

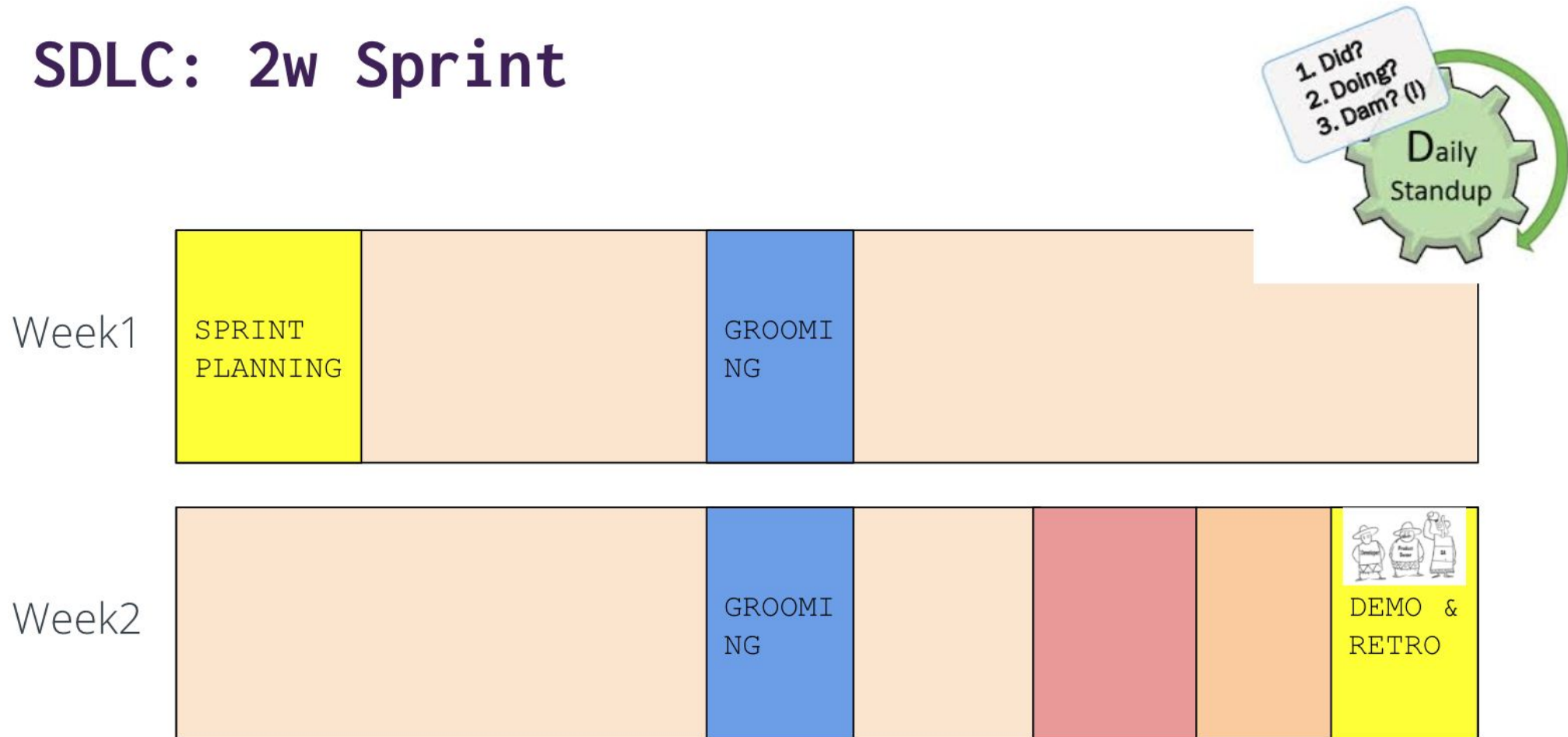


## II. EL PROCESO DE PRUEBAS

*“Quality is never an accident. It is always the result of high intention, sincere effort, intelligent direction and skillfull execution. It represents the wise choice of many alternatives.”*

## II – El proceso de pruebas

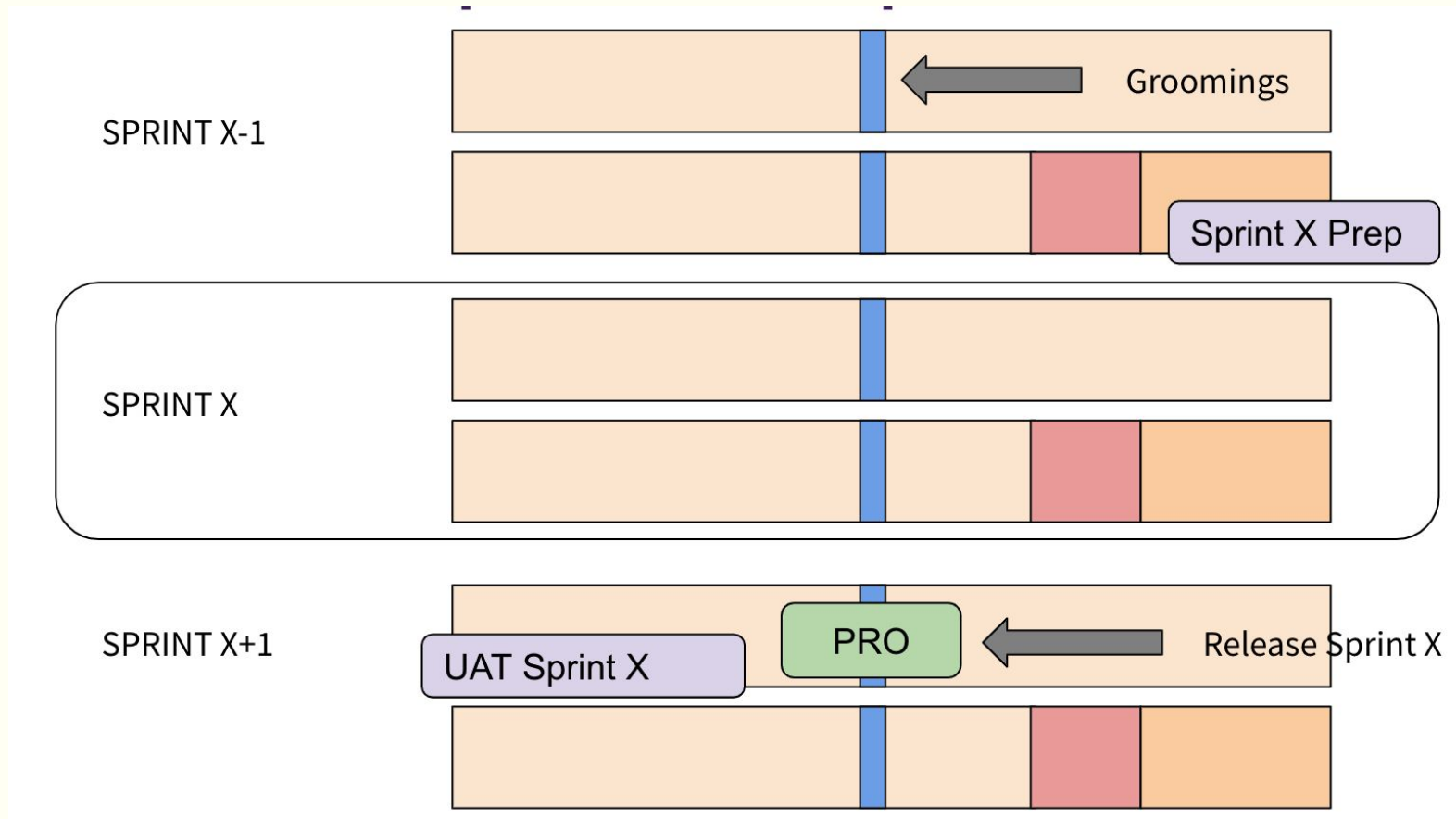
### SDLC: 2w Sprint



## II – El proceso de pruebas

### Actividades de QA

- Previo al Sprint
  - Refinamiento de Requisitos
  - Priorización de Defectos
  - Preparación del Sprint
- Sprint
  - Planificación
  - Inspección de Requisitos
  - *Pair-Testing*
  - Pruebas Exploratorias
  - Diseño de Pruebas
  - Ejecución de Pruebas
  - Verificación
  - Regresión
  - Automatización
- Después del Sprint
  - QA Sign-Off
  - UAT - Validación
  - *Testing* en Producción
  - *Hot-Fixes*



