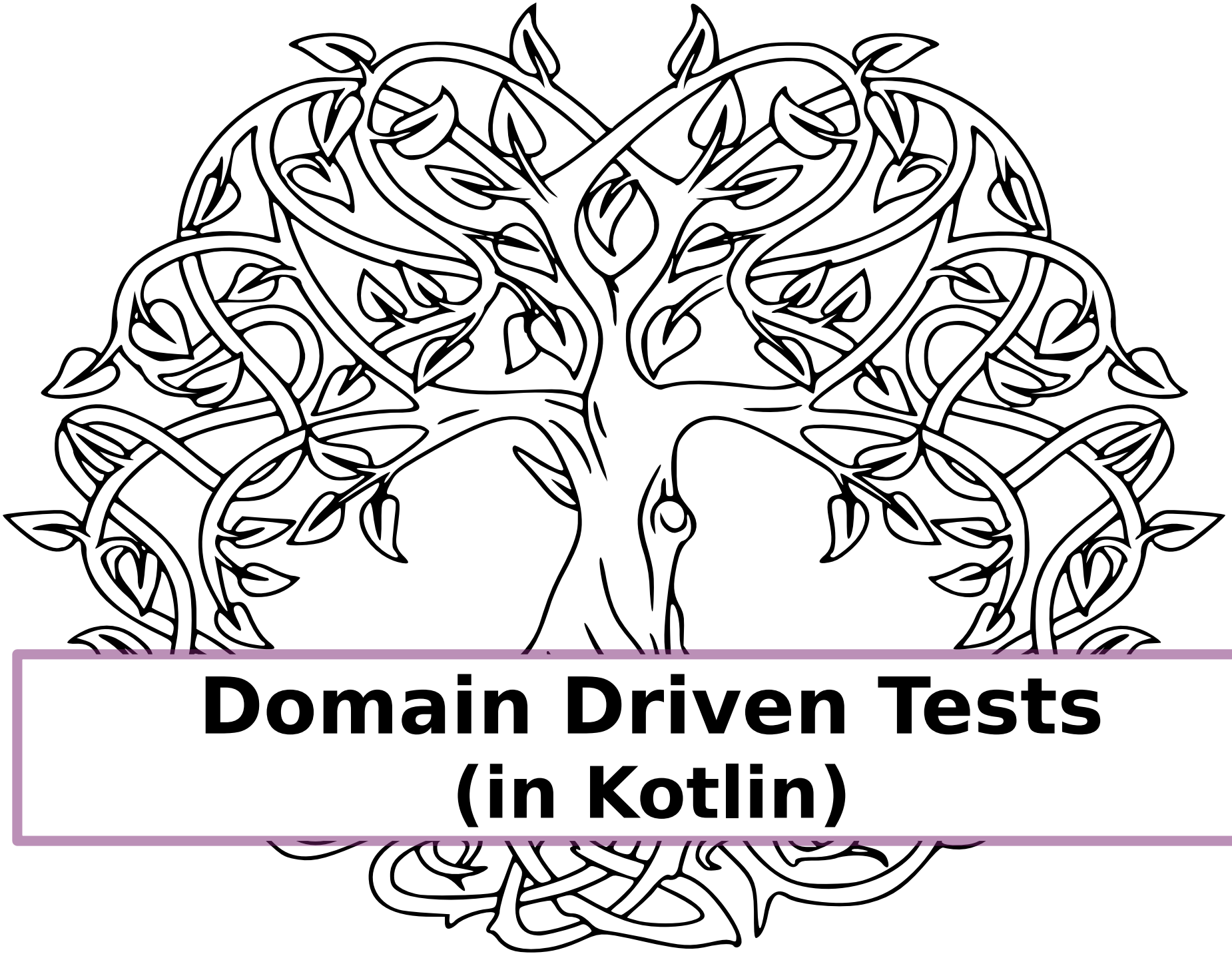


Domain Driven Tests (in Kotlin)

