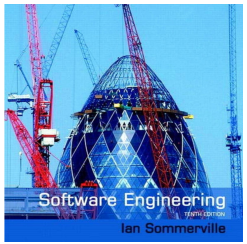
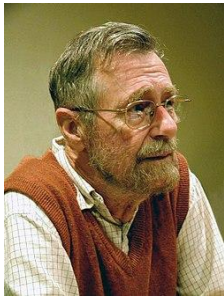
Introducción a las Pruebas de Software



“Testing can **only show** the **presence** of **errors**, not their **absence**”

Dijkstra(1972)



Agenda

8.1 Development testing

8.2 Test-driven development

8.3 Release testing

8.4 User testing(Canary, Dark launch, BetaTest)

¿Qué son las pruebas de SW?

Myers’79

Las pruebas de SW es el **proceso** de ejecutar el programa con la intención de encontrar **errores**”

IEEE ‘90

…es el **proceso** de **operar** un sistema o sus componente bajo **condiciones especificadas**, observando los **resultados**, y realizando una **evaluación** de algunos aspectos del sistema o componente.

…es el **proceso** de analizar un ítem de sw para **detectar** las **diferencias** entre lo que existe y las condiciones requeridas (i.e. bugs) y evaluar la funcionalidad del ítem de sw.

Galin ‘04

…es un **proceso formal** llevado a cabo por un **equipo especializado** de pruebas en el cual se **examina** una unidad de sw,varias unidades integradas o un paquete completo de software ejecutando los programas en un computador.

Validation & Verification

Verification

Software verification is the **process** of checking that the software **meets** its stated **functional** and **non-functional requirements**

Validation

…is a more general process. The aim of software validation is to ensure that the software meets the **customer's expectations**

Validation Are we building the **right** product? (Correcto)

Verification Are we building the product **right**? (Correctamente)

States of Testing

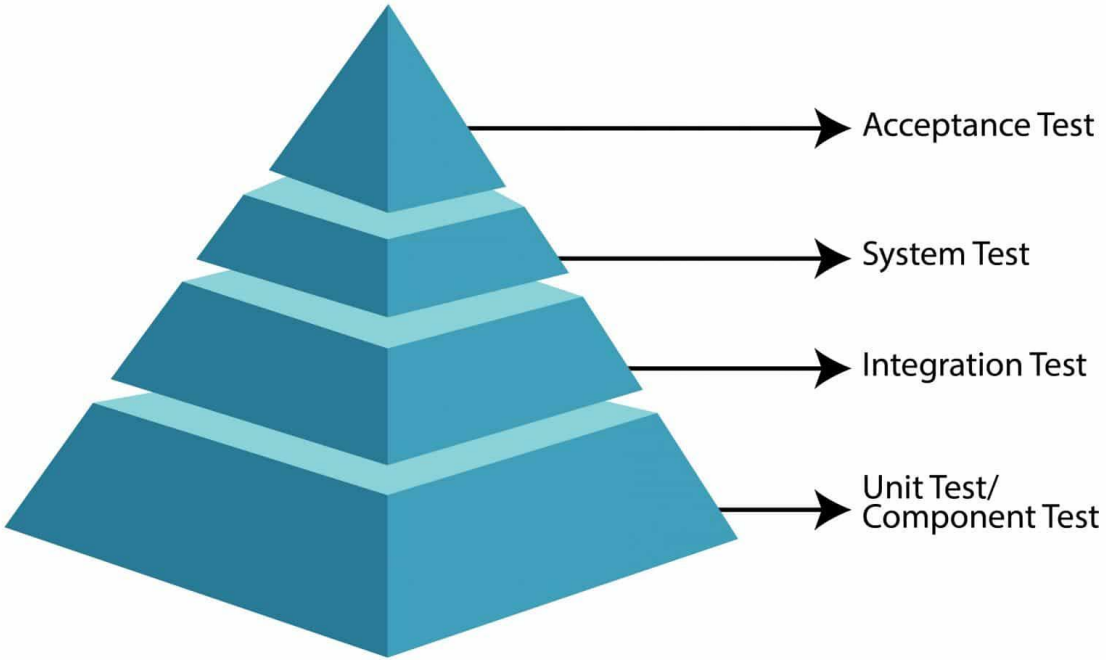
Development testing

Release testing

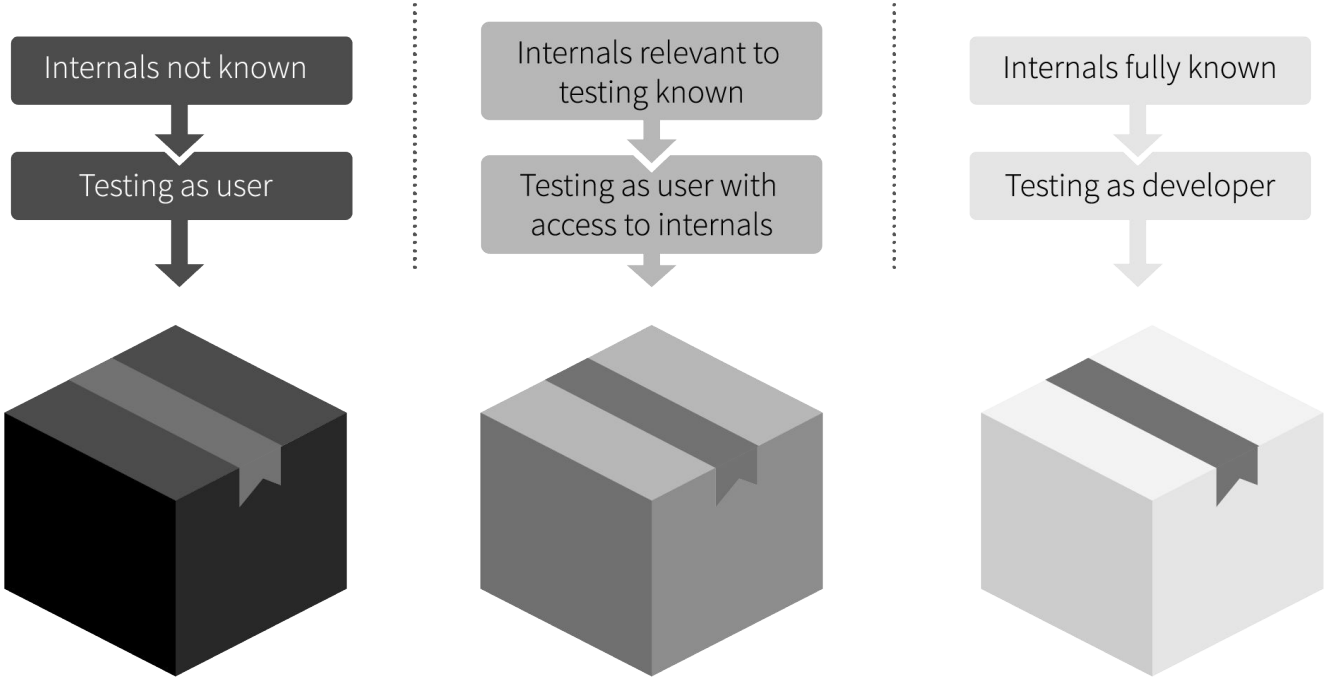
User testing

8.1 Development testing

States


Acceptance Test
System Test
Integration Test
Unit Test/
Component Test

Testing Approaches



Unit Test (Asserts)

```java
/** Determines if three doubles can be sides of triangle.
*/
public class Triangle
{
   /** side 1. */
  private double a;
   /** side 2. */
  private double b;
   /** side 3. */
  private double c;

/** Constructor for takes in  a triangle based on lengths of three si
      @param aIn side 1
      @param bIn side 2
      @param cIn side 3
*/
  public Triangle(double aIn, double bIn, double cIn)
  {
    a = aIn;
    b = bIn;
    c = cIn;
    if (a <= 0 || b <= 0 || c <= 0) {
      throw new IllegalArgumentException("Sides: " + a + " " + b
          + " " + c
          + " -- each must be greater than zero.");
    }
    if ((a >= b + c) || (b >= a + c) || (c >= a + b)) {
      throw new RuntimeException("Sides: " + a + " " + b + " " + c
          + " -- not a triangle.");
    }
  }

   /** Classifies a triangle based on lengths of three sides.
      @return classification of triangle
   */
  public String classify() {
    String result;
    if ((a == b) && (b == c)) {
      result = "equilateral";
    }
    else if ((a != b) && (a != c) && (b != c)) {
      result = "scalene";
    }
    else {
      result = "isosceles";
    }

    return result;
  }
}
```