## II – El proceso de pruebas

### Requisitos



## II – El proceso de pruebas

---

### *Test Basis* – Historias de Usuario

# Historia de Usuario

- **Independiente:** Tanto como sea posible, la dependencia entre historias harán la planificación, priorización y estimación mucho más difícil
- **Negociable:** Los detalles de la historia pueden ser cuestionados durante la planificación. Una historia con demasiado detalles puede a veces limitar las conversaciones
- **Valor** al cliente
- **Estimable:** Tiene que tener suficiente detalle para el equipo de desarrollo para poder estimar y priorizar la historia
- **Pequeña:** Cada historia debe representar un pequeño esfuerzo
- **Testeable:** El criterio de aceptación debe ser claro y no ambiguo, requisitos como “tiene que ser fácil de usar” o “no deber ir muy lento ayudan.

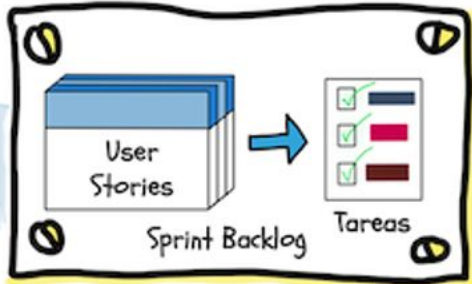


Keep  
It  
Simple

## II – El proceso de pruebas

---

### Historias de Usuario – Definición



Title

User should be able to save story

Traditional  
User Story

As a user  
I want to save a story I'm reading  
Because I found it useful

Scenario

Given that I'm reading a story  
When I tap the *icon* to save a story  
Then save it to my 'Saved Stories'

Additional info

INFO  
*icon* = bookmark icon  
Design - [URL for design / assets]

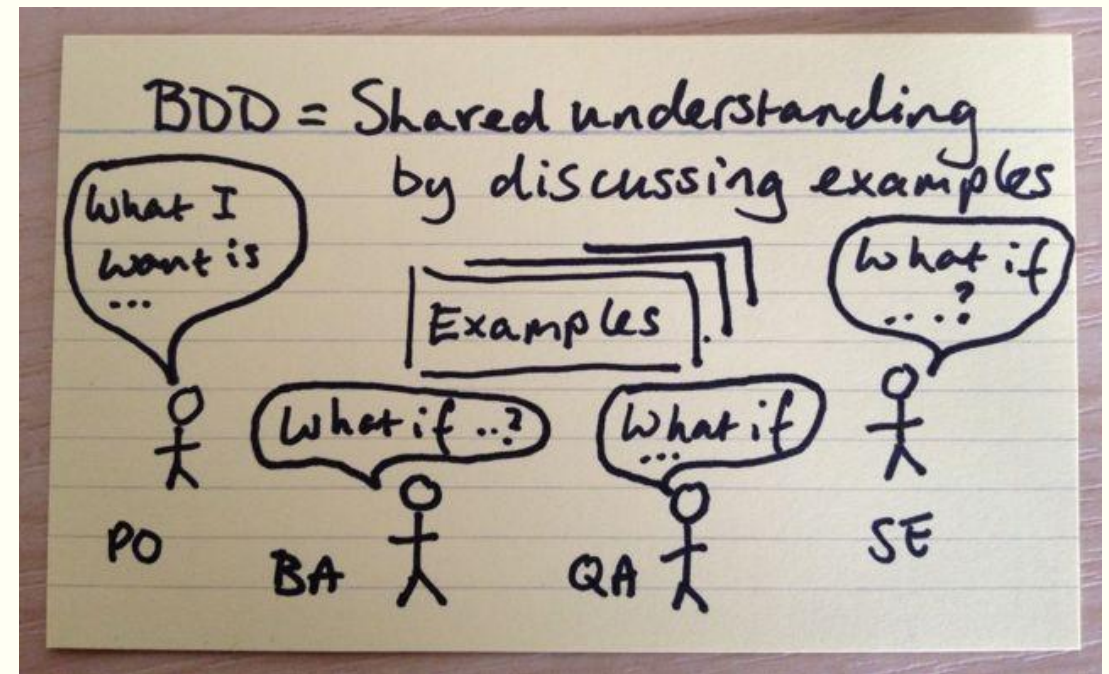


## II – El proceso de pruebas

---

### Historias de Usuario – Escenarios

- Todas las definiciones BDD se escriben en un idioma común.
- El principal objetivo es que el equipo describa los detalles de cómo se debe comportar la aplicación a desarrollar, y de esta forma será comprensible por todos.
- Acceso a una comunicación más clara y con la mínima jerga tecnológica.
- Hace posible que la colaboración entre los equipos técnicos y no técnicos se ejecute con mayor eficiencia.

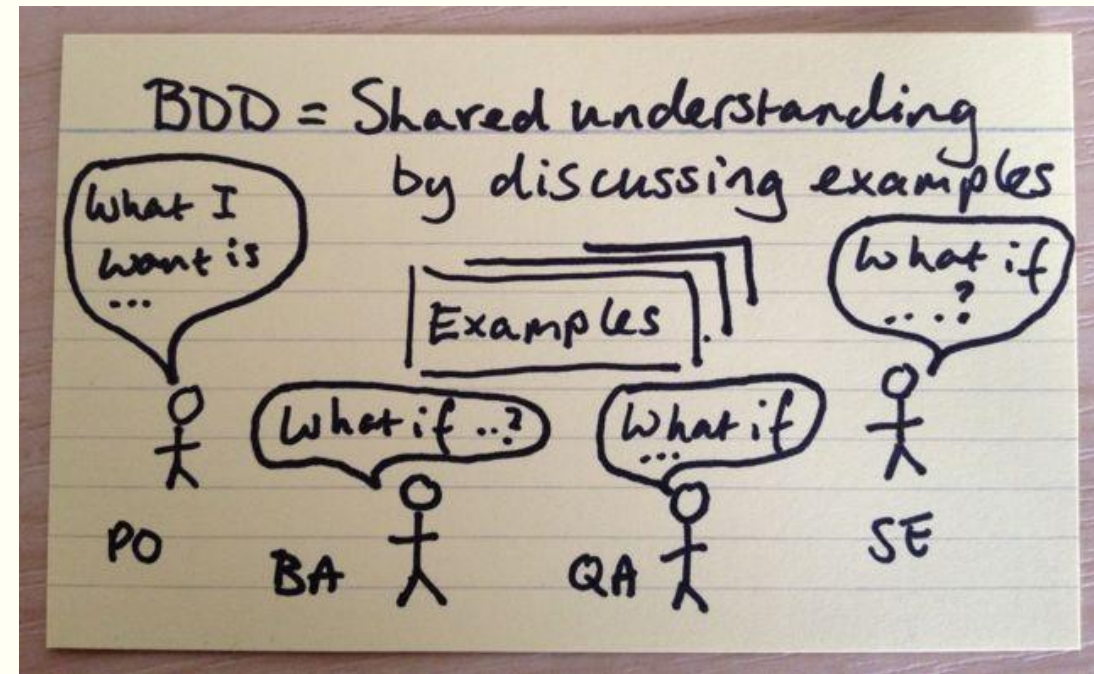


## II – El proceso de pruebas

---

### Historias de Usuario – Escenarios

- Cada requisito debe convertirse en historias de usuario, definiendo ejemplos concretos.
- Cada ejemplo debe ser un escenario de un usuario en el sistema.
- Ser consciente de la necesidad de definir "la especificación del comportamiento de un usuario"





## II – El proceso de pruebas

---

### Historias de Usuario – Escenarios

Este patrón también se utiliza en BDD para ayudar a la creación de historias de usuarios:

**As a ‘Como’:** Se especifica el tipo de usuario.

**I want ‘deseo’:** Las necesidades que tiene.

**So that ‘para que’:** Las características para cumplir el objetivo.

Un ejemplo práctico de historia de usuario sería:

- **Como** cliente interesado, **deseo** ponerme en contacto mediante el formulario, **para que** atiendan mis necesidades.