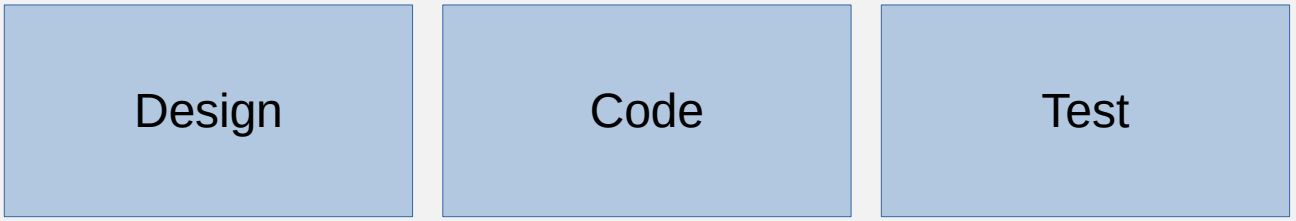
Chi sono

- Nicola.Pedot@informatelier.it
  - Ho lavorato in ambiente aziendale per 15 anni crescendo di ruolo da junior a tech lead.
  - Dal 2015 sviluppo e faccio corsi, in azienda e fuori; affianco gruppi di sviluppo.
  - Temi: Progettazione, Programmazione & Produttività.
  - Da sempre credente nelle Comunità.

InformAtelier

Fasi dello sviluppo

- Scegliendo ordine e priorità su:



Perchè testare

E’ un modo per:

- guadagnare garanzie di qualità;
- documentare;
- fare analisi.

Cosa testare

- Sistema:

  app, journey, funzione, ...
- Integrazione:

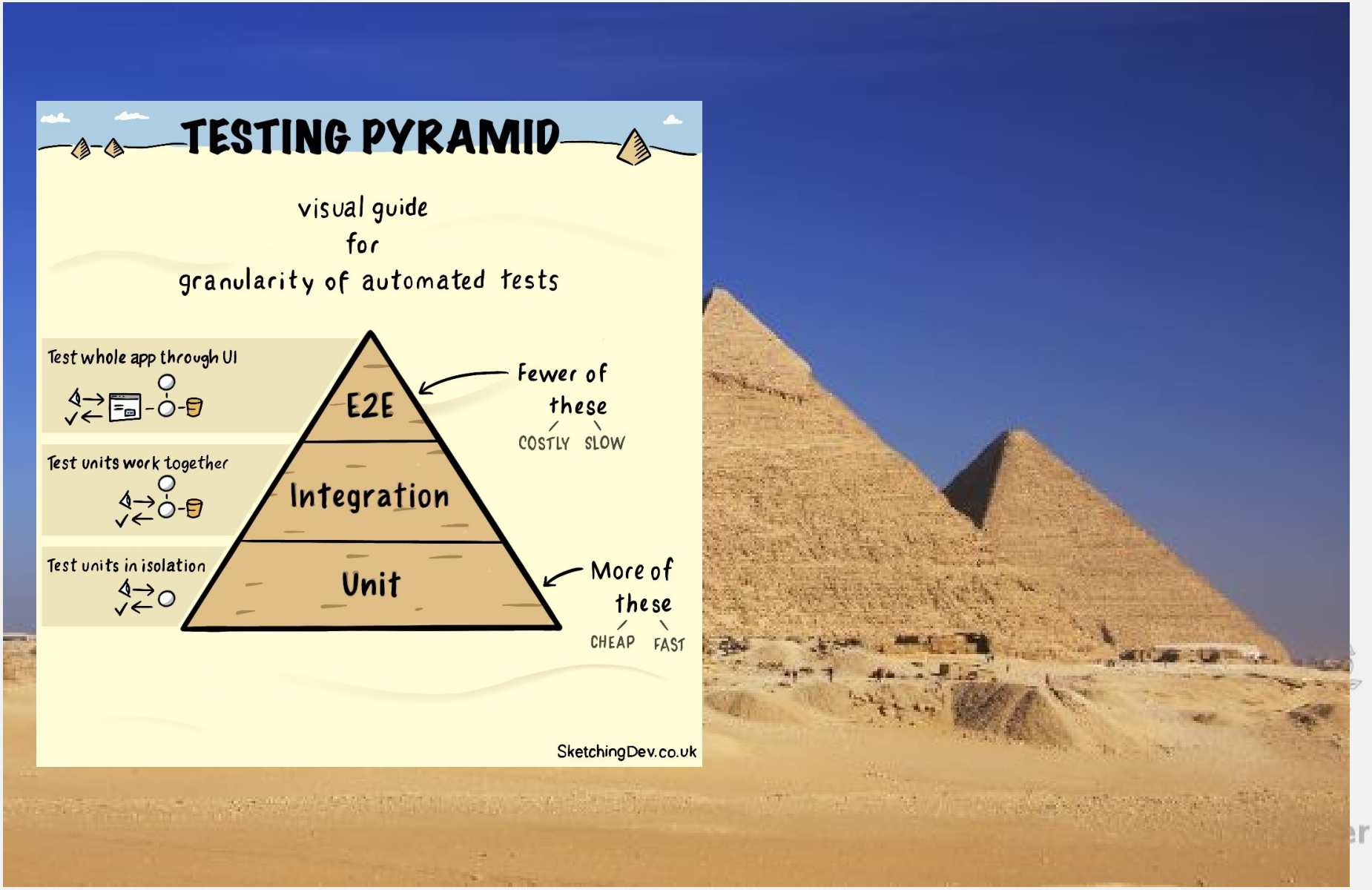
  tra classi, tra sistemi, tra container, ...
- Aspetto:

  log, sicurezza, audit, carico, ...
- Scenario:

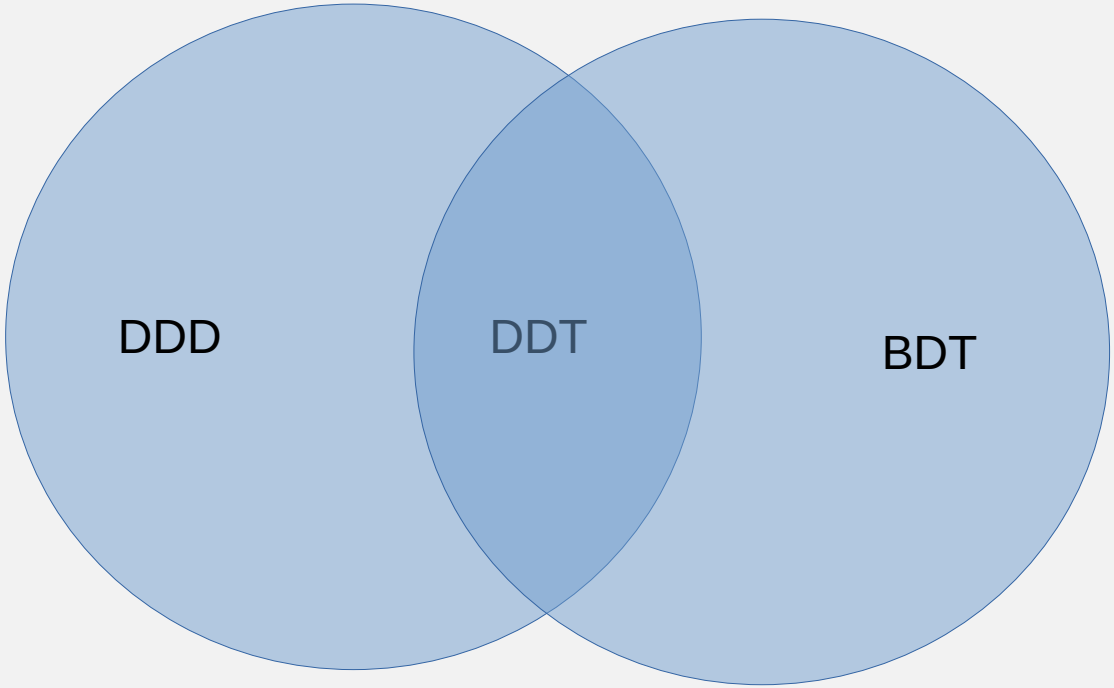
  casi d’uso normali ed eccezionali, ...
- Unità:

  minimo elemento testabile.

Quanto testare



Driven by



Cos’è un dominio

Parte di realtà di cui vogliamo catturare le regole e le strutture.