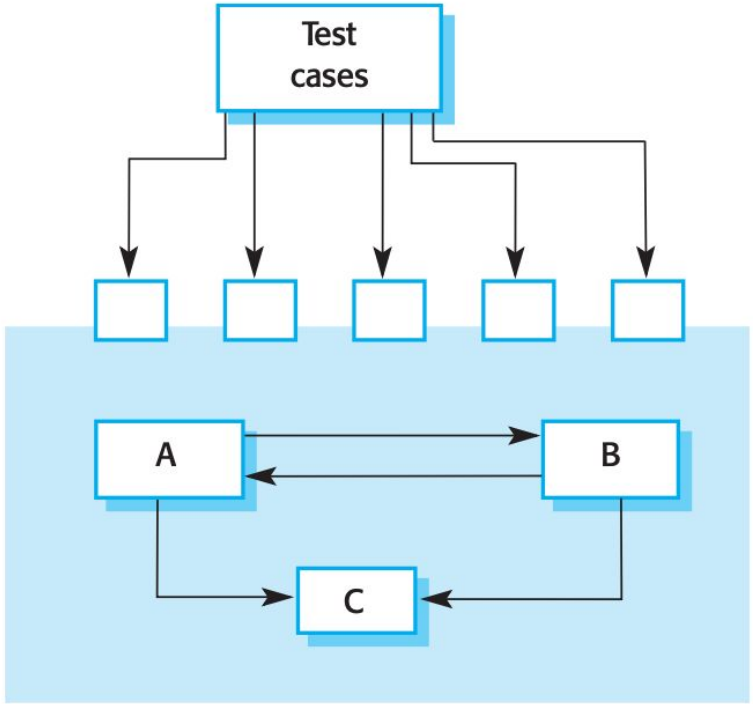
```java
/** Test 3 - for bad arguments. **/
@Test
public void argumentTest3() {
  boolean thrown = false;

  try {
    Triangle t = new Triangle(2, 5, -10);
  }
    catch (IllegalArgumentException e) {
      thrown = true;
    }

  Assert.assertTrue(thrown);
}

/** Test 4 - for "Not a triangle". **/
@Test (expected = RuntimeException.class)
public void notTriangleTest() {
  Triangle t = new Triangle(2, 5, 10);
}
```

```java
/** Test 5 - for "Equilateral" triangle. **/
@Test
public void equilateralTest1() {
  Triangle t = new Triangle(12, 12, 12);
  Assert.assertEquals("\nSides: " + 12 + " " + 12 + " " + 12,
            "equilateral",
            t.classify());
}

/** Test 6 - for "Isosceles" triangle. **/
@Test public void isoscelesTest1() {
  Triangle t = new Triangle(12, 12, 13);
  String result = t.classify();
  Assert.assertEquals("\nSides: 12, 12, 13",
                      "isosceles",
                      result);
}

/** Test 7 - for "Scalene" triangle.  **/
@Test public void scaleneTest1() {

  Triangle t = new Triangle(1, 2, Math.sqrt(2));
  Assert.assertEquals("\nSides: " + 1 + " " + 2 + " " + Math.sqrt
                  "scalene",
                  t.classify());
}
```

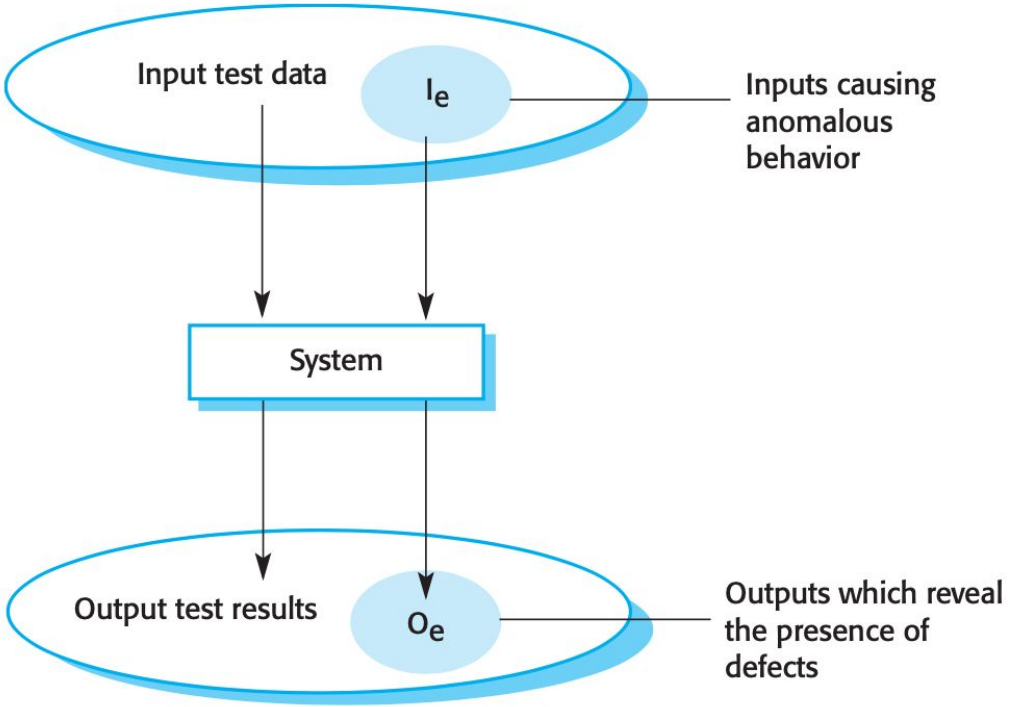
Pruebas Negativas

Pruebas normales

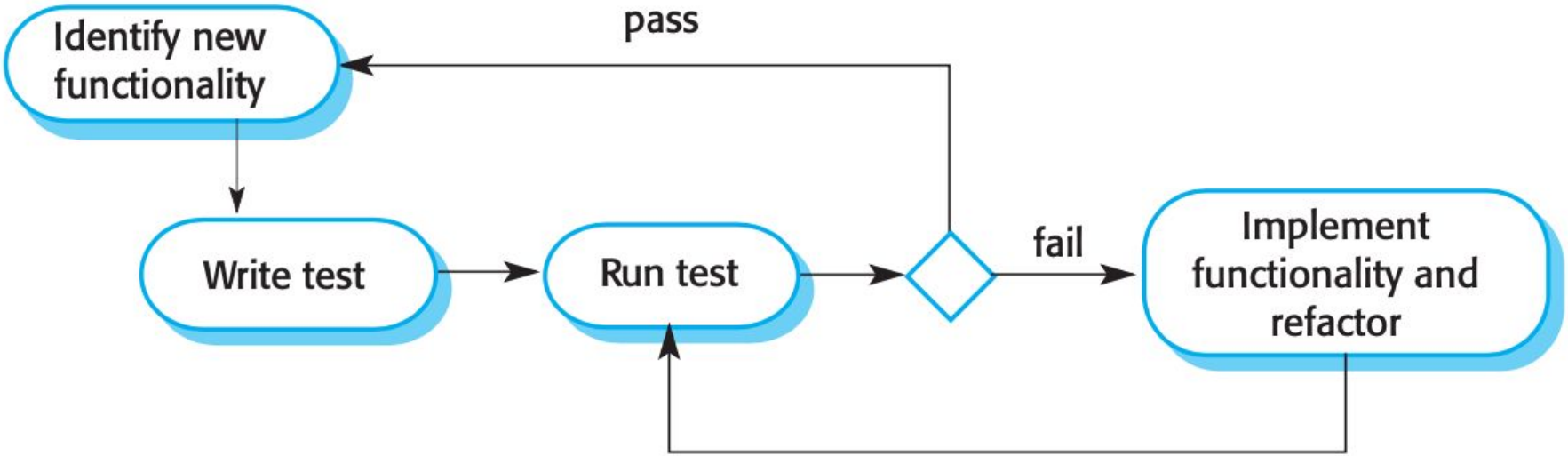
Component testing(Integration)



