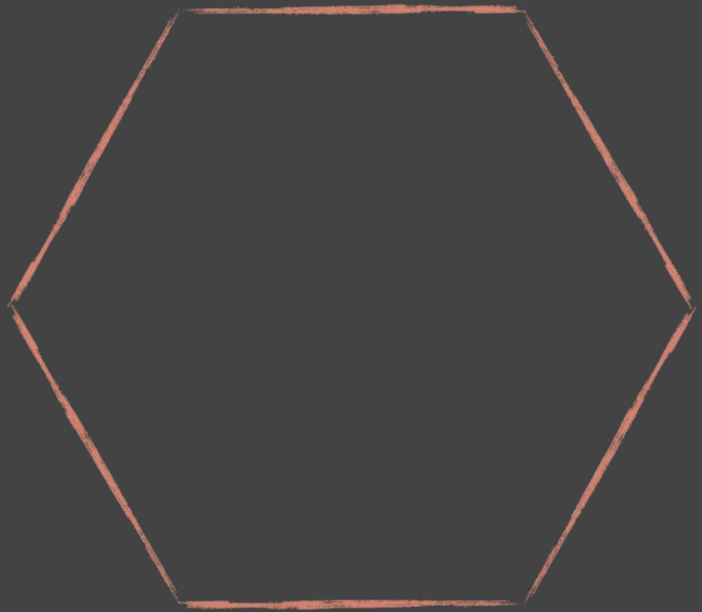
Hexagonal Architecture

a step to be a craftsman

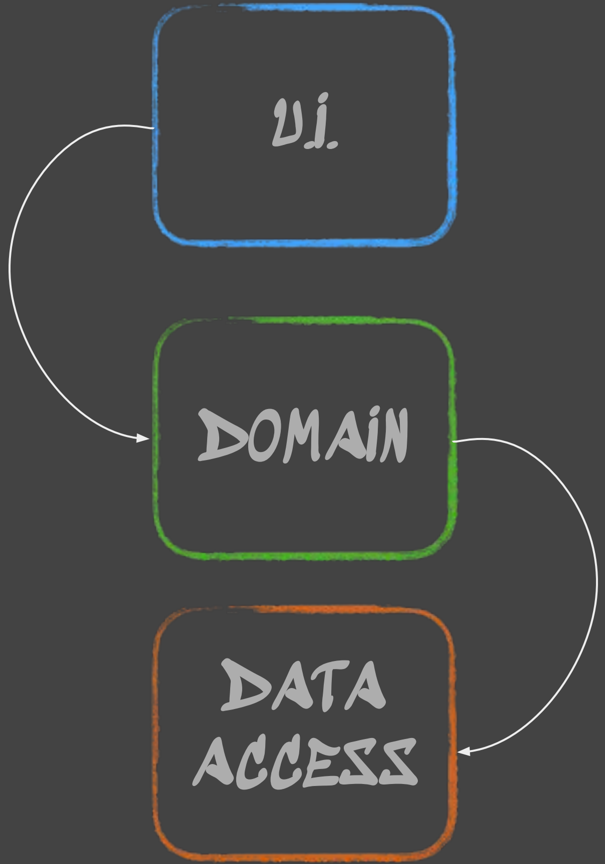
Yann Danot @ydanot

Olivier Albiez @OlivierAlbiez

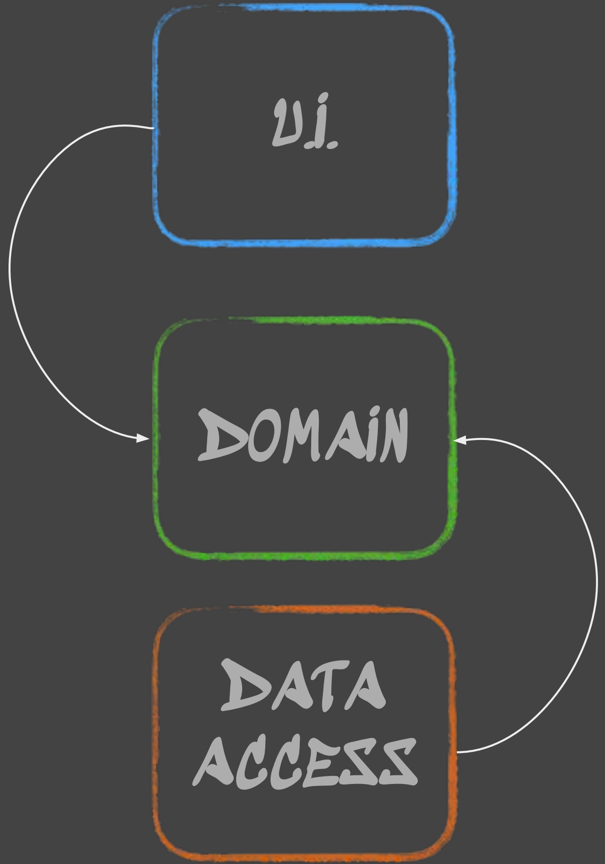
HEXAGONAL ?



