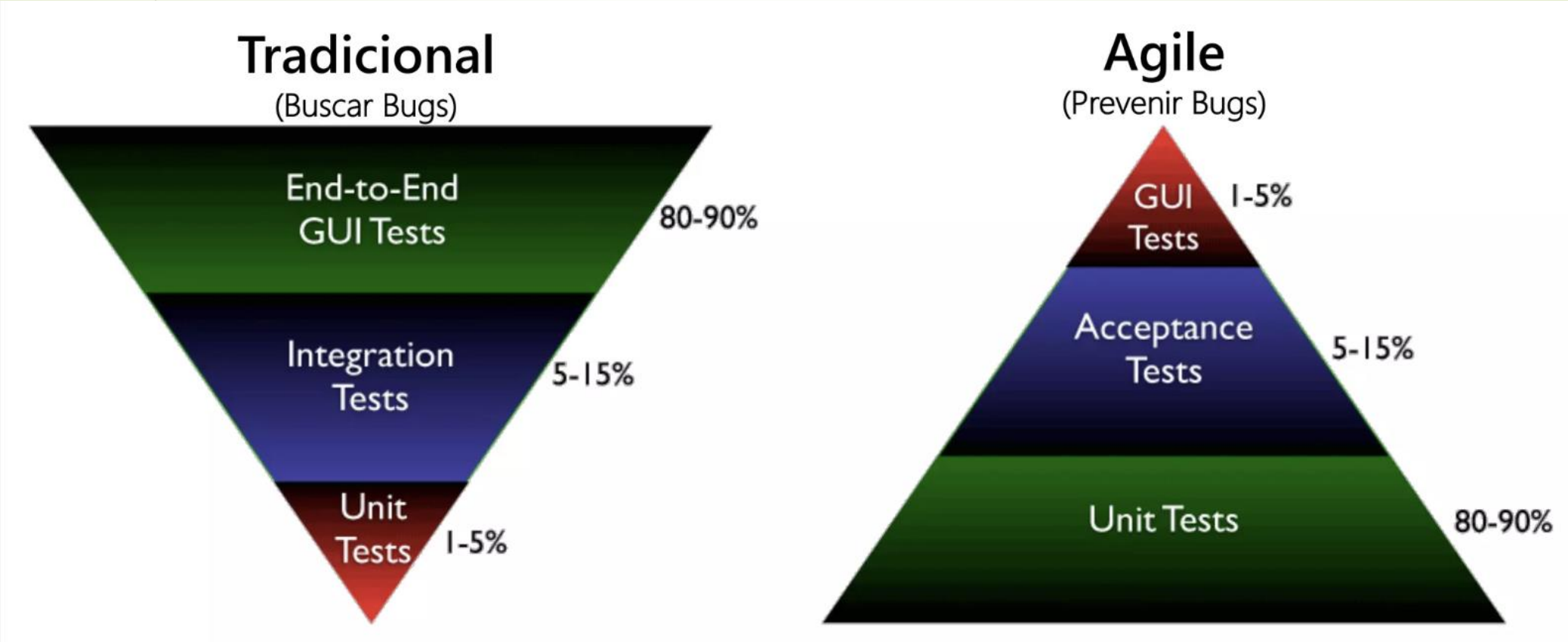
Pirámide de Testing Agile