System testing



8.2 TDD - Test-driven development (XP)

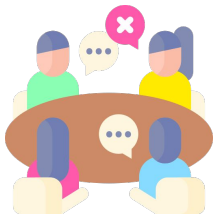

Identify new functionality
pass
Write test
Run test
fail
Implement functionality and refactor

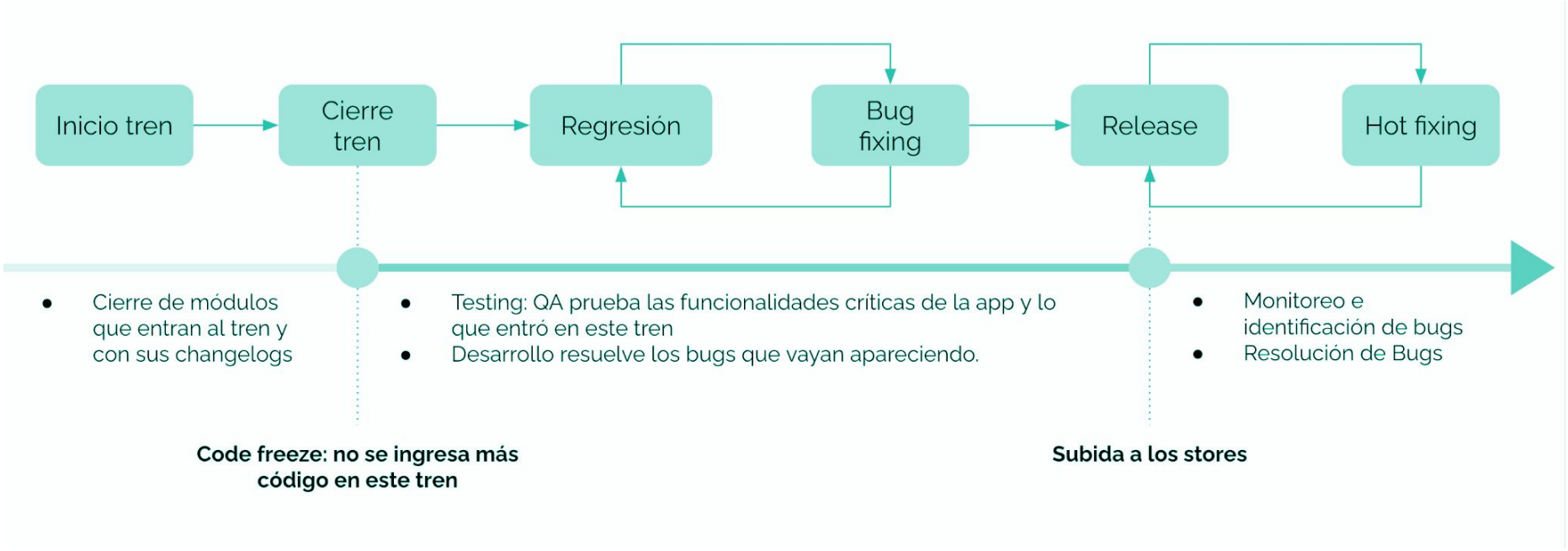
BDD - Behavior Driven Development

(Gherkin + Cucumber)

```
Feature: Login
  As a user
  I want to be able to log in to the application
  So that I can access my account information

  Scenario: Successful login
    Given I am on the login page
    When I enter my username and password
    And I click the login button
    Then I should be taken to the dashboard page
```

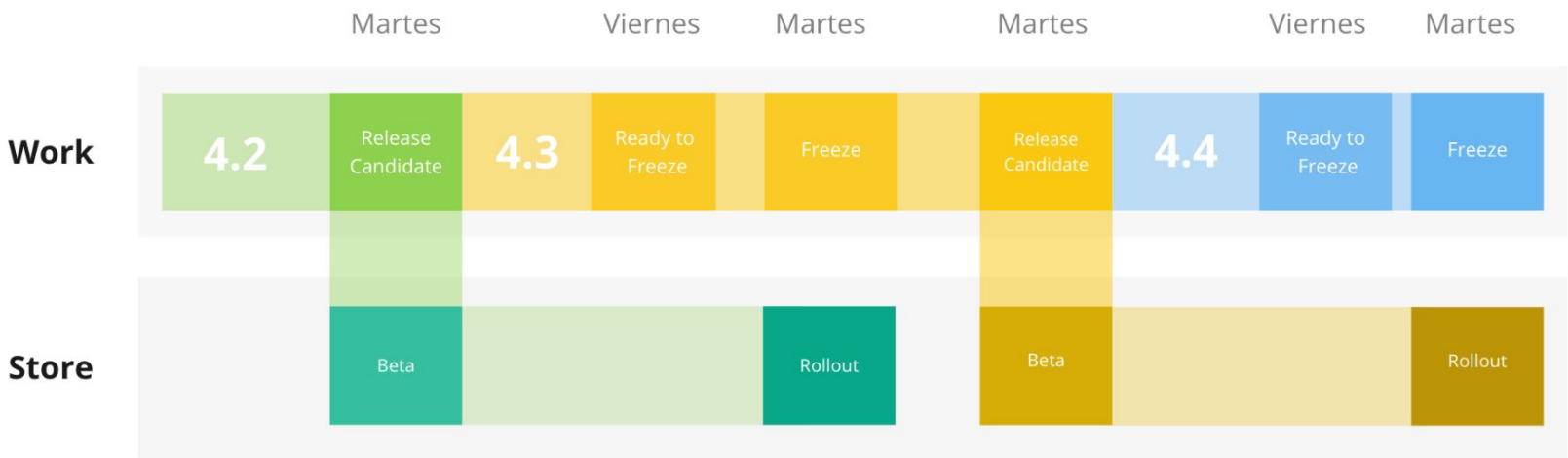
8.3 Release testing



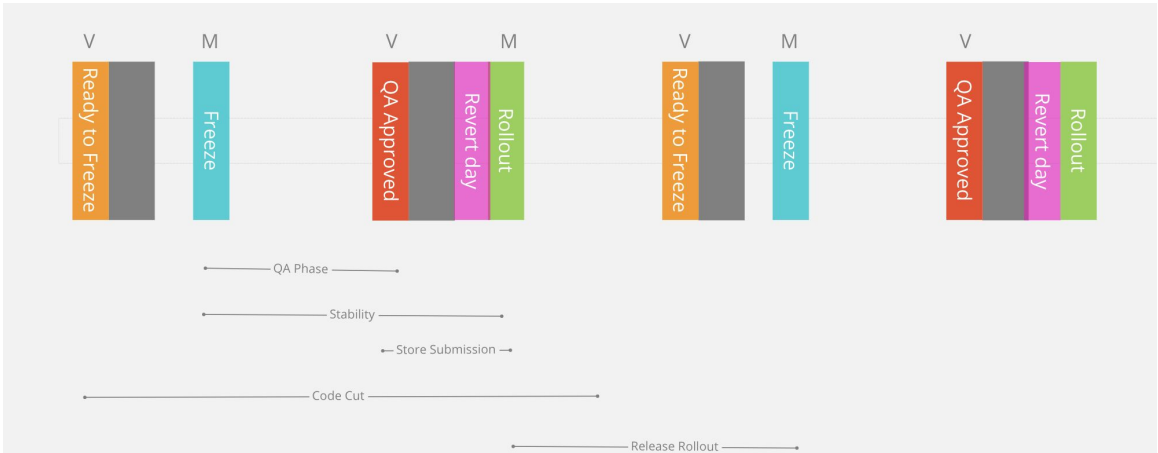

Inicio tren
Cierre tren
Regresión
Bug fixing
Release
Hot fixing
Cierre de módulos que entran al tren y con sus changelogs
Testing: QA prueba las funcionalidades críticas de la app y lo que entró en este tren
Desarrollo resuelve los bugs que vayan apareciendo.
Monitoreo e identificación de bugs
Resolución de Bugs
Code freeze: no se ingresa más código en este tren
Subida a los stores