## II – El proceso de pruebas

---

### Historias de Usuario – Escenarios

Para definir los casos BDD para una historia de usuario se deben definir bajo el patrón '*Given-When-Then*', que se define como:

**Given 'dado':** Se especifica el escenario, las precondiciones.

**When 'cuando':** Las condiciones de las acciones que se van a ejecutar.

**Then 'entonces':** El resultado esperado, las validaciones a realizar.

Un ejemplo práctico sería:

**Given:** Dado que el usuario no ha introducido ningún dato en el formulario.

**When:** Cuando hace clic en el botón Enviar.

**Then:** Se deben mostrar los mensajes de validación apropiados.

## II – El proceso de pruebas

---

### Historias de Usuario – Escenarios

Para definir los casos BDD para una historia de usuario se deben definir bajo el patrón 'Given-When-Then', que se define como:

**Given 'dado':** Se especifica el escenario, las precondiciones.

**When 'cuando':** Las condiciones de las acciones que se van a ejecutar.

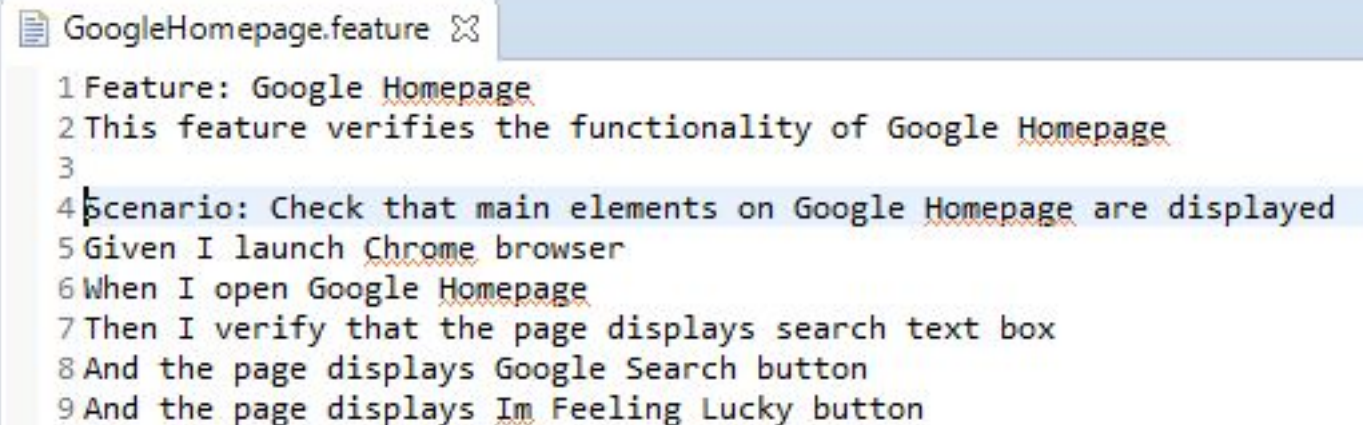
**Then 'entonces':** El resultado esperado, las validaciones a realizar.

Un ejemplo práctico sería:

**Given:** Dado que el usuario no ha in

**When:** Cuando hace clic en el botón

**Then:** Se deben mostrar los mensaj

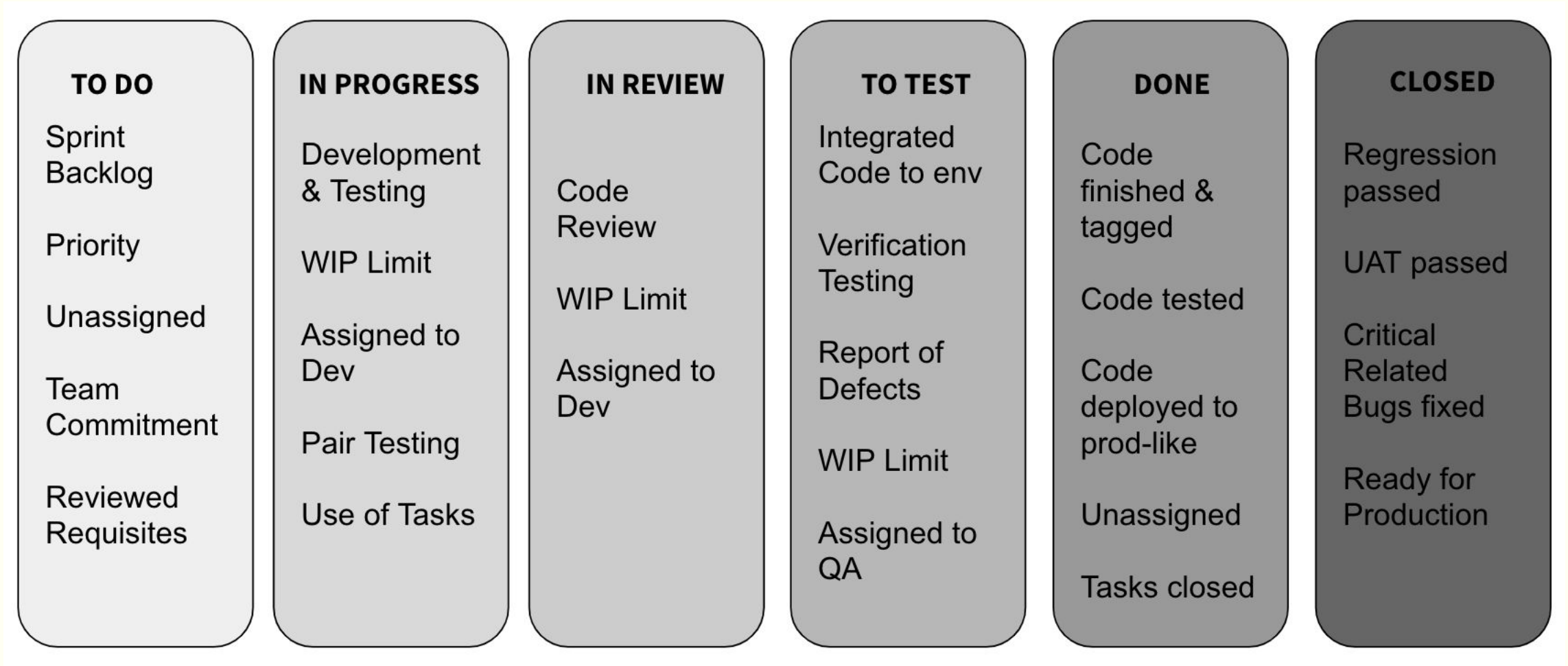


```
GoogleHomepage.feature ✕
1 Feature: Google Homepage
2 This feature verifies the functionality of Google Homepage
3
4 Scenario: Check that main elements on Google Homepage are displayed
5 Given I launch Chrome browser
6 When I open Google Homepage
7 Then I verify that the page displays search text box
8 And the page displays Google Search button
9 And the page displays Im Feeling Lucky button
```