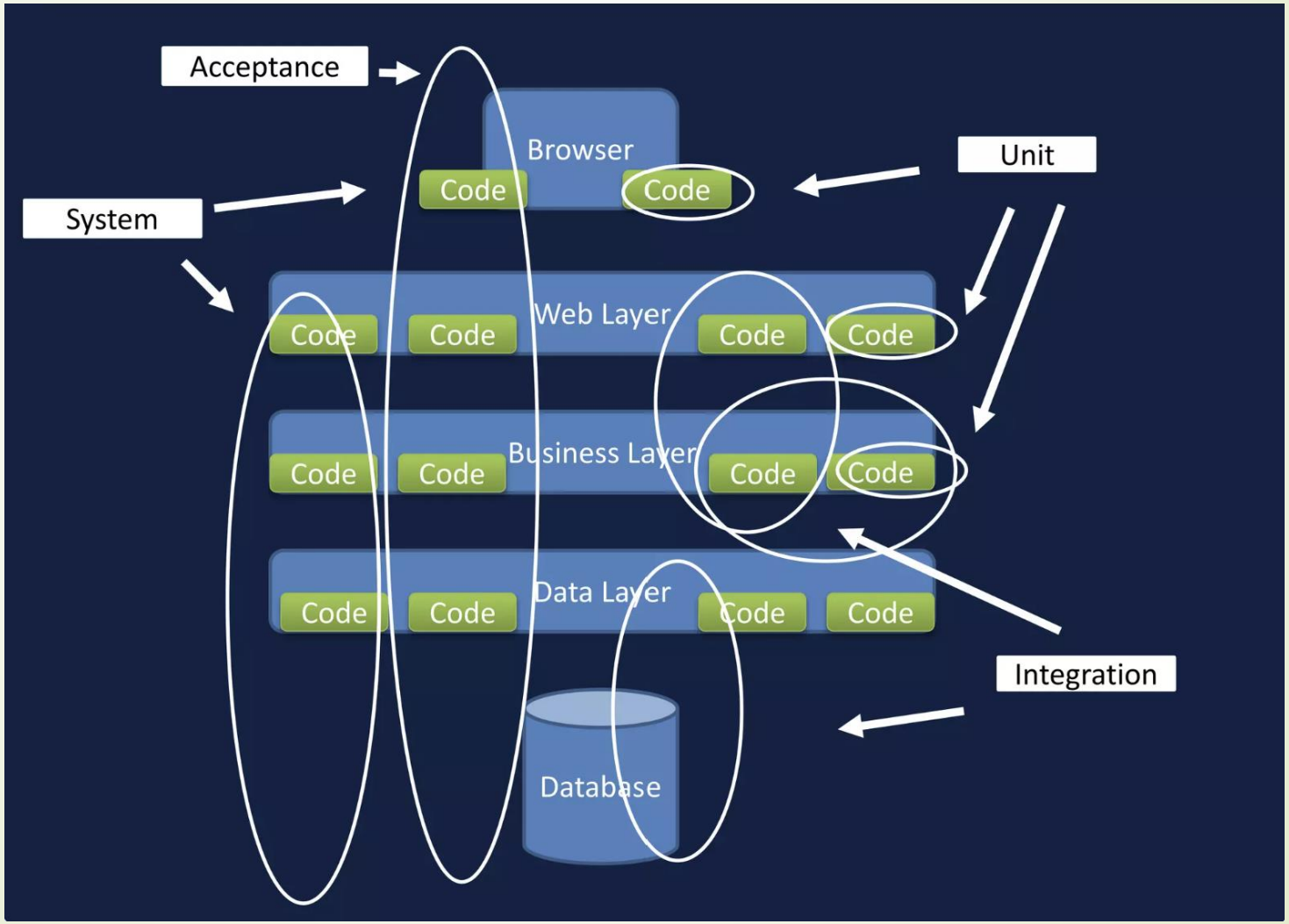
Entendiendo la pirámide del Testing ágil



