TDD

Test Driven Development

¿Cómo construimos software?

Figura 1: Código primero

Dando vuelta el método

Figura 2: Prueba primero

Agregando una vuelta de tuerca

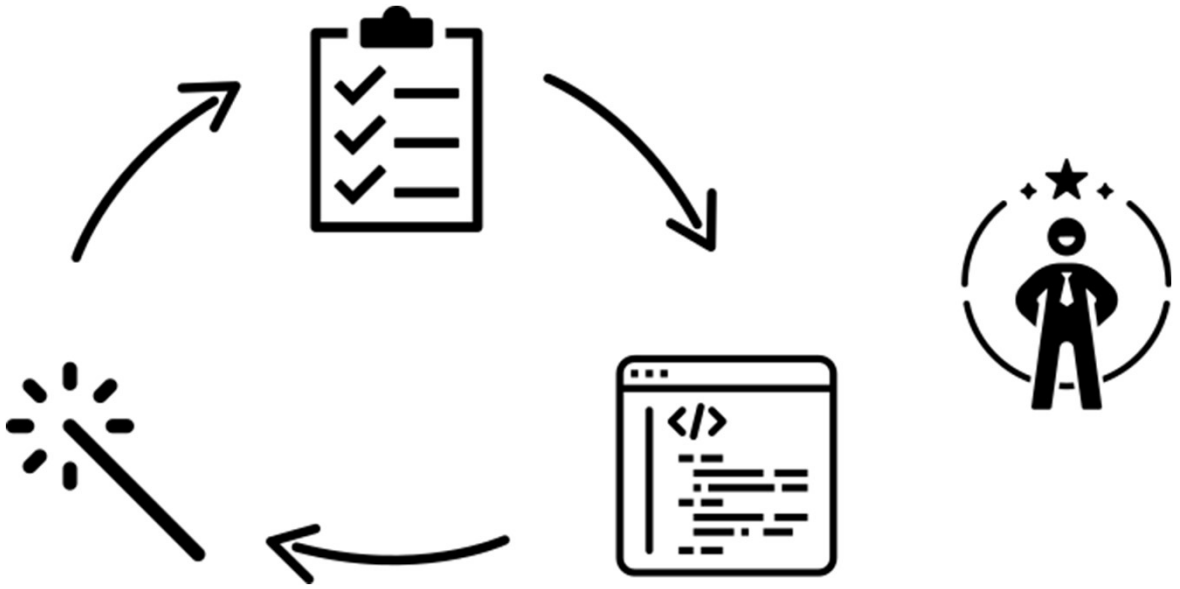


Figura 3: TDD

TDD: ¿Qué es?

Es una técnica de programación, por la cual el código se escribe para satisfacer pruebas que ponen en evidencia la necesidad de dicho código. Es por ello que se llama “desarrollo conducido por las pruebas”.

- Sólo se escribe nuevo código cuando una prueba ha fallado.
- Se elimina la duplicación.

Características

- Es una forma predecible de desarrollar software.
- Permite aprender todas las lecciones que el código tiene para enseñarnos.
- Mejora los resultados al bajar la probabilidad de fallas.
- Permite que el equipo tenga confianza en las partes.

Técnica

1. ROJO: Escribir una prueba **que falle.**
2. VERDE: Hacer que la prueba **pase.**
3. AZUL: **Reducir duplicación.**
4. Vuelta a empezar.