8.4 User testing

Next Releases Planning



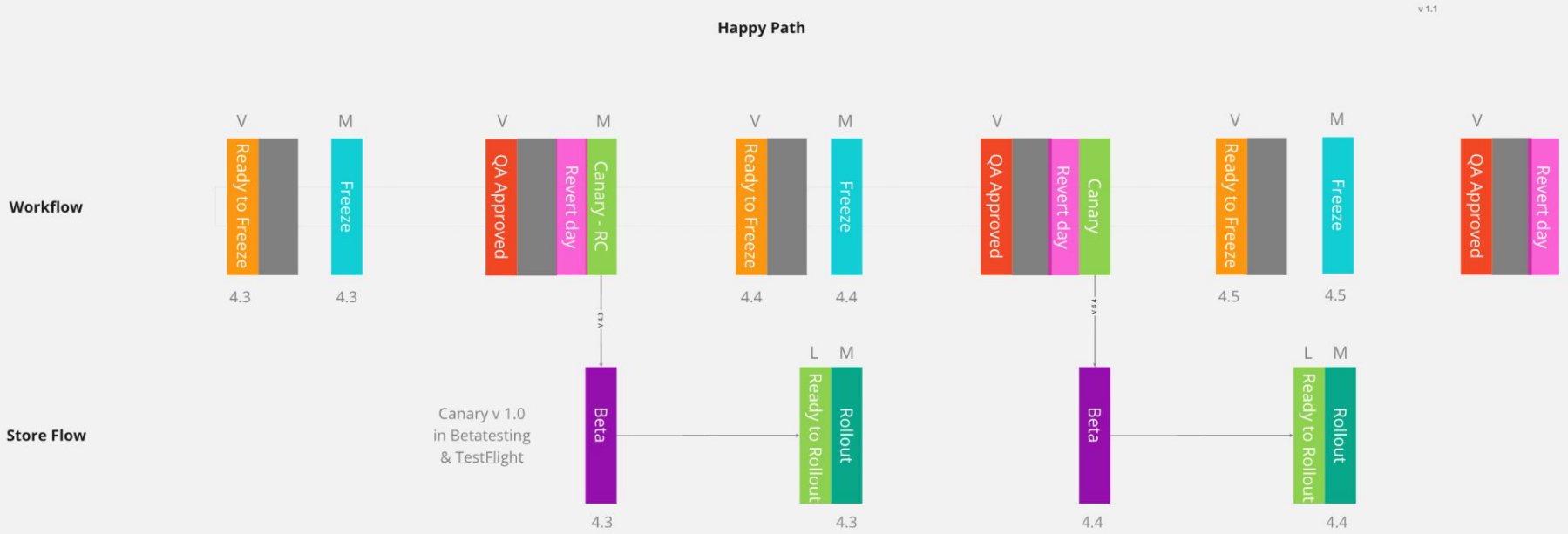
🚃 Release Train



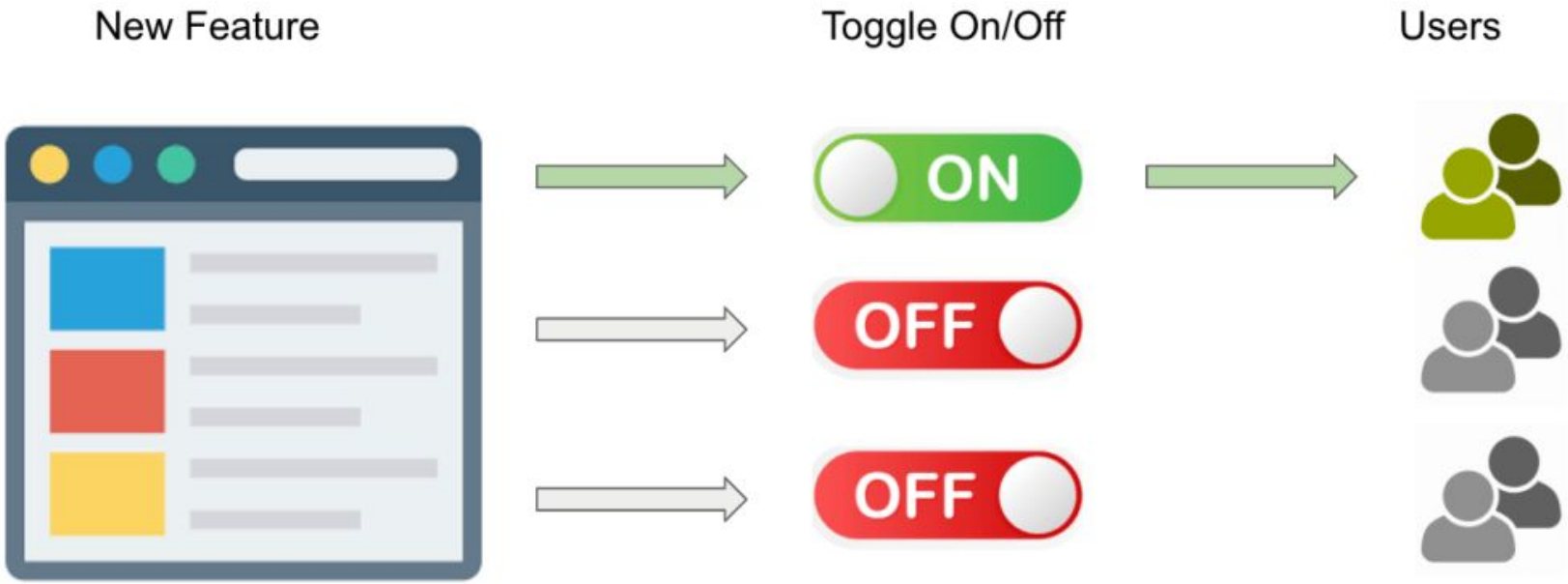
State of Modules in Release Cycle (Tablero de Jira)

To Do, In Progress, Ready to Freeze, Frozen, QA Approved, Bug Fixing, Reversed, Ready to Rollout, Released