LAYERS WITHOUT DIP



LAYERS WITH DIP



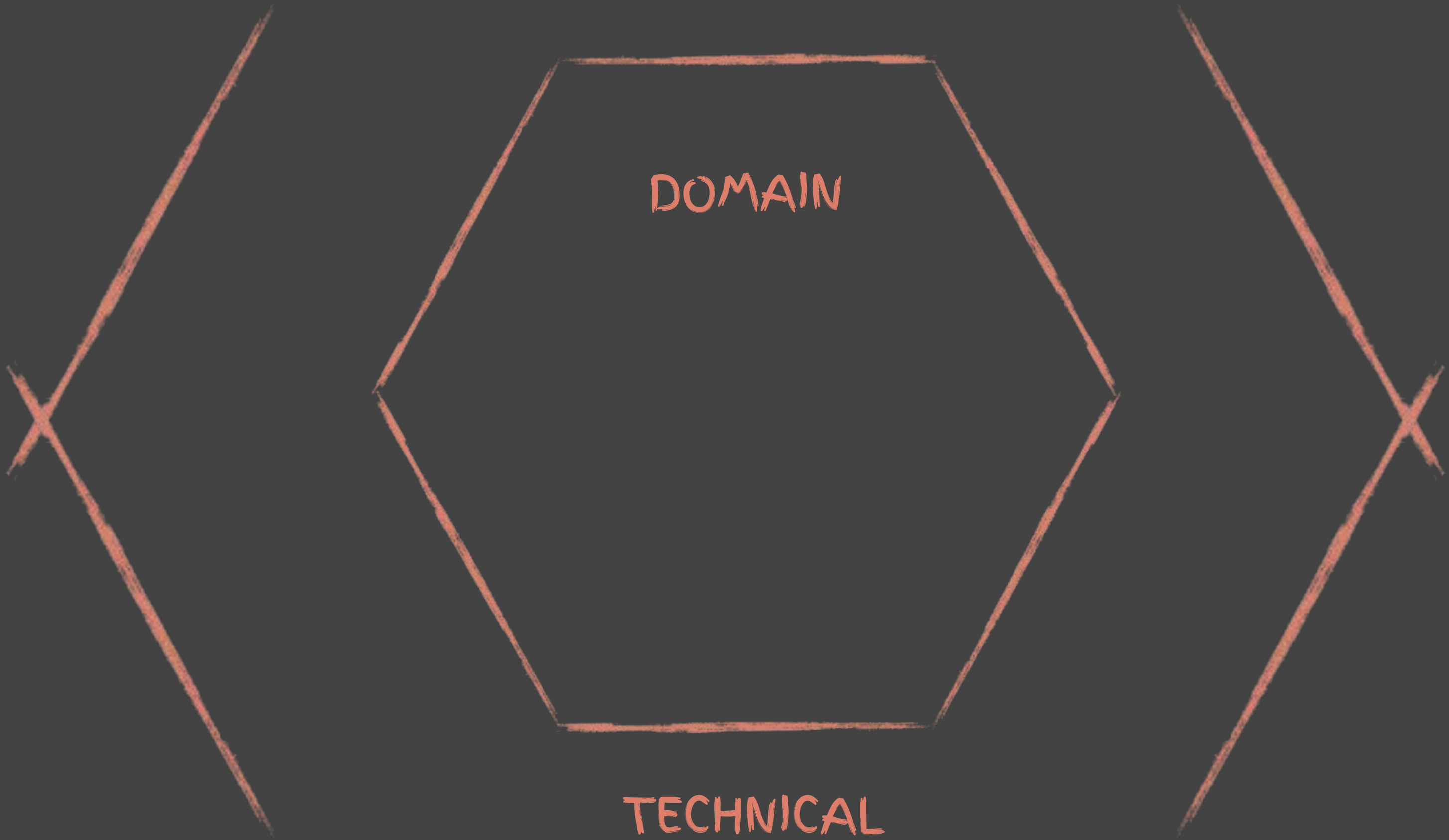
@ploeh

> “Allow an application to equally be driven by users, programs, automated test or batch scripts, and to be developed and tested in isolation from its eventual run-time devices and databases