Pirámide del Testing Tradicional vs. Agile

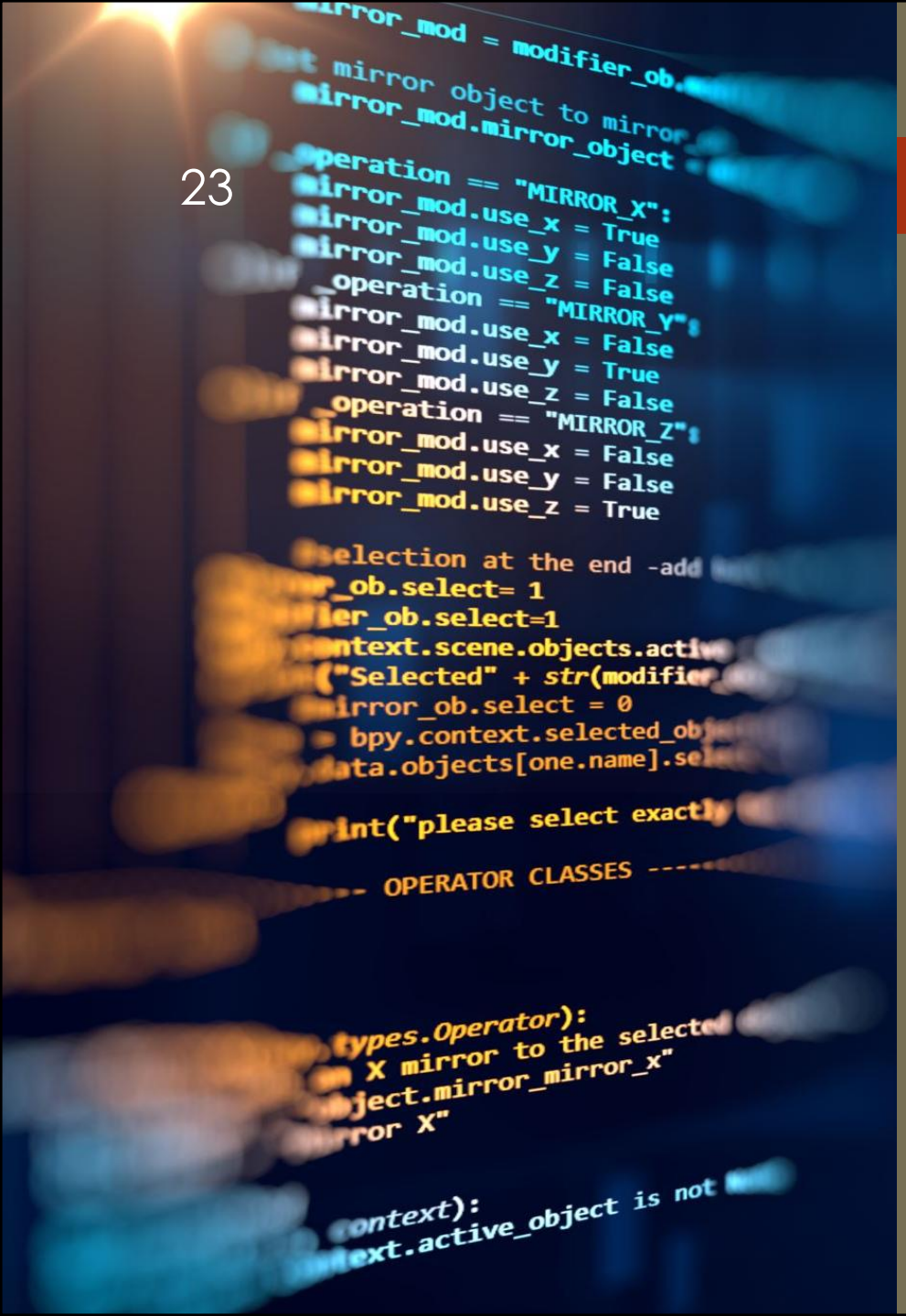


Niveles de Prueba



Beneficios de la automatización del testing

- Rápida ejecución
- Reusabilidad
- Mayor cobertura
- Mayor precisión
- Mayor alcance
- Reducción de costos