## II – El proceso de pruebas

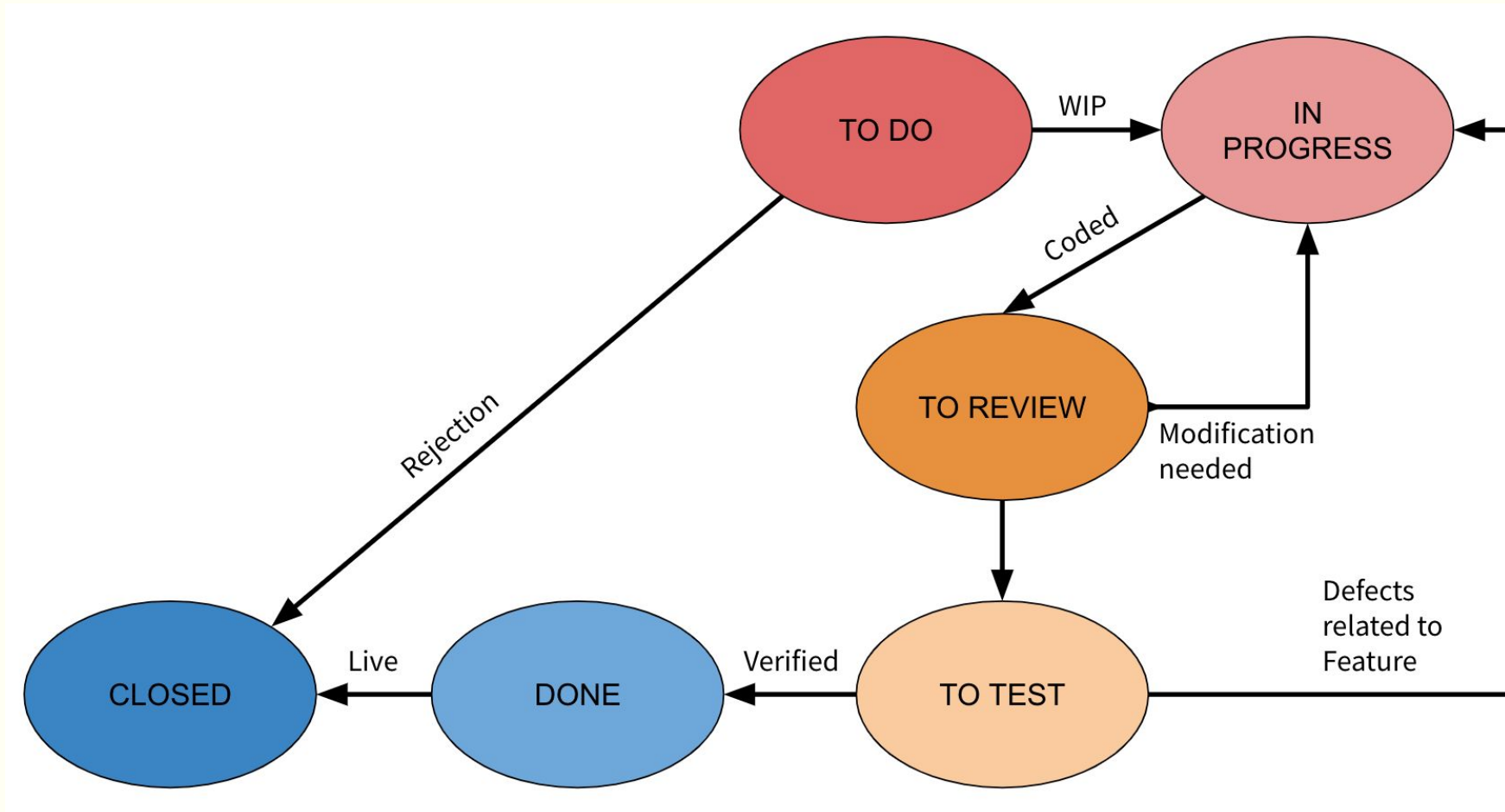
---

### Historias de Usuario - Flujo de Trabajo



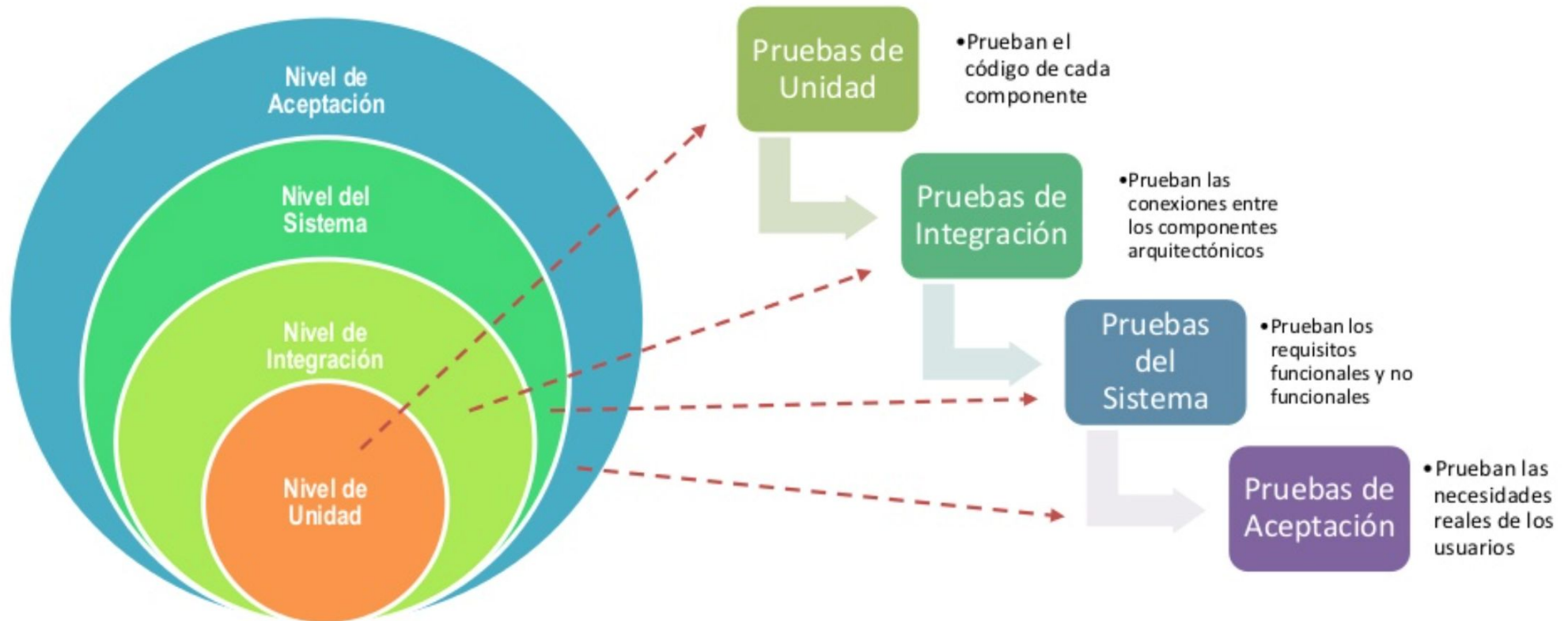
## II – El proceso de pruebas

### Flujo de Trabajo



## 2.1 – Niveles de Pruebas

- Los niveles de pruebas definen los tipos de pruebas que se deben realizar y el orden en que se deben ejecutar





## 2.1 – Niveles de Pruebas

---

### LEVELS OF TESTING

1	Unit Testing	Done by Developers
2	Integration Testing	Done by Testers
3	System Testing	Done by Testers
4	Acceptance Testing	Done by End Users

## 2.2 – Niveles de Pruebas

---

### 2.1.1 Unit/Component Testing



- Probar un sólo método se divide en un conjunto de pruebas aisladas que llamamos *Tests* Unitarios.
- Se trata de *tests* que prueban una lógica de negocio muy acotada dentro de una entidad específica. Estos *tests* pueden llegar a involucrar más de una función, pero nunca más de una clase.
- Si hay más de una clase involucrada en una funcionalidad (lo que es habitual), debemos *mockear* dichas clases.