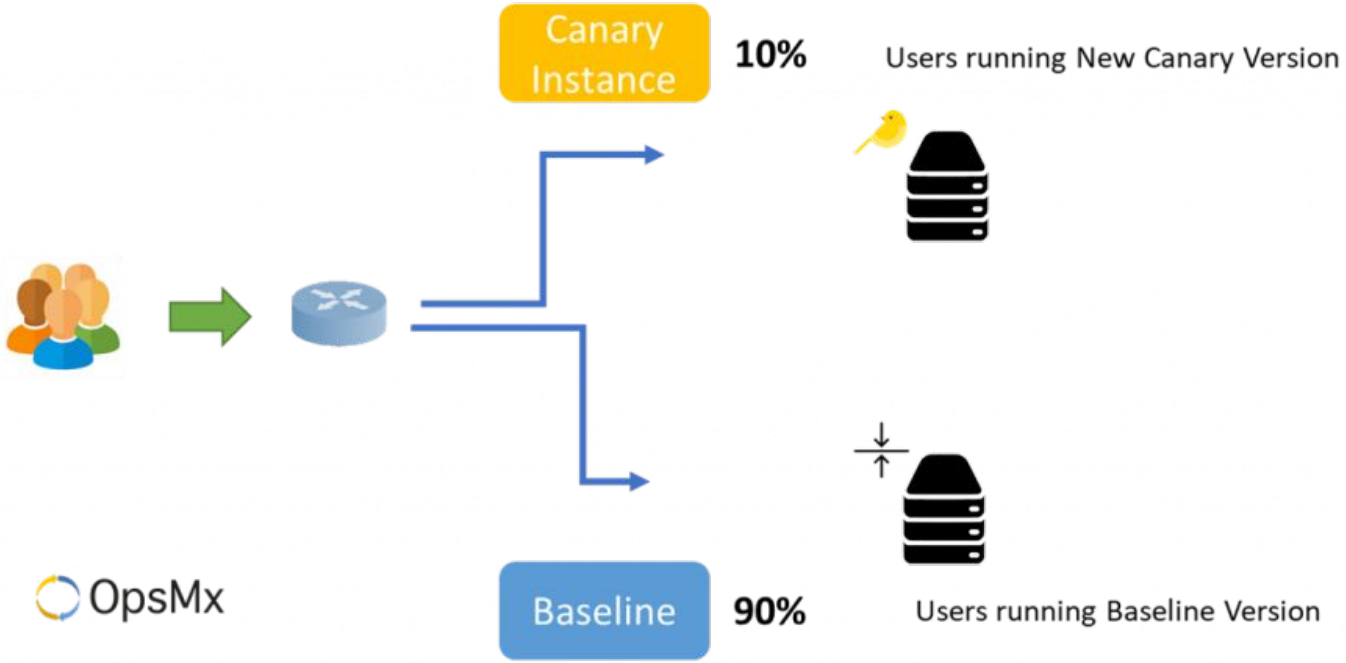
Beta/dogfood Release



Dark Launch - Feature Flag



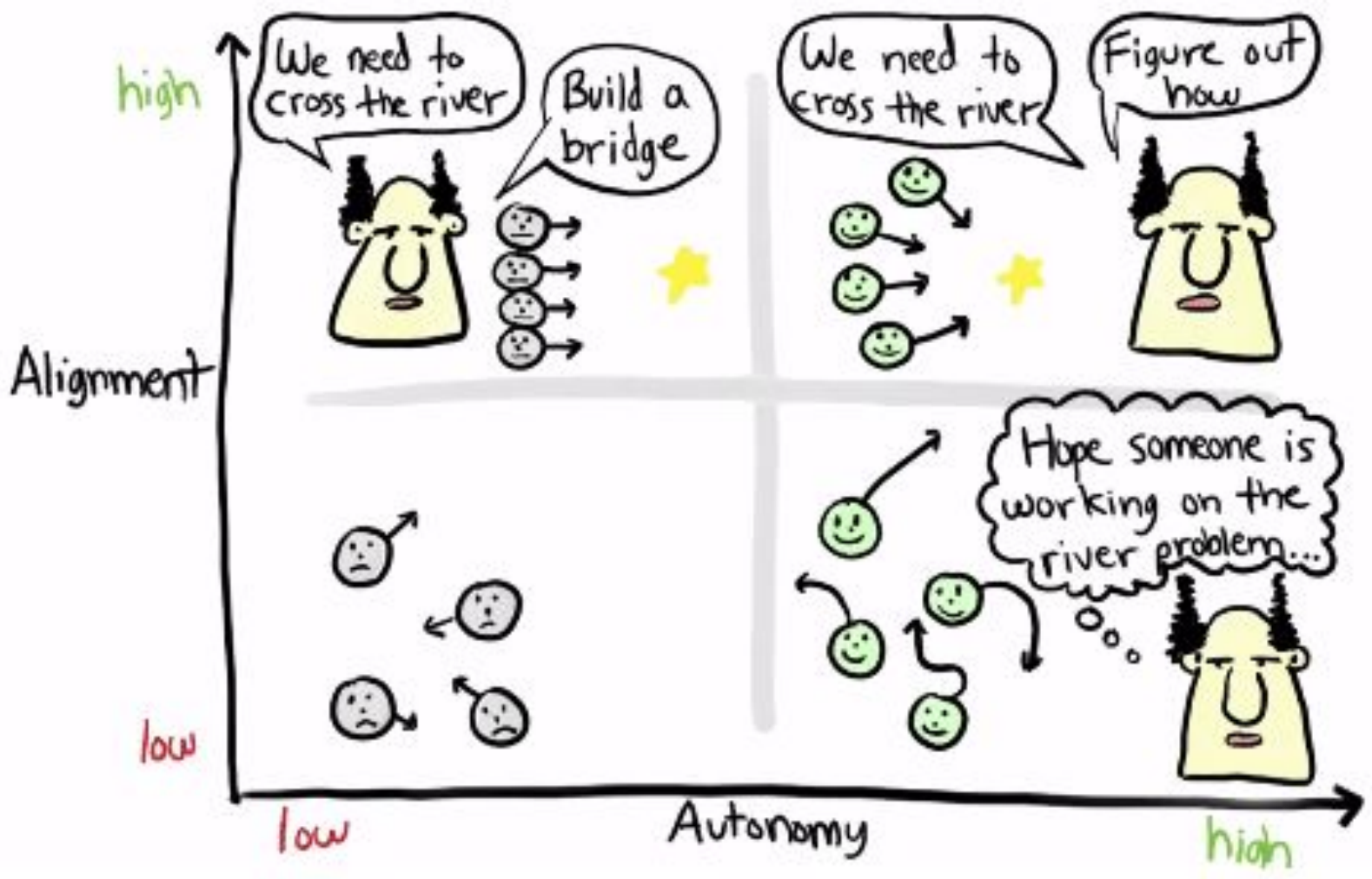
Canary

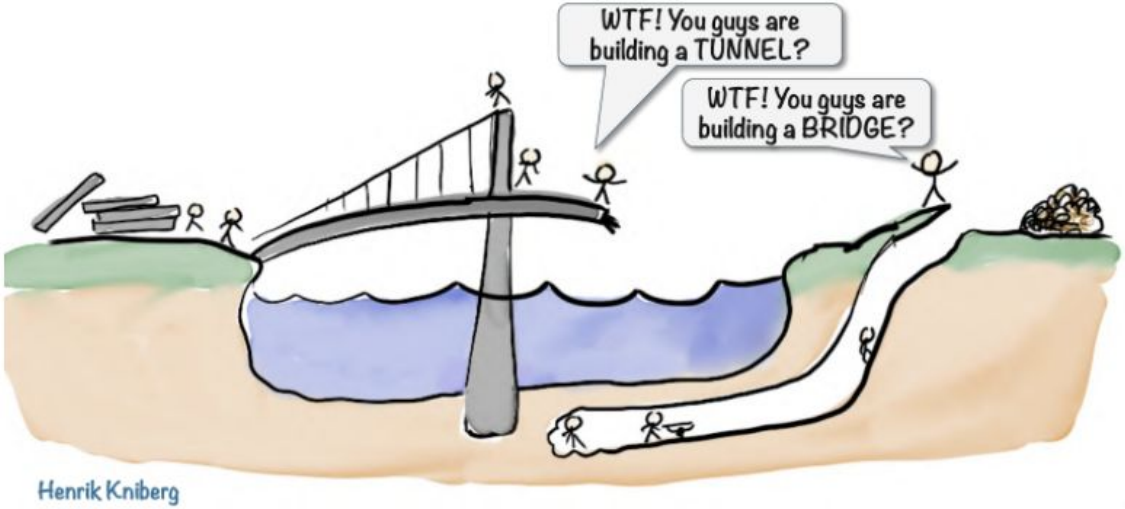

Canary Instance
10%
Users running New Canary Version
Baseline
90%
Users running Baseline Version
OpsMx

Preguntas?

Contexto


Alignment
Autonomy


high
Alignment
low
low
Autonomy
high
We need to cross the river
Build a bridge
We need to cross the river
Figure out how
Hope someone is working on the river problem...