El mantra de un vistazo

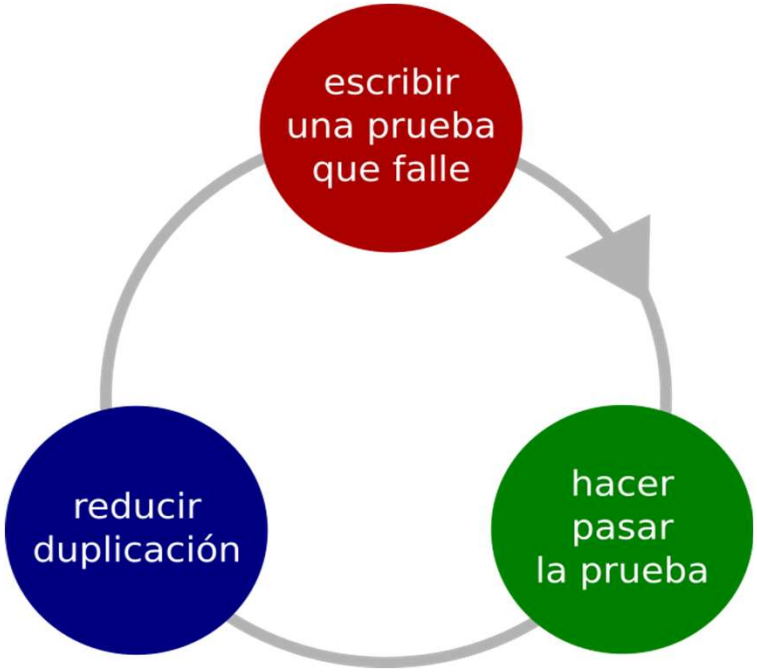