- “Every software program relates to some activity or interest of its user. That subject area to which the user applies the program is the domain of the software.”
  - Eric Evans

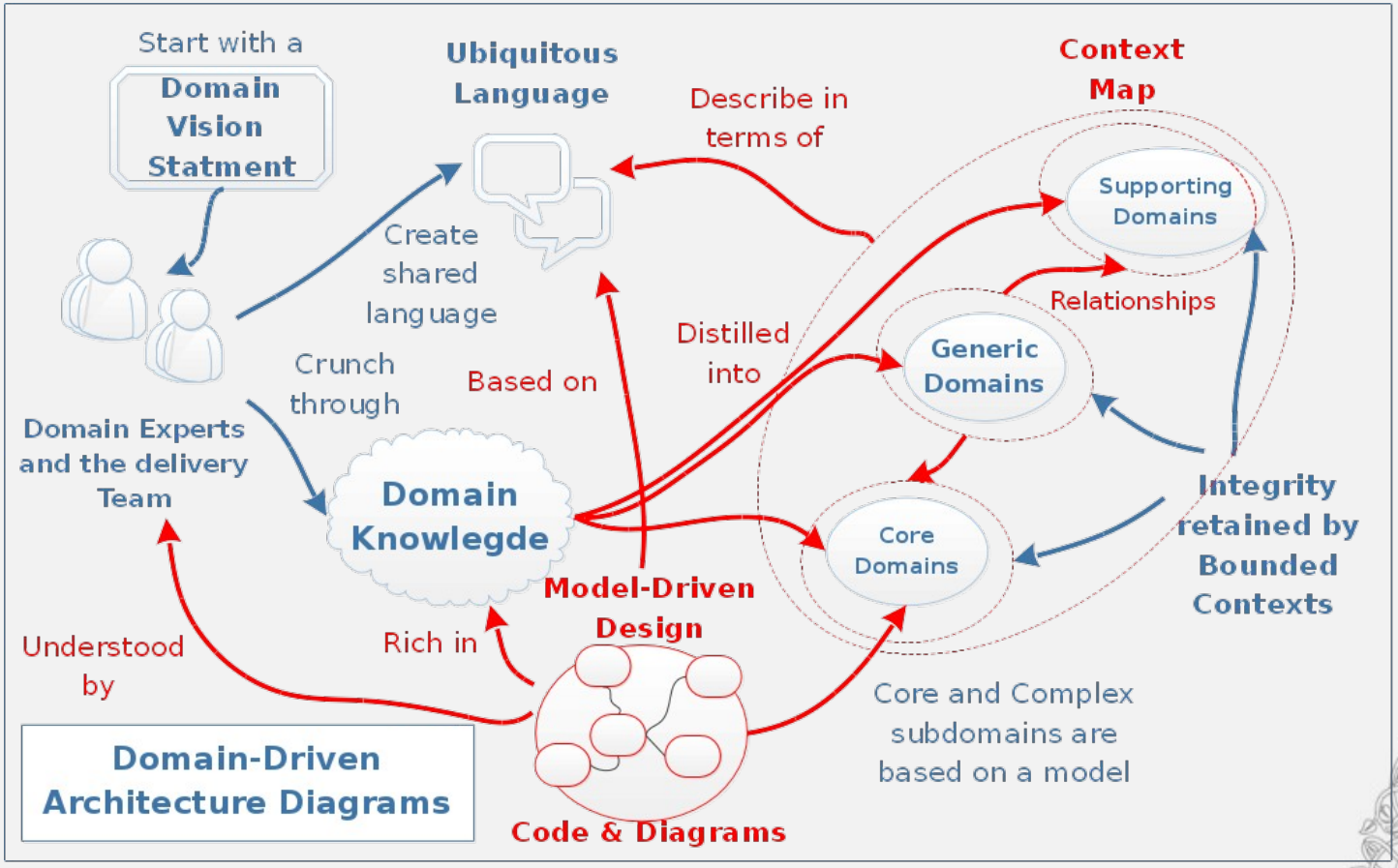
Domain Driven

Usare la conoscenza condivisa sul dominio attorno al tuo prodotto per aiutare ad aumentare la qualità del software.