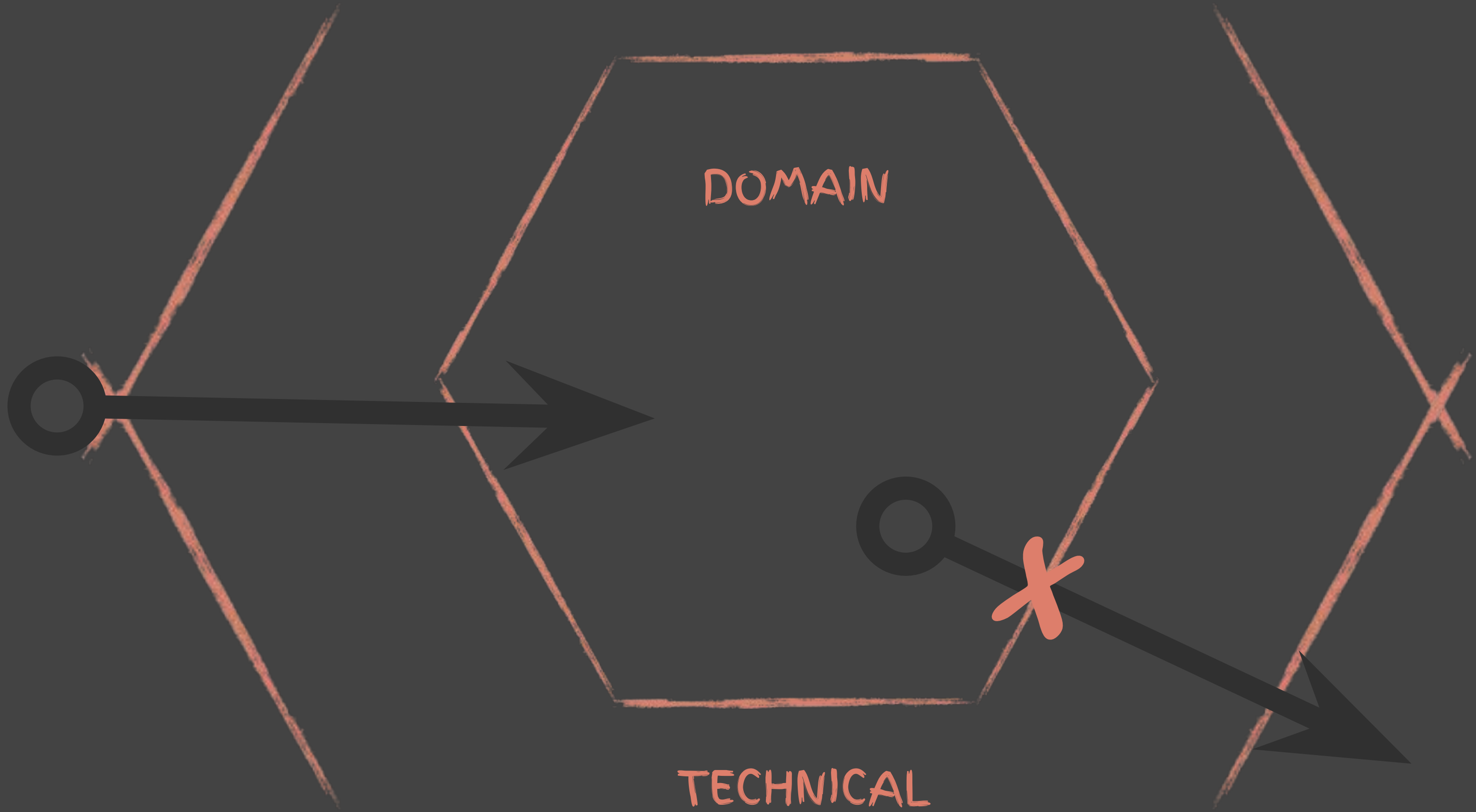
A. Cockburn

Segregregregate business and technical logic

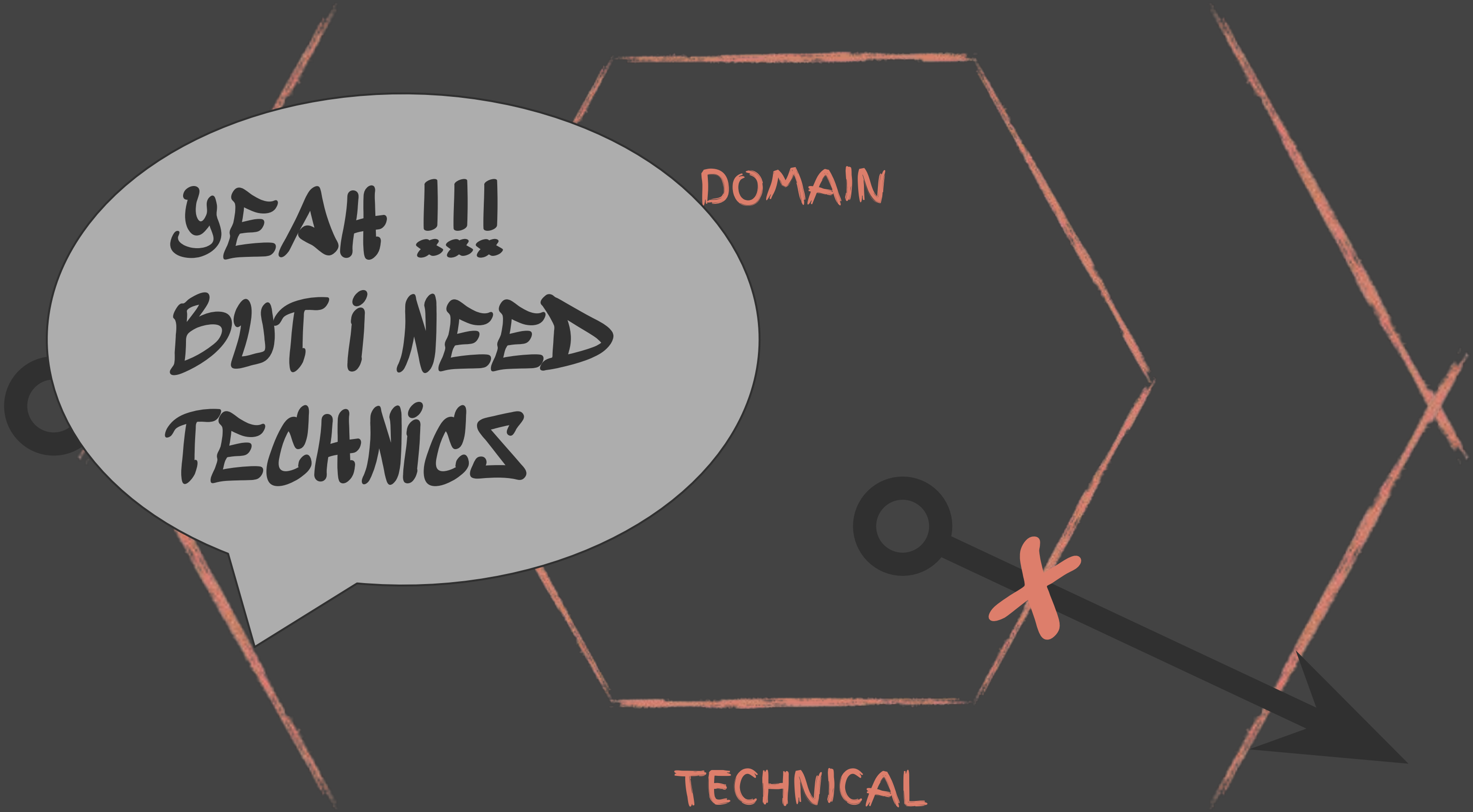
SEGREGATE BUSINESS AND TECHNICAL



DEPENDENCIES FROM OUTSIDE



DEPENDENCIES FROM OUTSIDE



S.O.L.I.D

Dependency Inversion Principle

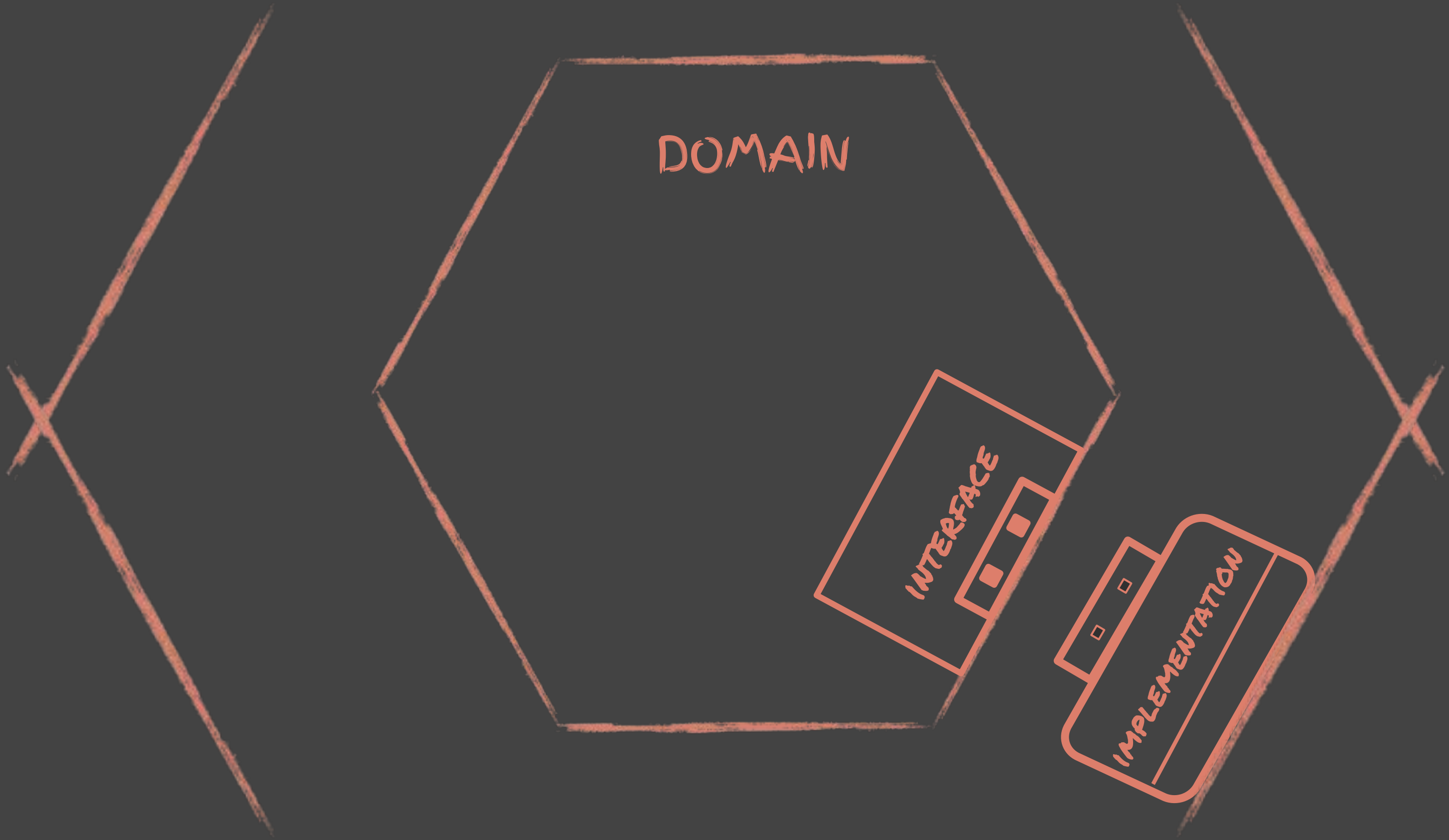
WHAT DO YOU THINK ?

```java
public class OrderProcessor {

   public double calculateTotal(Order order, Connection cnx) throws SQLException {
       double itemTotal = order.getItemTotal();
       double discountAmount = DiscountCalculator.calculateDiscount(order);
       double taxAmount = 0.0d;
       if (order.getCountry() == US)
           taxAmount = findTaxAmount(order, cnx);
       else if (order.getCountry() == UK)
           taxAmount = findVatAmount(order, cnx);
       double total = itemTotal - discountAmount + taxAmount;
       return total;
   }

   private double findVatAmount(Order order, Connection cnx) throws SQLException {
       Resources r = new Resources();
       try {
           PreparedStatement statement = r.push(cnx.prepareStatement( "select amount from vat where country=?" ));
           statement.setString(1, order.getCountry().name());
           ResultSet resultSet = r.push(statement.executeQuery());
           return resultSet.getDouble(1);
       }finally {
           r.dispose();
       }
   }

   private double findTaxAmount(Order order, Connection cnx) throws SQLException {
       ...
   }
}
```