## 2.2 – Niveles de Pruebas

---

### 2.1.1 Unit/Component Testing

```
probar.py x
11
12 class TestOperaciones(unittest.TestCase):
13     def setUp(self):
14         # Aquí, opcionalmente, ejecuta lo que deberías ejecutar antes
15         # de comenzar cada test.
16         pass
17
18     def test_suma(self):
19         esperado = 3
20         actual = suma(1, 2)
21         # Pásalo en el orden: actual, esperado
22         self.assertEqual(actual, esperado)
23
24     def test Resta(self):
25         esperado = 5
26         actual = resta(10, 5)
27         # Pásalo en el orden: actual, esperado
28         self.assertEqual(actual, esperado)
29
30     def test_multiplicacion(self):
31         esperado = 50
32         actual = multiplicacion(10, 5)
```

## 2.2 – Niveles de Pruebas

---

### 2.1.2 Integration Testing



- Un *test* de integración es aquel que realiza un *test* que involucra a más de una entidad (digamos clases).
- El objetivo no es probar cada una de ellas, sino cómo colaboran entre sí.
- Es importante aislar la prueba del resto de elementos de la plataforma, esto es, no depender de conexiones de red ni llamadas a servicios reales.
- Su objetivo es identificar errores introducidos por la combinación de programas o componentes probados unitariamente, para asegurar que la comunicación, enlaces y los datos compartidos ocurran apropiadamente.

## 2.2 – Niveles de Pruebas

### 2.1.2 Integration Testing

Test  
Description

Test  
Script

Test  
Results

The screenshot displays the Postman application interface. At the top, there's a navigation bar with 'Runner', 'Import', and 'Builder' tabs. Below this, a filter bar shows 'Filter by gender and n'. The main area is divided into sections: 'Filter by gender and nationality' with a description, a request bar with 'GET' method and URL 'https://randomuser.me/api?gender=male&nat=us', and tabs for 'Authorization', 'Headers', 'Body', 'Pre-request Script', and 'Tests'. The 'Tests' tab is active, showing a JavaScript test script. Below the script, the 'Tests (5/5)' section shows the results of the tests, all of which passed.

Filter by gender and n × + ...

No Environment

Examples (0)

Filter by gender and nationality

This request tests the `gender` and `nat` (nationality) filters in combination. The results should contain a single, **male** user, from the **United States**. The tests verify that the user's gender and title are correct, and that the address is in one of the 50 U.S. states.

GET https://randomuser.me/api?gender=male&nat=us Params Send Save

Authorization Headers Body Pre-request Script Tests Cookies Code

```
2 var data = JSON.parse(responseBody);
3 var user = data.results[0];
4
5 tests["A single user was returned"] = data.results.length === 1;
6
7 // Gender tests
8 tests["Gender is male"] = user.gender === "male";
9 tests["Title is Mr."] = user.name.title === "mr";
10
11 // Nationality tests
12 tests["The user is from the United States"] = user.nat === "US";
13 tests["The address is in the United States"] = arrayOfStates.includes(user.location.state);
14
```

Body Cookies (1) Headers (12) Tests (5/5) Status: 200 OK Time: 123 ms Size: 1.24 KB

All Passed Skipped Failed

- PASS A single user was returned
- PASS Gender is male
- PASS Title is Mr.
- PASS The user is from the United States
- PASS The address is in the United States

## 2.2 – Niveles de Pruebas

---

### 2.1.3 System Testing



- Cuando un test incluye todo el flujo (todas las capas) de un sistema, suele denominarse *Test End to End*.
- Una prueba E2E equivaldría a ejecutar la app en un dispositivo (emulador o real), y probar una funcionalidad, desde que el usuario interactúa con la UI hasta que el *feedback* se muestra por pantalla.
- Esta prueba tiene como objetivo verificar que se han integrado adecuadamente todos los elementos del sistema y que realizan las operaciones apropiadas funcionando como un todo.
- Es similar a la prueba de integración pero con un alcance mucho más amplio.



## 2.2 – Niveles de Pruebas

### 2.1.3 System Testing

Project Name	Bank Webiste Functionality								
Module Name	Login Functionality								
Created By	Archana								
Created Date	yyyy-mm-xx								
Executed By	XYZ								
Executed Date	YYYY-MM-DD								
Test Case ID	Test Case Description	Pre Steps	Test Step	Preconditions	Test Data	Expected Result	Actual Result	Status	Comments
Test the Login Functionality in Banking	Verify login functionality with valid username & password		Navigate to login page			Able to see the login page	As expected	Pass	
			Enter valid username	Valid Username	username: choudaryac97@gmail.com	Credential can be entered	As expected	Pass	
			Enter valid password	Valid password	password: XXXXXXXX@1	Credential can be entered	As expected	Pass	
			Click on login button			User logged	User logged successfully	Pass	
Test the Login Functionality in Banking	Verify login functionality with valid username & invalid password		Navigate to login page			Able to see the login page	As expected	Pass	
			Enter valid username	Valid Username	username: choudaryac97@gmail.com	Credential can be entered	As expected	Pass	
			Enter valid password	Invalid password	password: XXXXXXXX@2	Credential can be entered	As expected	Pass	
			Click on login button			User logged	Unsuccessful login	Fail	