OKR misalignment

Modelos Tradicionales

Waterfall

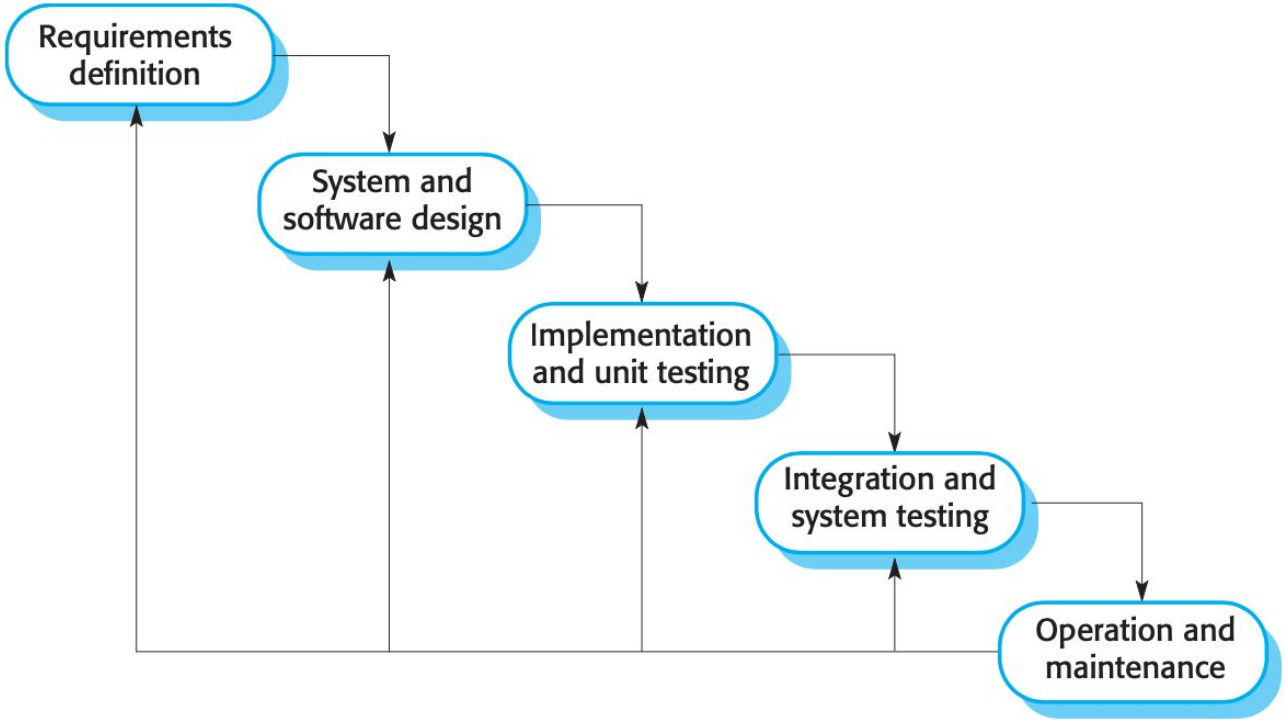
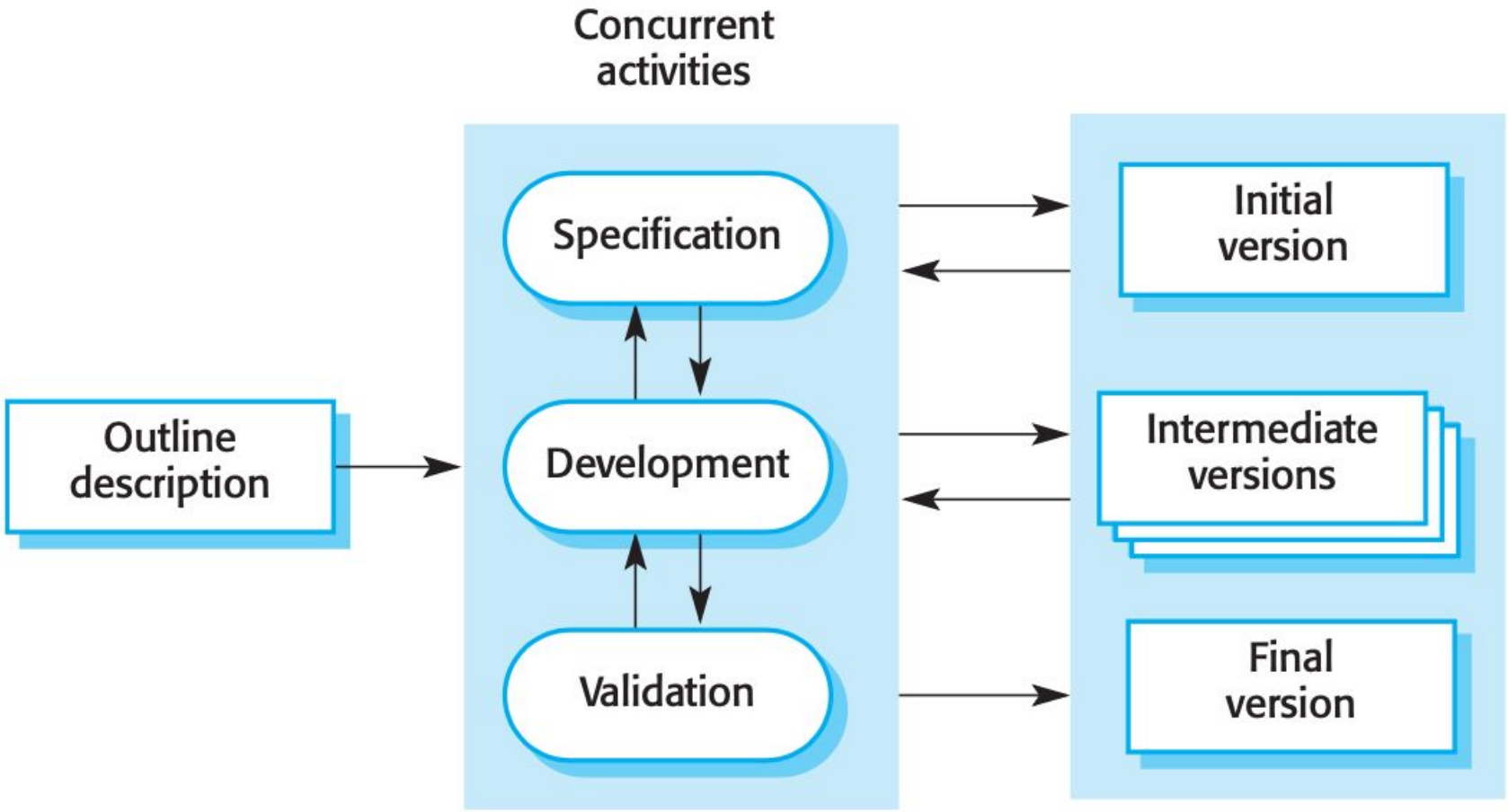
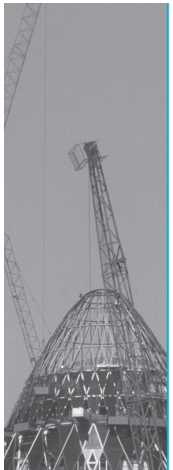


Figure 2.1 The waterfall model

Incremental


Concurrent activities
Specification
Development
Validation
Outline description
Initial version
Intermediate versions
Final version

Agile Software Development



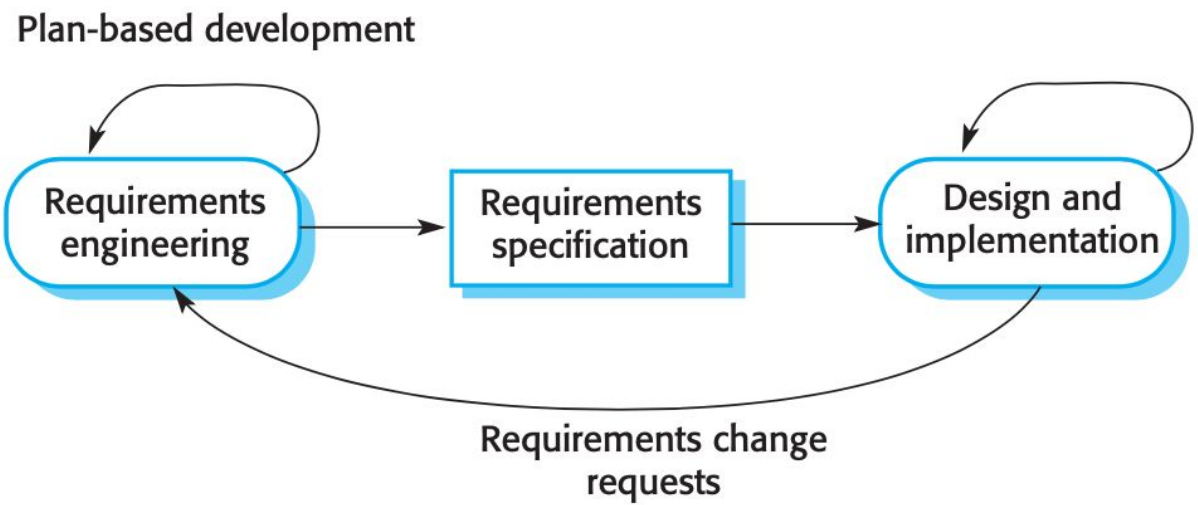
3

Agile software development

Objectives

The objective of this chapter is to introduce you to agile software development methods. When you have read the chapter, you will:

- understand the rationale for agile software development methods, the agile manifesto, and the differences between agile and plan-driven development;
- know about important agile development practices such as user stories, refactoring, pair programming and test-first development;
- understand the Scrum approach to agile project management;
- understand the issues of scaling agile development methods and combining agile approaches with plan-driven approaches in the development of large software systems.