Conoscenza minima di un dominio

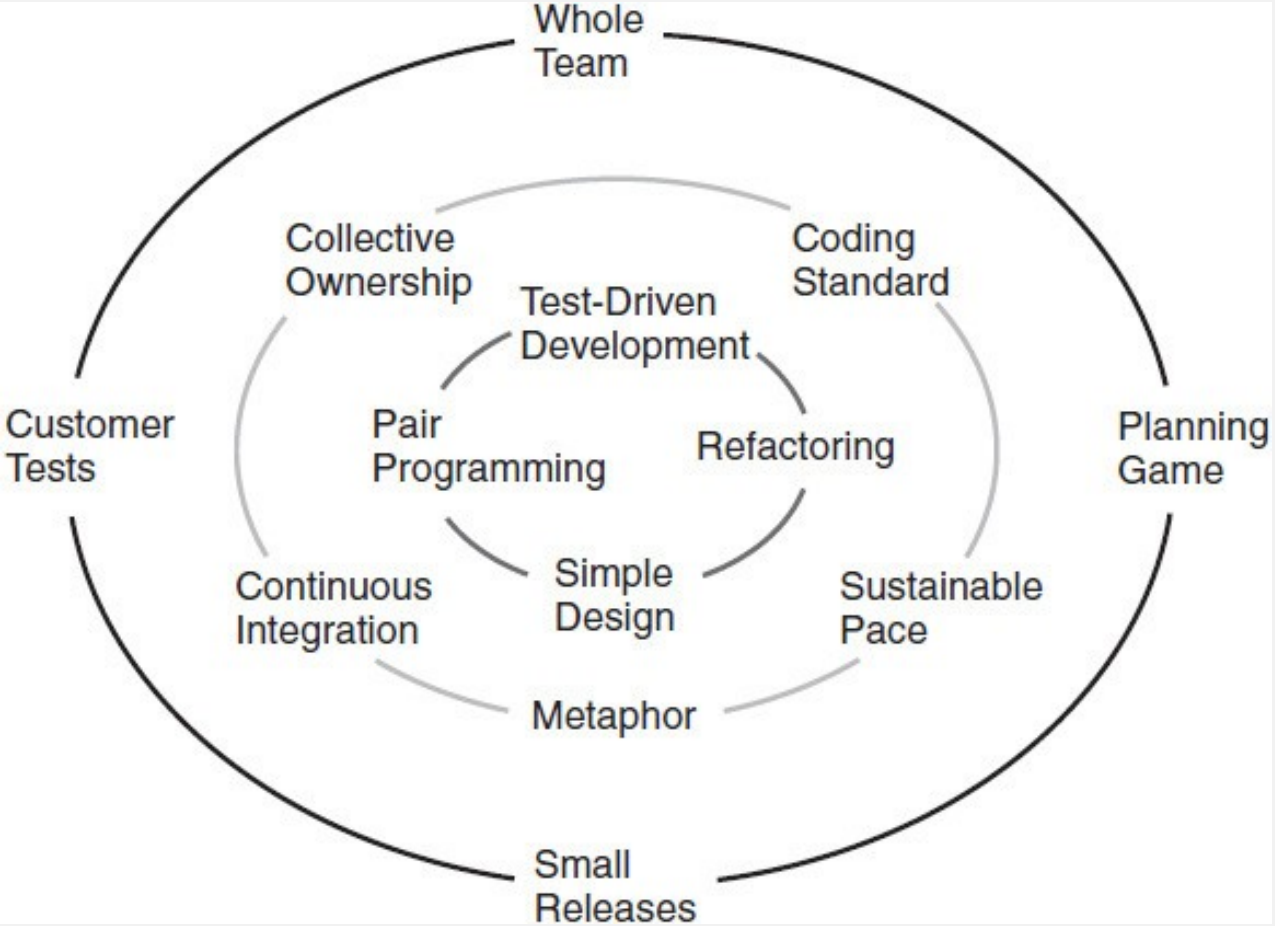
- contesto,
- obiettivo,
- flusso degli eventi,
- termini.