## 2.2 – Niveles de Pruebas

---

### 2.1.4 Acceptance Testing

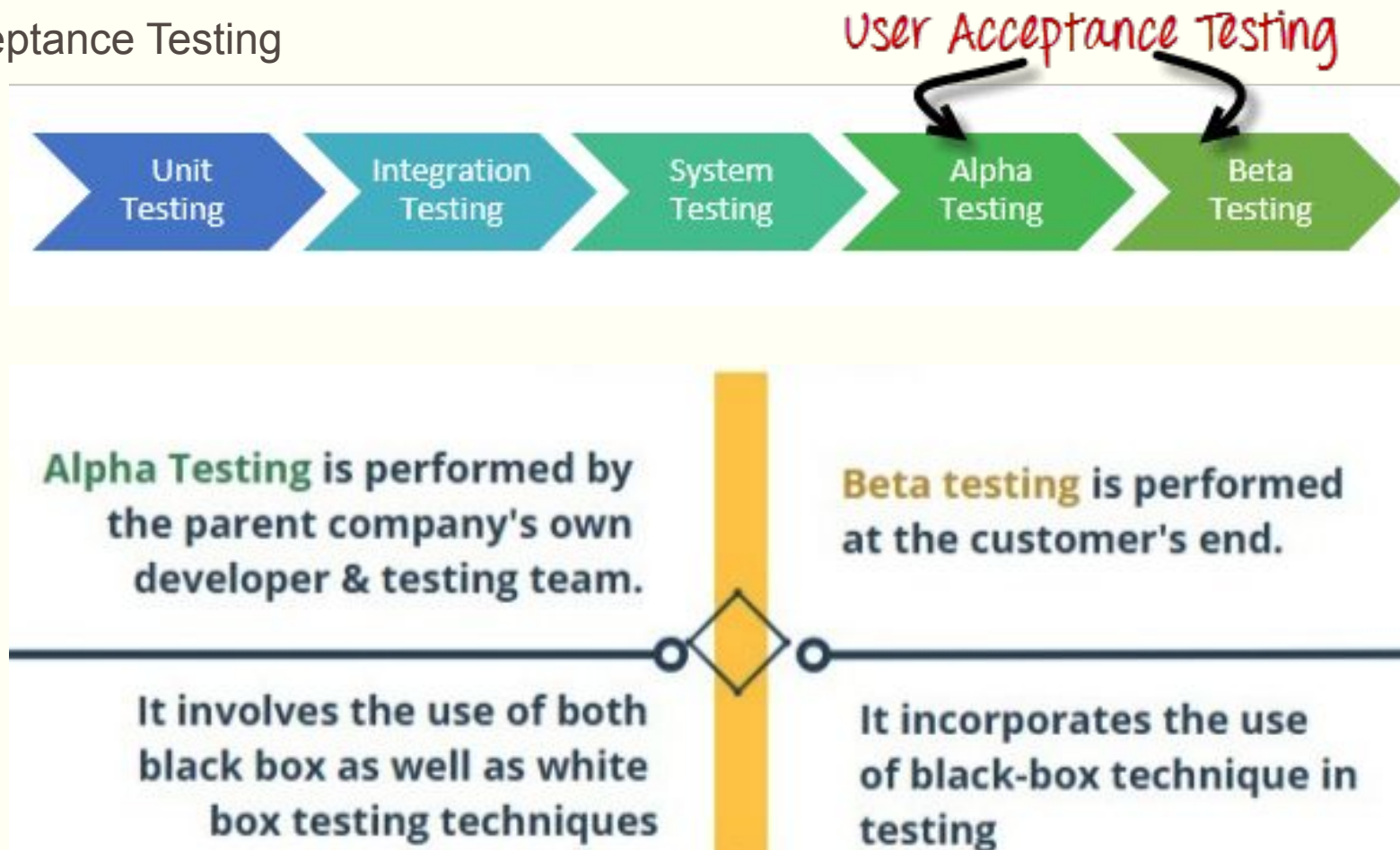


- Son realizadas principalmente por los usuarios con el apoyo del equipo del proyecto.
- El propósito es confirmar que el sistema está terminado, que desarrolla puntualmente las necesidades de la organización y que es aceptado por los usuarios finales.
- La prueba de aceptación puede tener lugar a lo largo de semanas o meses, descubriendo así errores acumulados que pueden ir degradando el sistema.
- Es muy recomendable que las pruebas de aceptación se realicen en el entorno en que se va a explotar el sistema incluyendo el personal que lo va a manejar.
- En caso de un producto de interés general, se realizan pruebas con varios usuarios que reportarán sus valoraciones sobre el producto.

## 2.2 – Niveles de Pruebas

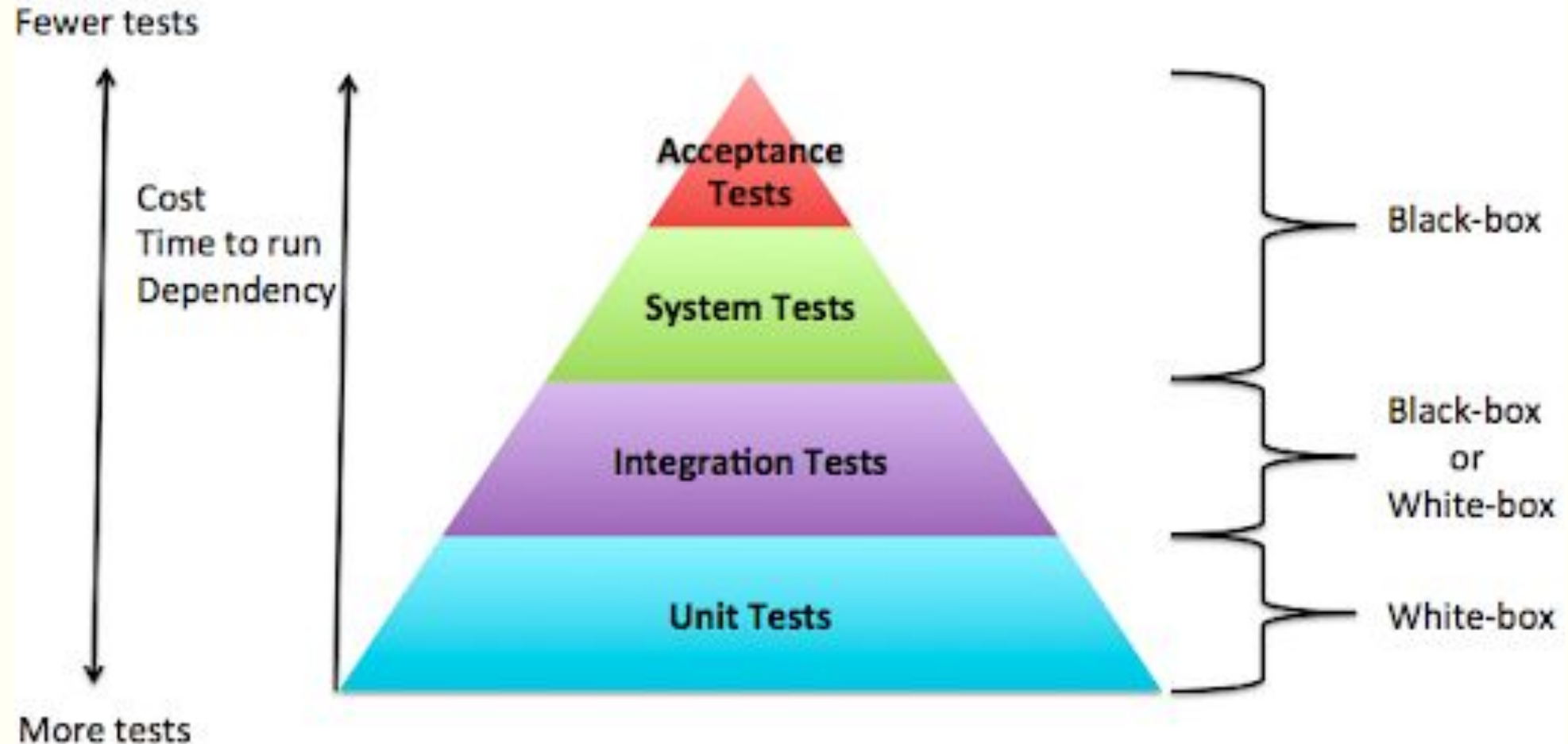
---

### 2.1.4 Acceptance Testing



## 2.1 – Niveles de Pruebas

---



## 2.2 – Tipos de Pruebas

---

### 2.2.3 Manual VS. Automática

**El *testing manual*:** es el método de testeo en el que la mano humana es de vital importancia. El QA crea y ejecuta los casos de prueba adoptando el rol del destinatario final e identificando diversas casuísticas.

- **Guía:** permite el conocimiento y el entendimiento de para qué sirve el software. Qué problemas es capaz de solucionar y cuáles no y cuál será la respuesta en términos de usabilidad.
- **Comprensión:** permite un acercamiento a las sensaciones de los futuros usuarios del software.
- **Casualidad:** un ser humano probando un software siempre se encontrará situaciones inesperadas y sorprendentes

## 2.2 – Tipos de Pruebas

---

### 2.2.3 Manual VS. Automática

**El *testing* automático:** es el método de ejecutar una prueba sin el protagonismo de la intervención humana. Los casos de prueba se programan usando una herramienta específica de automatización.