DEPENDENCY INVERSION PRINCIPLE

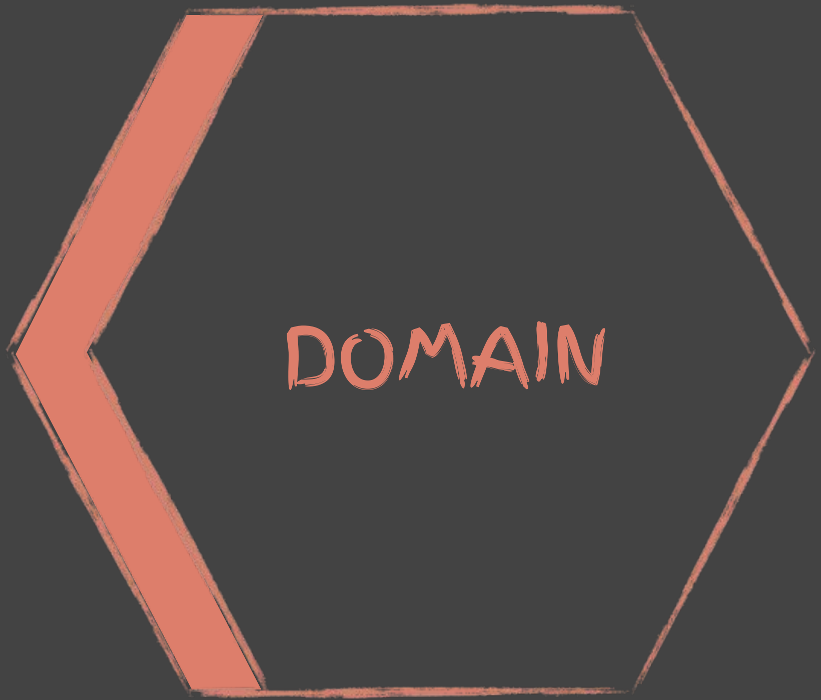
Entities must depend on abstractions not on concretions. It states that the high level module must not depend on the low level module, but they should depend on abstractions.