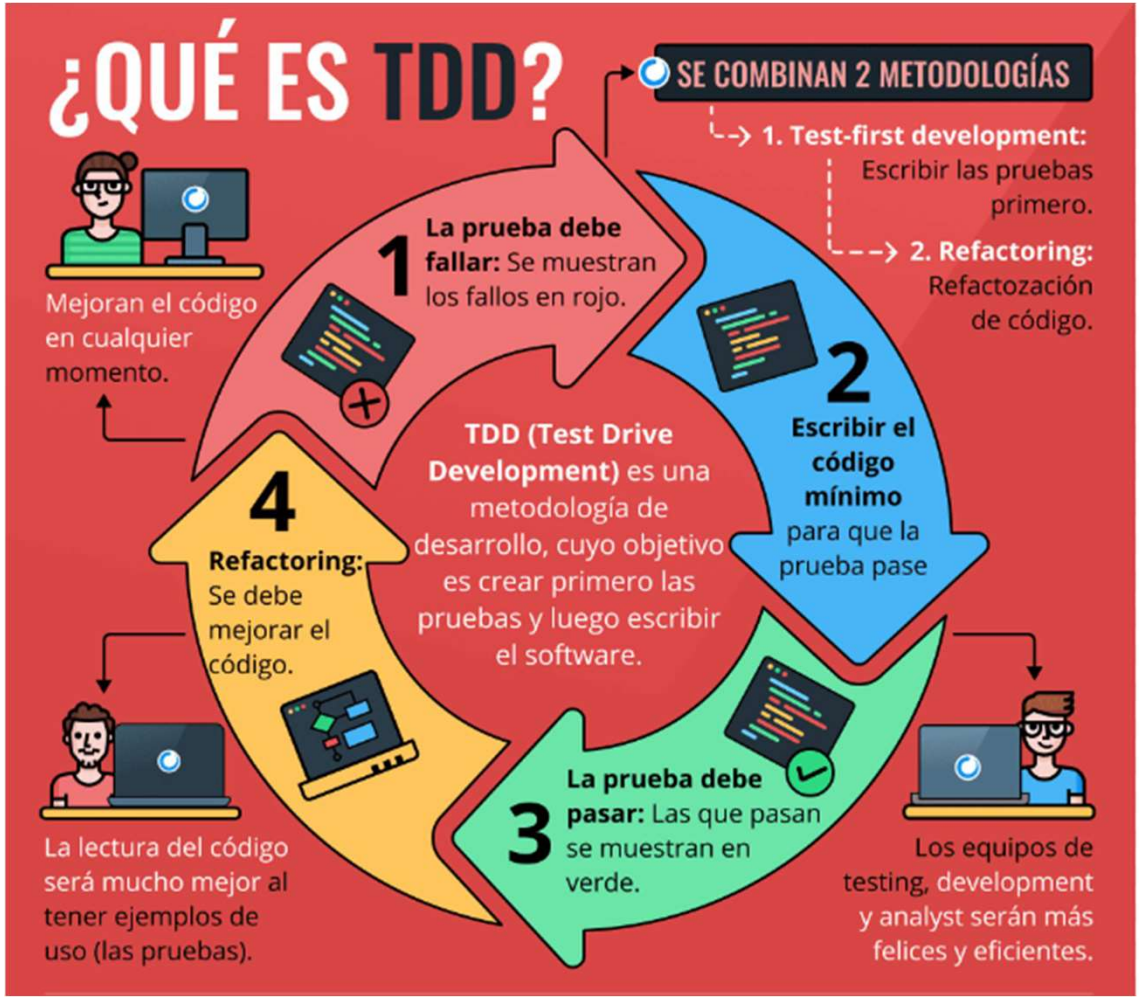
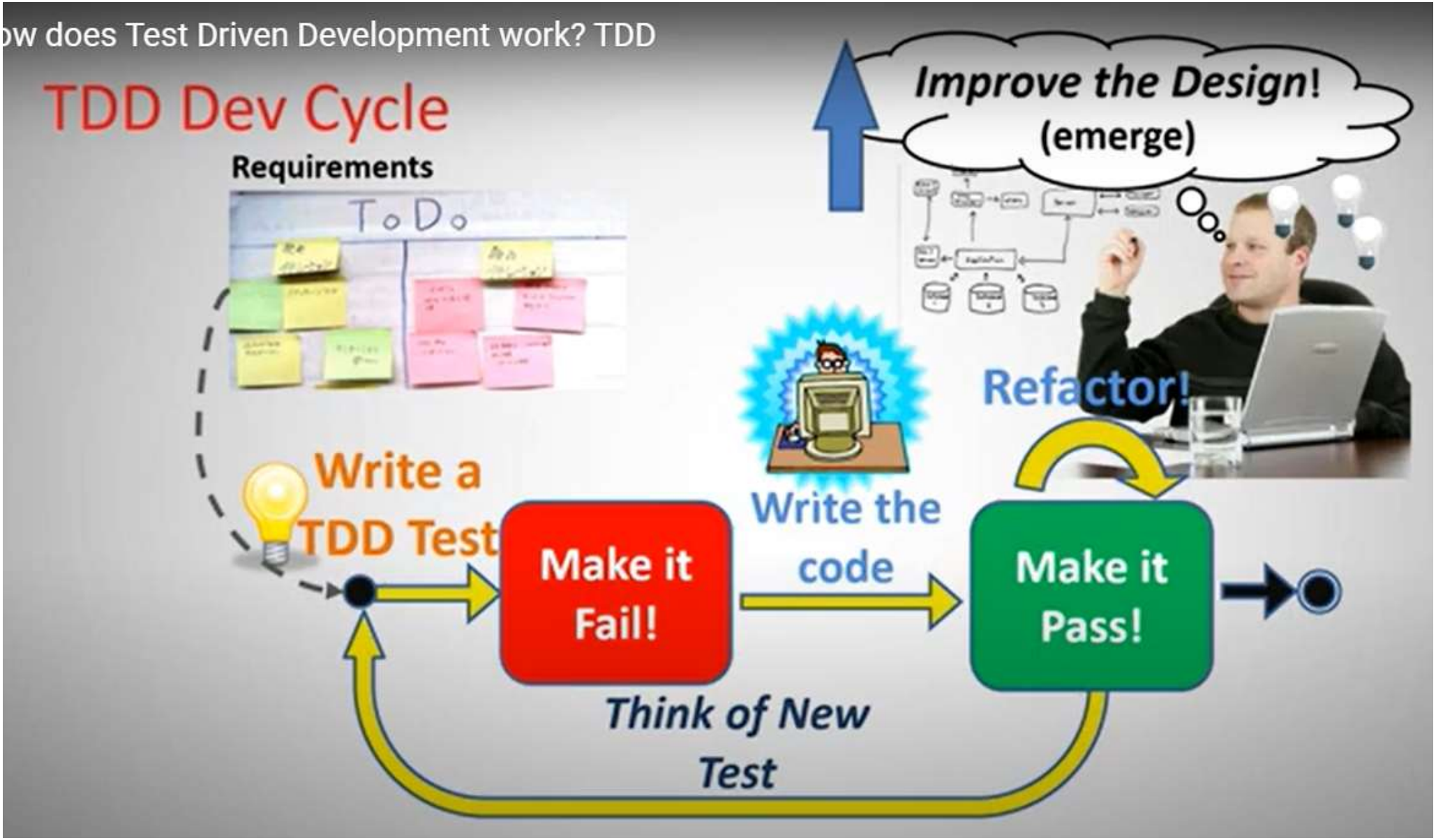