Plan-based development
Requirements engineering
Requirements specification
Design and implementation
Requirements change requests
Agile development
Requirements engineering
Design and implementation

Agile Manifesto

2001

4 Principios

Individuals and interactions over processes and tools

Working software over comprehensive documentation

Customer collaboration over contract negotiation

Responding to change over following a plan

That is, while there is value in the items on
the right, we value the items on the **left more.**

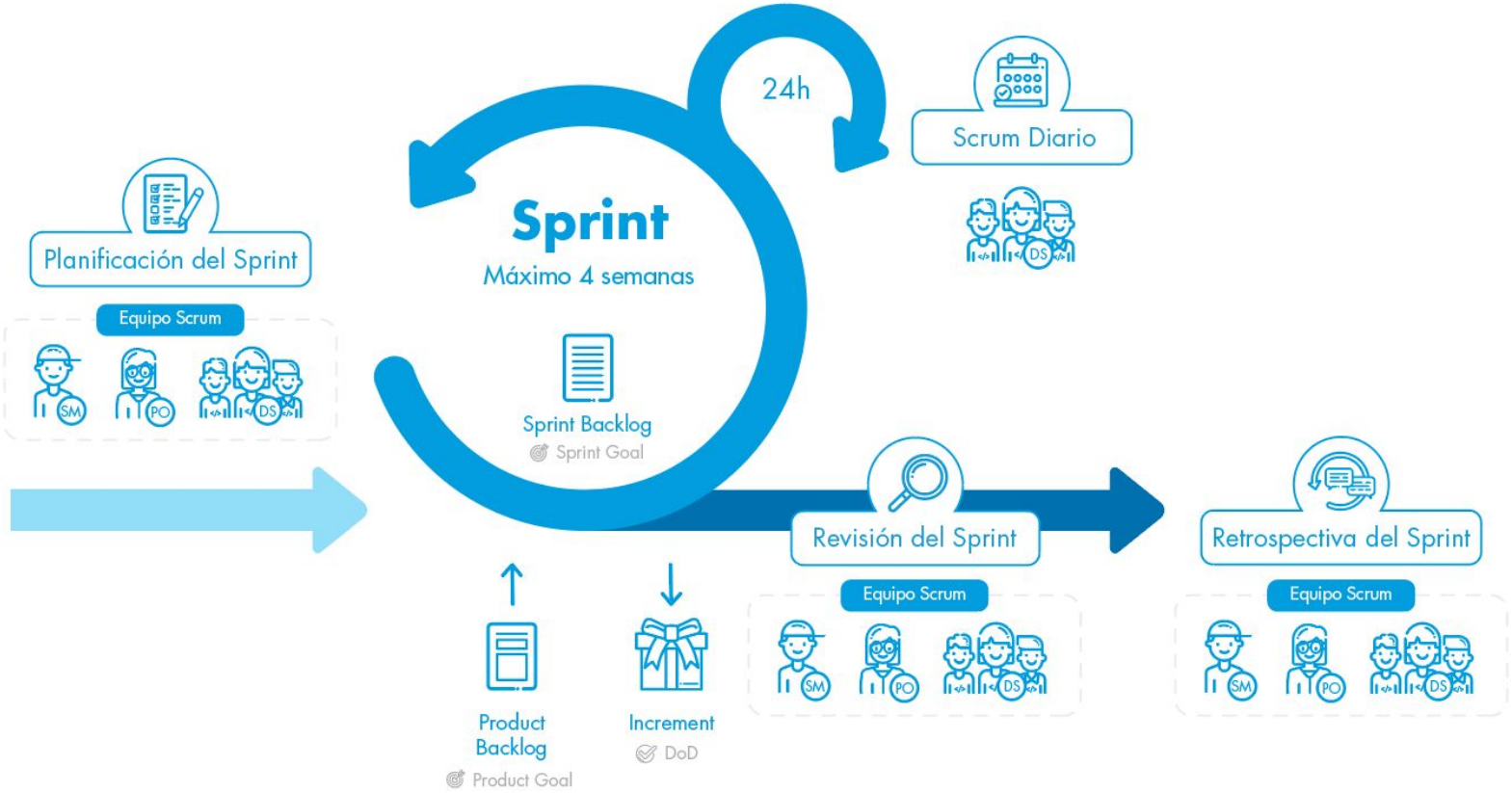
Scrum

Ken Schwaber, Mike Beedle and Jeff Sutherland

Scrum is a **lightweight framework** that helps people, teams and organizations **generate value** through **adaptive solutions** for complex problems.

Valores

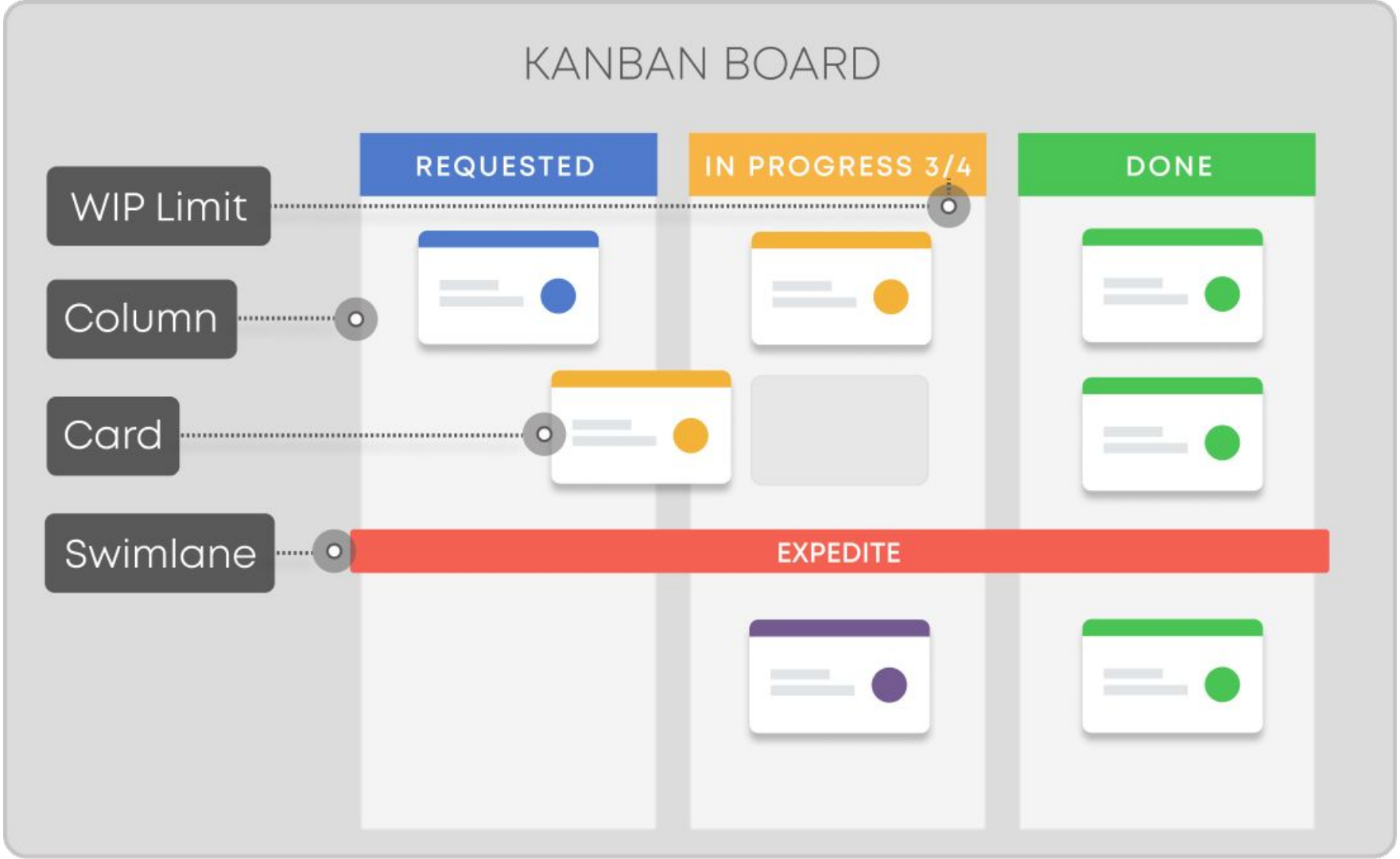
- Compromiso
- Foco
- Franqueza
- Respeto
- Coraje