- ***Feedback* muy rápido:** en cuestión de minutos seremos capaces de comprobar si una gran cantidad de pruebas pasan la validación o no.
- **Regresión:** una vez que se ha programado un test este puede volver a usarse tantas veces como se quiera con unos costes prácticamente insignificantes.
- **Evitar errores manuales:** En aquellas pruebas repetitivas y pesadas para el humano, por ejemplo, comparación de datos, chequeos repetitivos, etc.



## 2.3 – Diseño de Pruebas

---

### Caso de Prueba

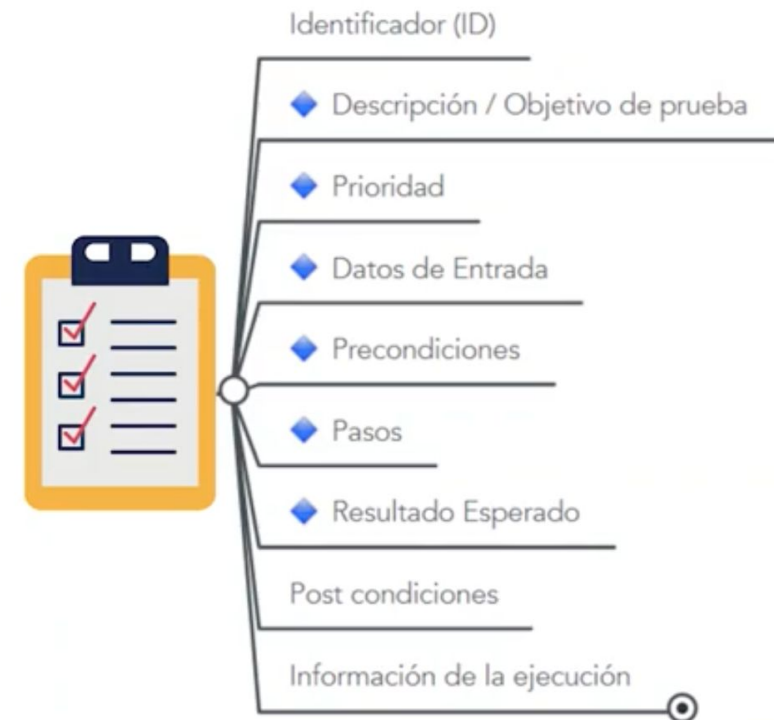
- Un **caso de prueba** o **test case** es, un conjunto de condiciones o variables bajo las cuales un analista determinará si un sistema software, o una característica de éstos es parcial o completamente satisfactoria.
- Con el propósito de comprobar que todos los requisitos de una aplicación son revisados, debe haber al menos un caso de prueba para cada requisito.
- Si la aplicación es creada sin requisitos formales, entonces los casos de prueba se escriben basados en la operación normal de programas de una clase similar.
- Lo que caracteriza un escrito formal de caso de prueba es que hay una *entrada conocida* y una *salida esperada*, los cuales son formulados antes de que se ejecute la prueba.

## 2.3 – Diseño de Pruebas

### Caso de Prueba

- Entendimiento del comportamiento de la aplicación.
- Facilita la automatización.
- Facilita la transferencia de conocimiento.
- Se puede medir la cobertura de pruebas.
- Se puede hacer análisis de riesgo.
- Se pueden obtener métricas.
- Da visibilidad del estado del *testing*.
- Ayuda a trazar fallos y requisitos.

### **PARTES DE UN CASO DE PRUEBA**



## 2.3 – Diseño de Pruebas

---

### Caso de Prueba

Estándares:

- El objetivo de un caso de prueba tiene que ser claro e independiente de otros casos de pruebas.
- Los pre-requisitos y los datos necesarios tienen que estar identificados.
- Los pasos se especifican de forma clara con un resultado esperado para cada paso.
- El caso de pruebas es agnóstico.
- El resultado esperado tiene que ser fácil de comparar con el resultado actual.
- Los pasos tienen que ser fáciles de entender, de reproducir y de automatizar.

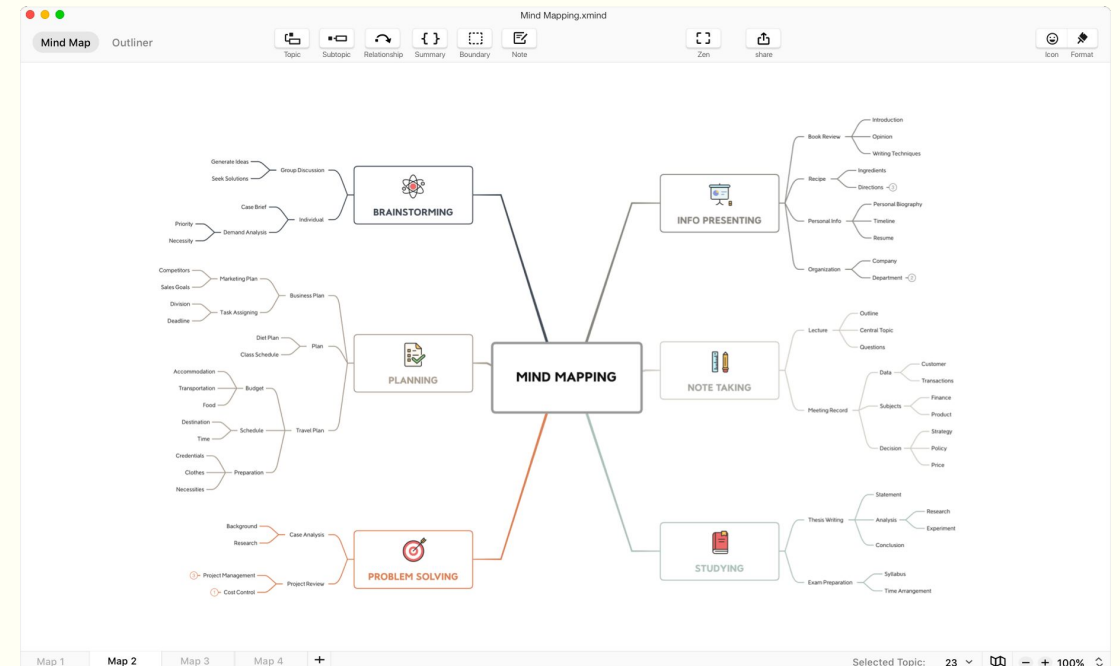
Errores comunes:

- Casos de prueba demasiado largos.
- Los pre-requisitos y los datos de entrada incompletos.
- Uso de nomenclatura que ya no existe.
- Resultados esperados ambiguos.
- Criterio ambiguo para pasar o fallar el test.

## 2.3 – Diseño de Pruebas

### Herramientas: X-Mind

- Herramienta para creación de mapas de ideas.
- Herramienta para creación de casos de prueba a alto nivel.
- Herramienta para pruebas exploratorias.
- <https://www.xmind.net/>



## 2.3 – Diseño de Pruebas

---

### Caso de Prueba - Partes

**Objetivo**: Es la parte más importante del caso de prueba, nos ayuda a entender qué es lo que hay que probar específicamente. Se deriva directamente de las especificaciones y tiene que ser claro y conciso. Idealmente, la prueba podría realizarse sólo leyendo el título.

**Pre-Requisitos**: Especifica el estado del sistema antes de la ejecución de la prueba. También en esta parte incluimos los datos requeridos para la ejecución de la prueba.

**Descripción de la prueba**: Aquí encontramos los pasos para ejecutar la pruebas y probar la funcionalidad o especificación paso por paso. Los pasos tienen que ser detallados para delimitar los fallos encontrados.

**Pasos**: Los pasos tienen que ser objetivos y representar acciones aisladas.

**Resultado Esperado**: Representa la salida de las acciones descritas anteriormente. El resultado final se deriva directamente de la especificación, cualquier variación será un **defecto**.