Figura 4: De un vistazo


¿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.


ow does Test Driven Development work? TDD
TDD Dev Cycle
Requirements
To Do
Improve the Design!
(emerge)
Refactor!
Write a
TDD Test
Make it
Fail!
Write the
code
Make it
Pass!
Think of New
Test

Consecuencia

A medida que las pruebas se vuelven más específicas, el código se vuelve más general.

- *Robert Martin*

La idea básica es que a medida que agrega pruebas, las pruebas se vuelven más y más específicas. Esto tiene sentido ya que las pruebas son, después de todo, especificaciones. Cuantas más especificaciones tenga, más específico se vuelve todo el cuerpo de especificaciones.

Recapitulando

- Cómo construimos software
- Invertir el proceso
- Agregar etapas y una metodología
- TDD
- Ejemplos

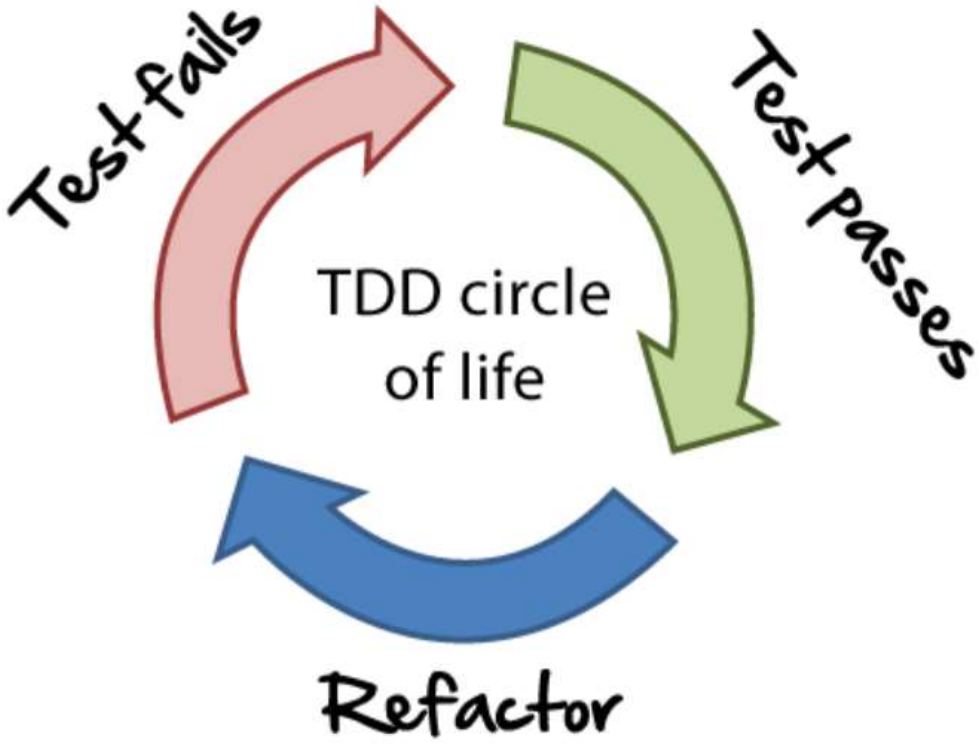


Figura 6: Ciclo de vida cuando se agrega nuevo código

Ejemplo

Figura 5: Cuenta Ganado

Para profundizar

- Beck, K. (2003). Test-driven development: By example. Addison-Wesley.

- Martin, R. The Transformation Priority Premise.

  Último 05 de septiembre, 2023, de https://blog.cleancoder.com/unclebob/2013/05/27/TheTransformationPriorityPremise.html