Domain Driven Design overview



https://blog.knoldus.com/reactive-architecture-domain-driven-design-ddd/

XP & Metaphor



https://explainagile.com/agile/xp-extreme-programming/practices/metaphor/

Comunicazione
è
LA
chiave

InformAtelier

Le parole sono pietre

Possono essere usate per:

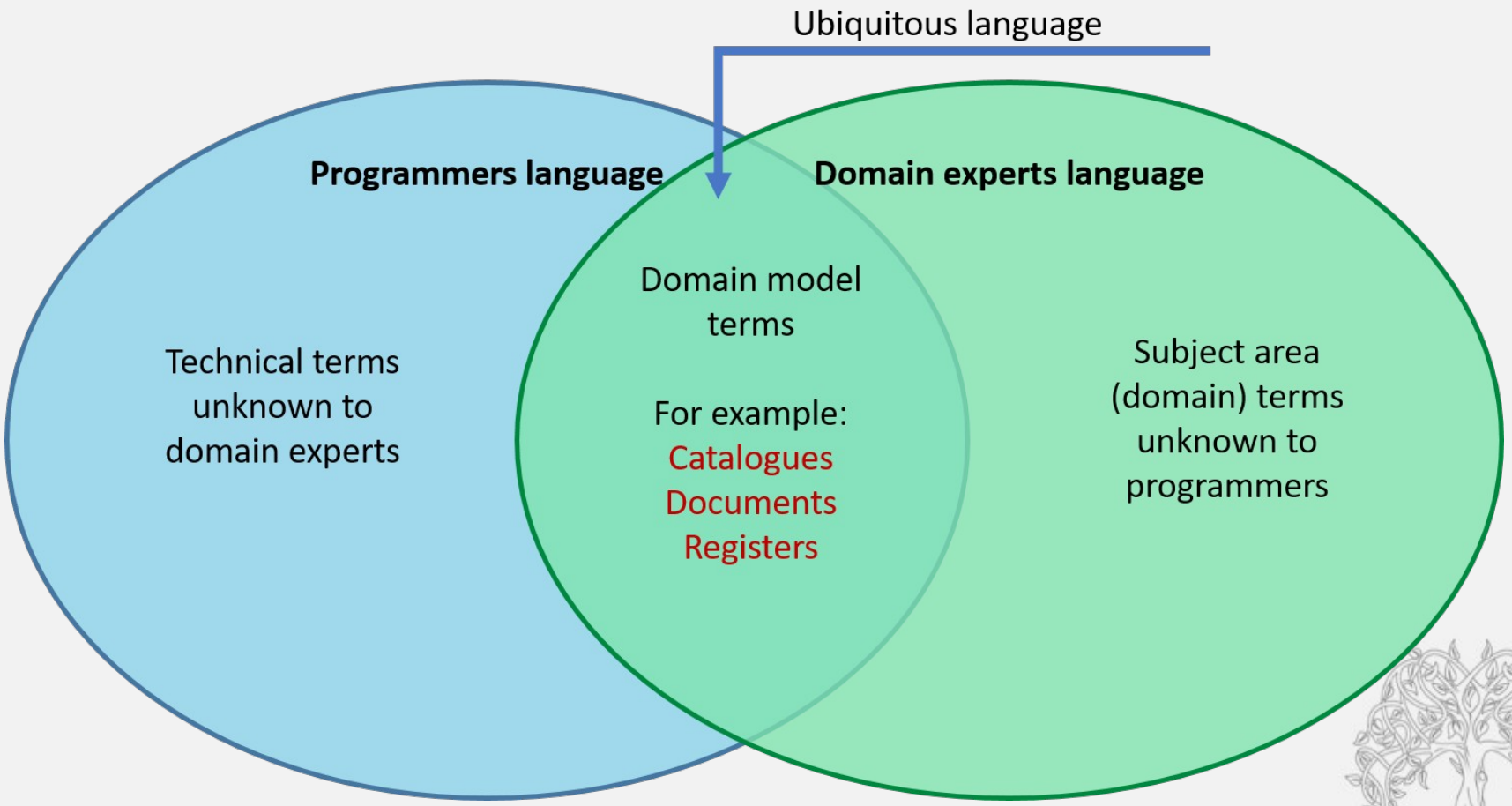
- farsi del male.
- costruire.