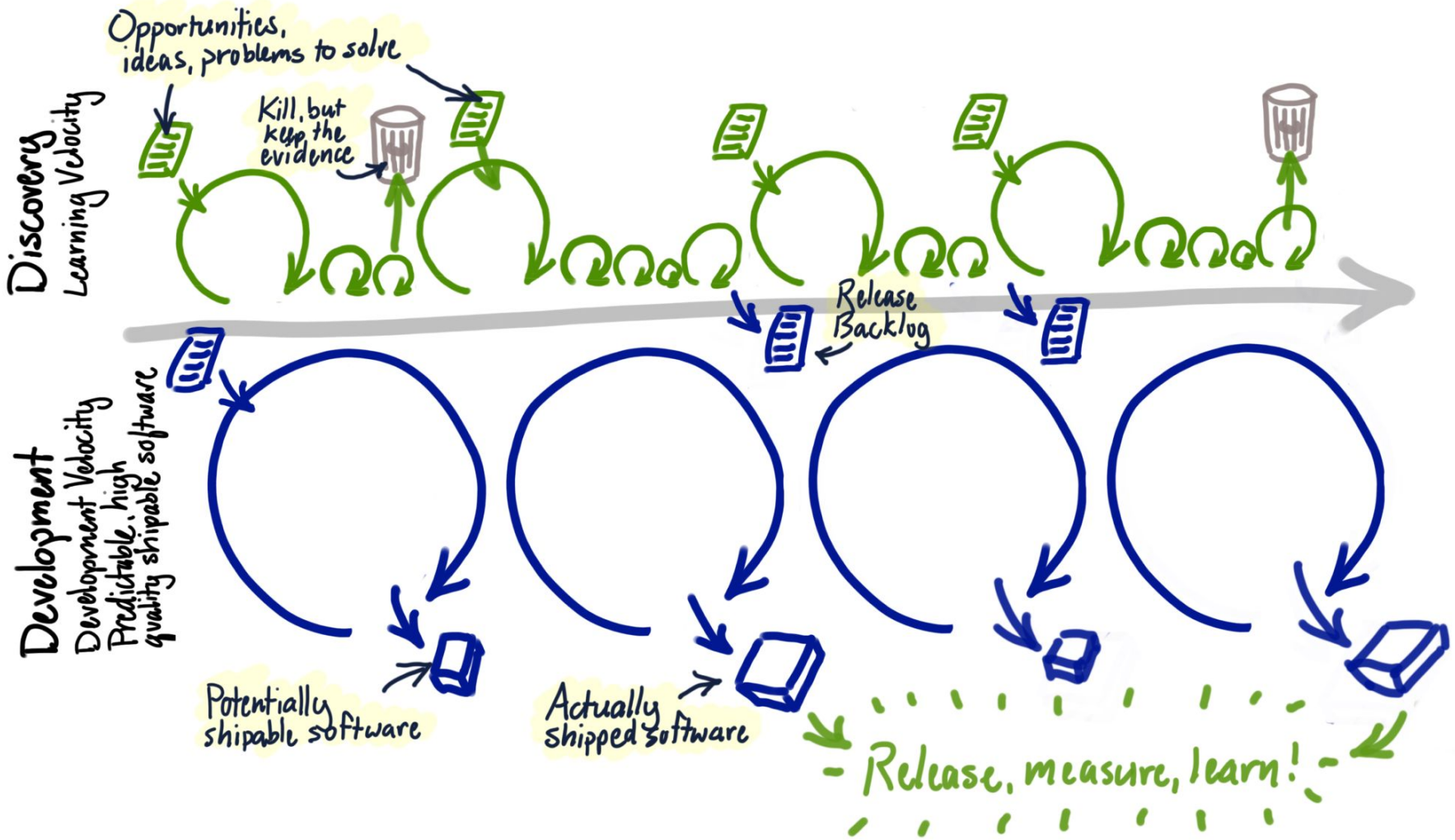


https://netmind.net/es/scrum-el-pasado-y-el-futuro/

Kanban

Taiichi Ohno, an industrial engineer at Toyota,

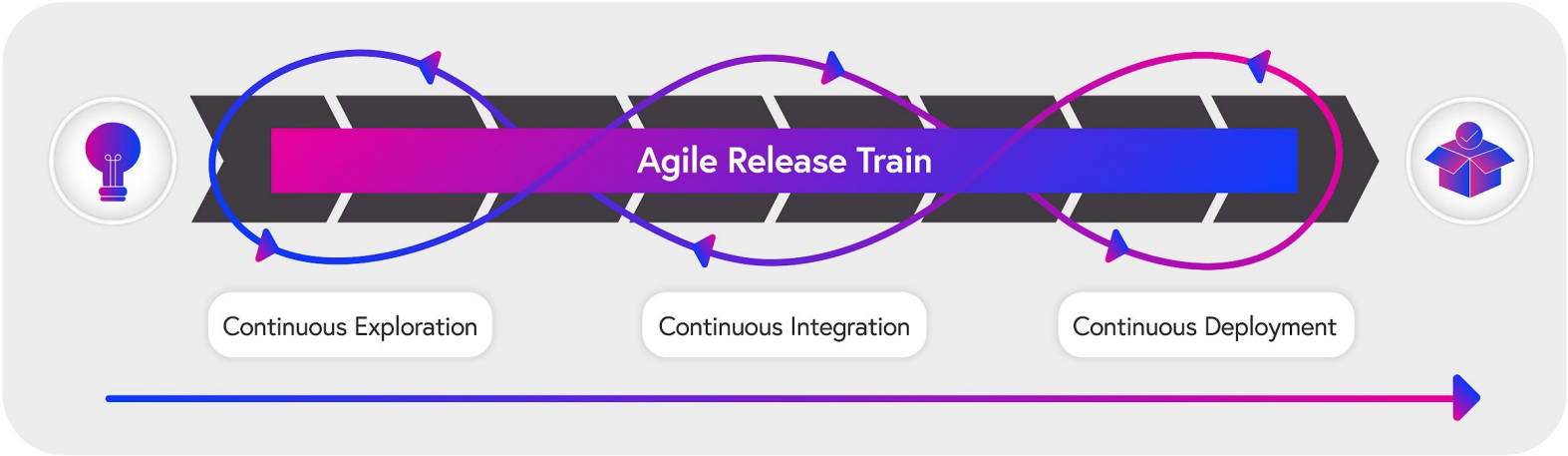

KANBAN BOARD
REQUESTED
IN PROGRESS 3/4
DONE
WIP Limit
Column
Card
Swimlane
EXPEDITE


Opportunities, ideas, problems to solve
Kill, but keep the evidence
Discovery
Learning Velocity
Release Backlog
Development
Development Velocity
Predictable, high quality shipable software
Potentially shipable software
Actually shipped software
Release, measure, learn!

Release Process

,

Continuous Delivery Pipeline



Release train/Build Train