## 2.3 – Diseño de Pruebas

### Caso de Prueba Tradicional

CI

Edit Test Case

Title \*

Format table with built-in style

Section \*

Prerequisites

Template \*

Test Case (Text)

Type \*

Other

Priority \*

Medium

Estimate

5m

References

Automation Type

None

Preconditions

In vulputate libero at http://www.example.com/.  
  
\* Donec placerat, dui vitae  
\* Ut auctor mi erat ac dolor.  
\* Lorem ipsum dolor sit amet, consectetur adipiscing elit.

The preconditions of this test case. Reference other test cases with [C#](e.g. [C17]).

Steps

In vulputate libero \*\*in nulla feugiat tincidunt\*\*:  
  
\* Felis libero varius orci, in vulputate  
\* Massa turpis scelerisque diam.  
\* Nunc et felis est. Phasellus laoreet nibh vel augue  
\* Faucibus at varius est pretium.

The required steps to execute the test case.

Expected Result

Countrylist.csv

Other File, 2B

Drop files here to attach, or click to browse.

Actions

Delete a test case to remove it from its test suite. This also deletes all related running tests.

Delete this test case



## 2.3 – Diseño de Pruebas

### *Scenario* (VS. Caso de Prueba)

Test Case		S
Título	<i>Scenario</i>	
Pre-condiciones	<i>Given</i>	
Pasos	<i>When</i>	
Resultado Esperado	<i>Then</i>	

Test Case 1: User can log into application

Preconditions: An user already registered

Steps:1. Go to application  
2. Click login button  
3. Enter valid username  
4. Enter valid password  
5. Click to login

Expected Result:  
User is logged in. Main page is displayed.

Scenario 1: User can log into application

Given An user already registered

When user goes to application  
And clicks login button  
And enters valid username  
And enters valid pass  
And clicks to login

Then user is logged in  
And Main page is displayed

## 2.4 – Ejecución de las pruebas

---

### Resultados de Ejecución

#### Pass

- Resultado actual coincide con el resultado esperado
- Buen indicador de la calidad del sistema

#### Fail

- Resultado actual NO coincide con el resultado esperado
- Se debe reportar una incidencia y generar un informe (defecto)

#### Blocked

- La prueba no se puede ejecutar por que la precondition no se cumple

#### Not Applicable

- La prueba no está relacionada con lo que se está probando

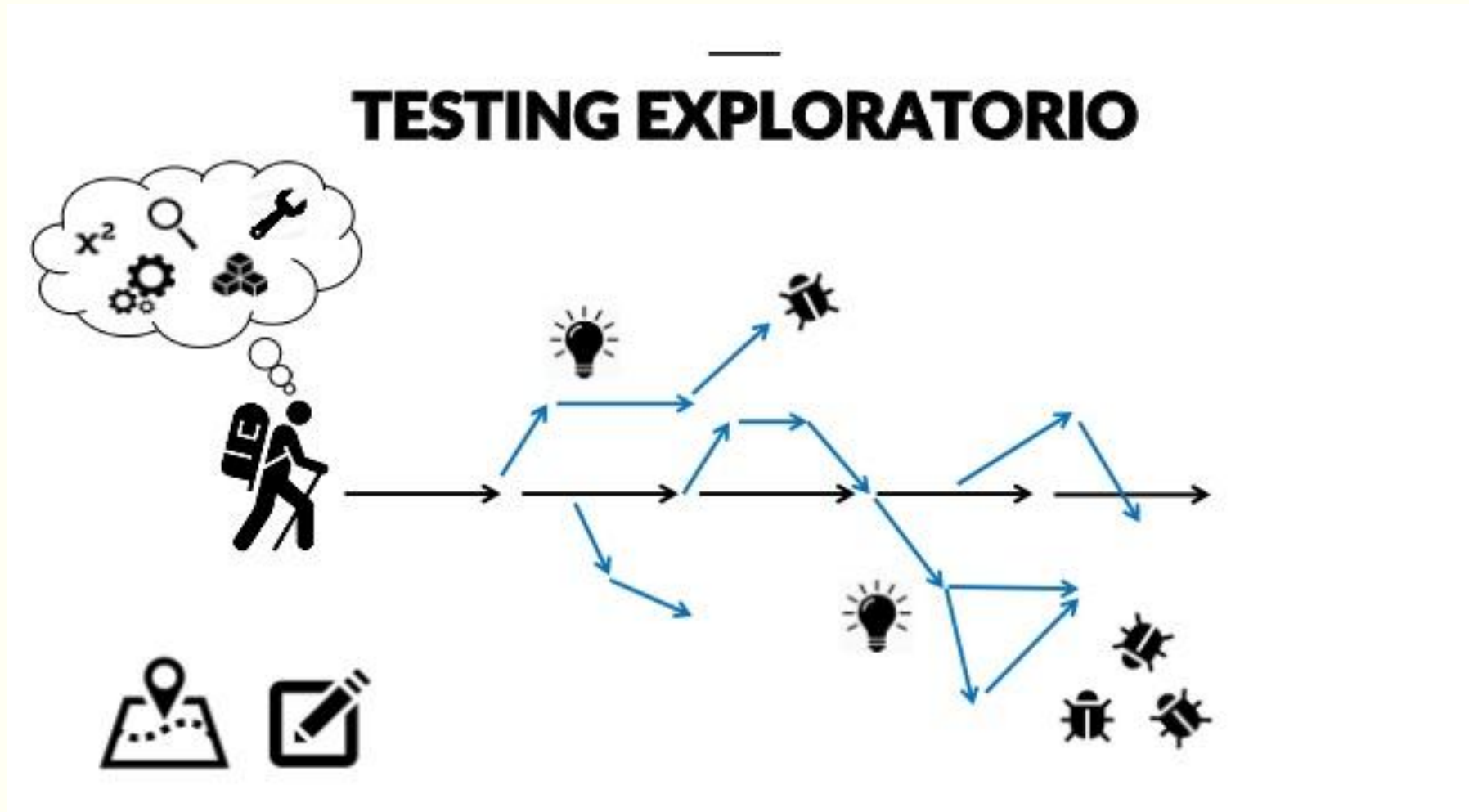
#### Not Run

- NO se ha ejecutado la prueba

## 2.4 – Ejecución de las pruebas

---

### 2.4.1 Pruebas Exploratorias



## 2.4 – Ejecución de las pruebas

---

### 2.4.1 Pruebas Exploratorias

#### ¿Qué es?

- Obtener información rápidamente
- Detectar defectos en lugares que no esperabas encontrarlos
- Tener foco en ciclos funcionales
- Aprender del producto

#### ¿Qué NO es?

- No es probar en forma ad-hoc
- No es probar sin tiempo definido
- No es NO documentar
- No es encontrar defectos que no podemos reproducir

## 2.4 – Ejecución de las pruebas

---

### 2.4.1 Pruebas Exploratorias



# Fuentes

---

- <https://www.itdo.com/blog/que-es-bdd-behavior-driven-development/>
- <https://www.paradigmadigital.com/dev/testing-android-tests-rapidos-y-faciles/>
- <https://www.softwaretestingmaterial.com/levels-of-testing/>
- [https://www.ecured.cu/Niveles\\_de\\_prueba\\_de\\_software](https://www.ecured.cu/Niveles_de_prueba_de_software)
- <https://testerhouse.com/teoria-testing/pruebas-funcionales/>
- [https://www.pragma.com.co/blog/conoce-que-son-las-pruebas-no-funcionales-d e-software](https://www.pragma.com.co/blog/conoce-que-son-las-pruebas-no-funcionales-d-e-software)
- <https://www.sipsa.net/testing-manual-vs-automatico/>
- <https://www.testbytes.net/blog/smoke-testing-explanation-example/>