SEGREGATE BUSINESS AND TECHNICAL



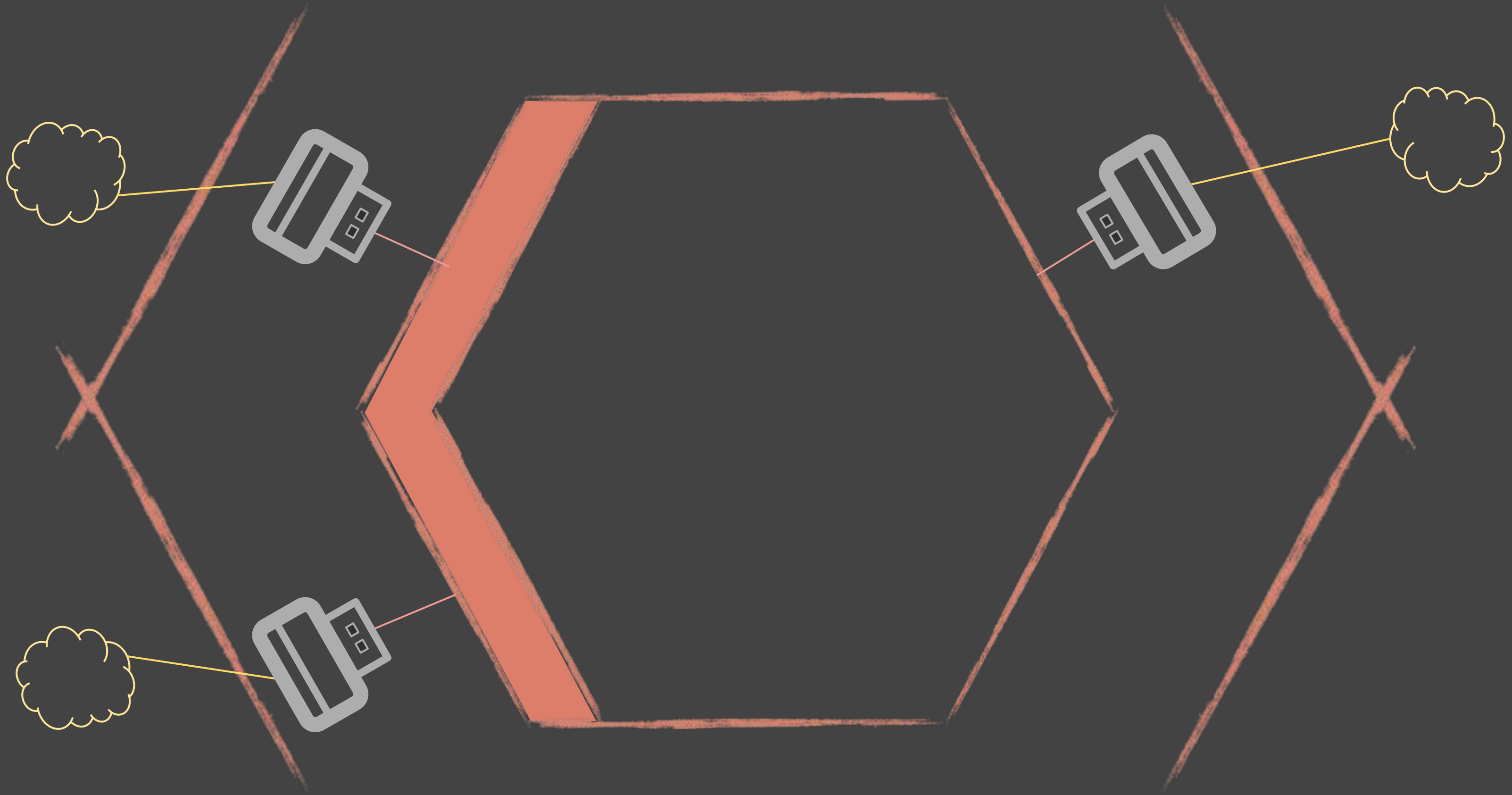
USE CASES DRIVEN ARCHITECTURE

USE CASES / APPLICATION LAYER



ADAPTATION FROM I TO THE WORLD

PORT/ADAPTER



ADAPTER

TESTING THE HEXAGONE

Your test code and production code interact with Hexagone the same way