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

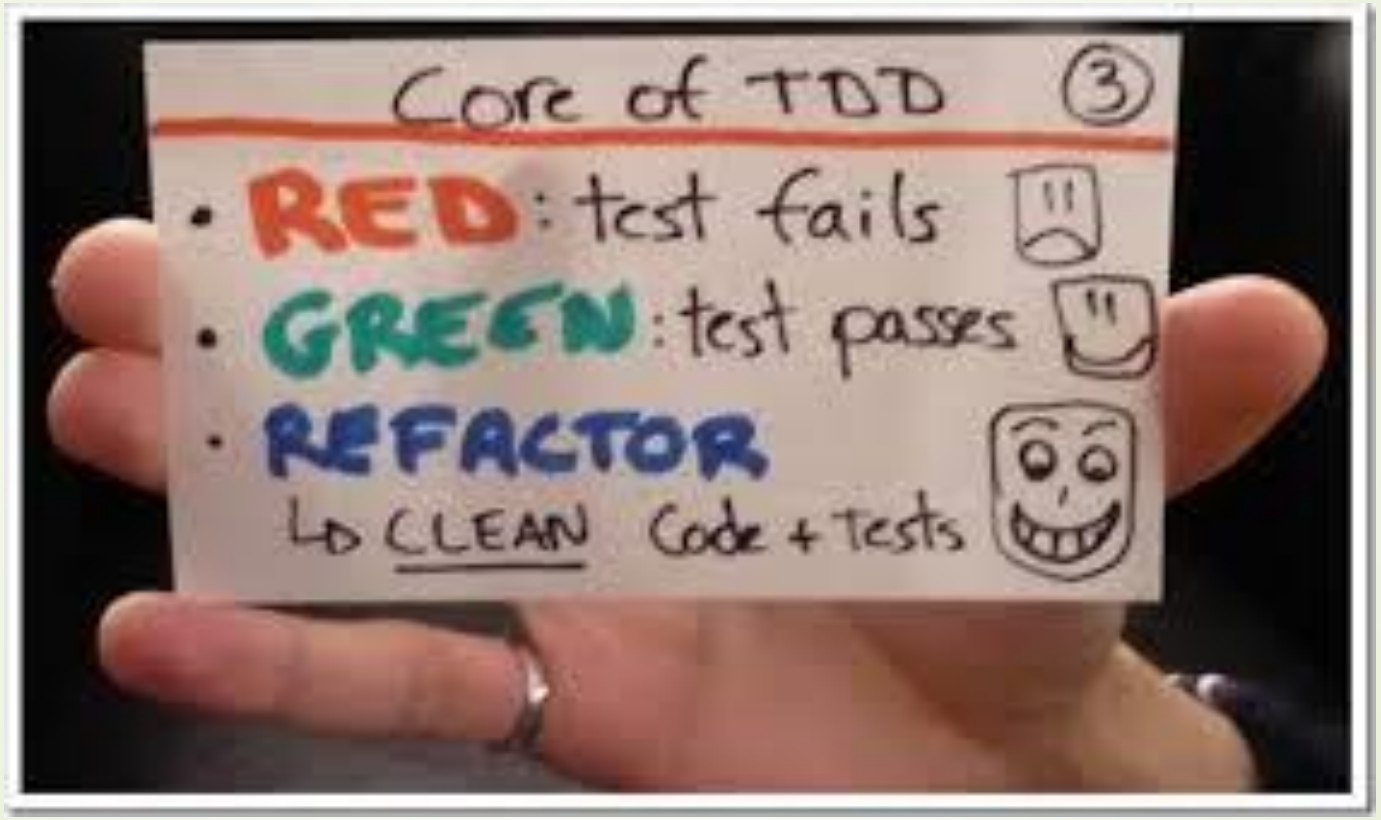
TDD

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

El Corazón de TDD



Las 3 leyes de TDD

No escribir una línea de código hasta que no hayas escrito antes un test case que falla

No es necesario escribir más test de los necesarios para que falle. Normalmente con un test que falle es suficiente.

No vas a escribir más código en Producción que el necesario para que el test pase.

En resumen


¿QUÉ ES TDD?
SE COMBINAN 2 METODOLOGÍAS
1. Test-first development: Escribir las pruebas primero.
2. Refactoring: Refactozación de código.
1 La prueba debe fallar: Se muestran los fallos en rojo.
2 Escribir el código mínimo para que la prueba pase
3 La prueba debe pasar: Las que pasan se muestran en verde.
4 Refactoring: Se debe mejorar el código.
TDD (Test Drive Development) es una metodología de desarrollo, cuyo objetivo es crear primero las pruebas y luego escribir el software.
Mejoran el código en cualquier momento.
La lectura del código será mucho mejor al tener ejemplos de uso (las pruebas).
Los equipos de testing, development y analyst serán más felices y eficientes.