- A volte per farsi male costruendo...

InformAtelier

Torre di Babilonia

Ubiquitous Language



Suggerimento (Pratica)

- Un glossario

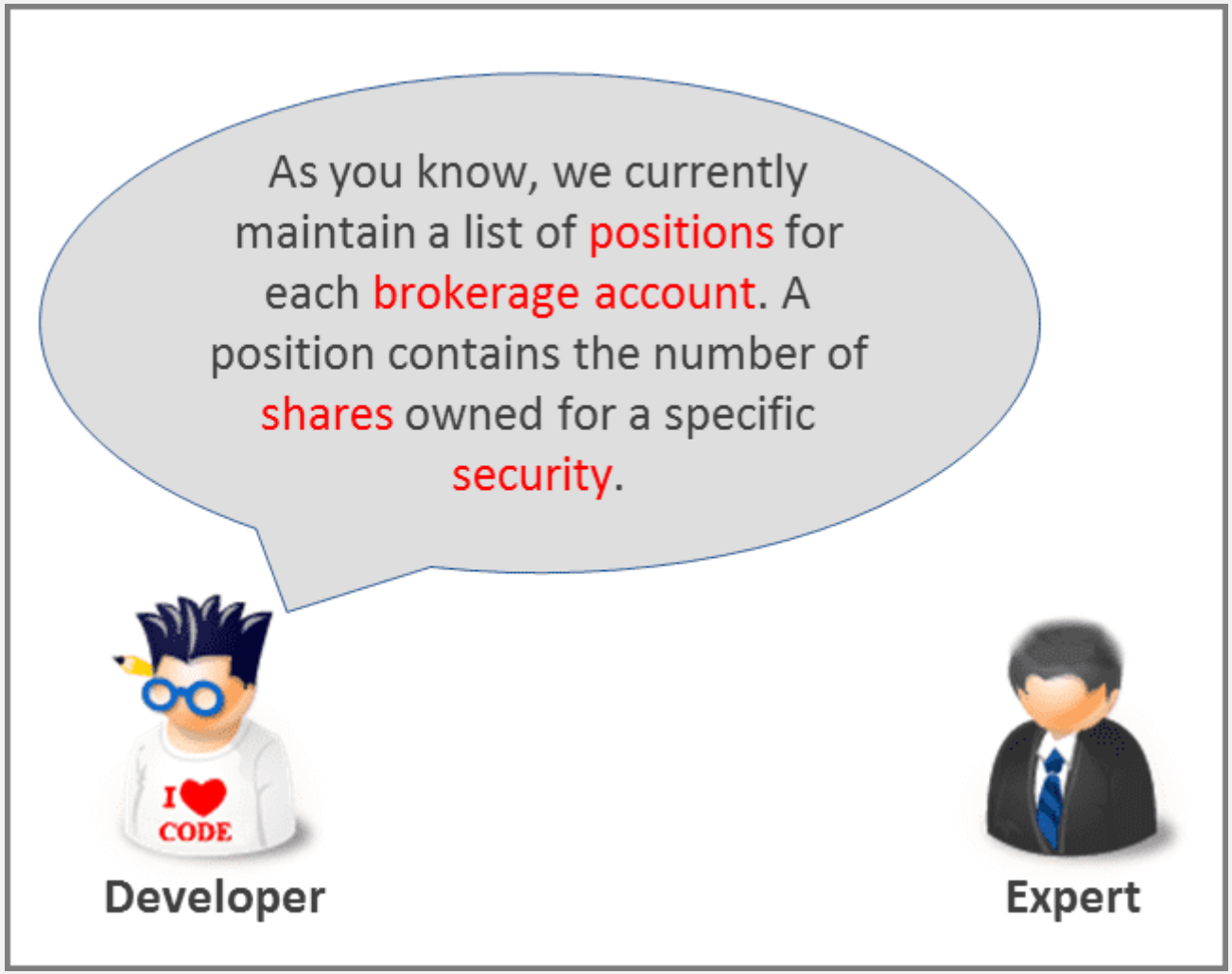
InformAtelier

Giuramenti (Valori)

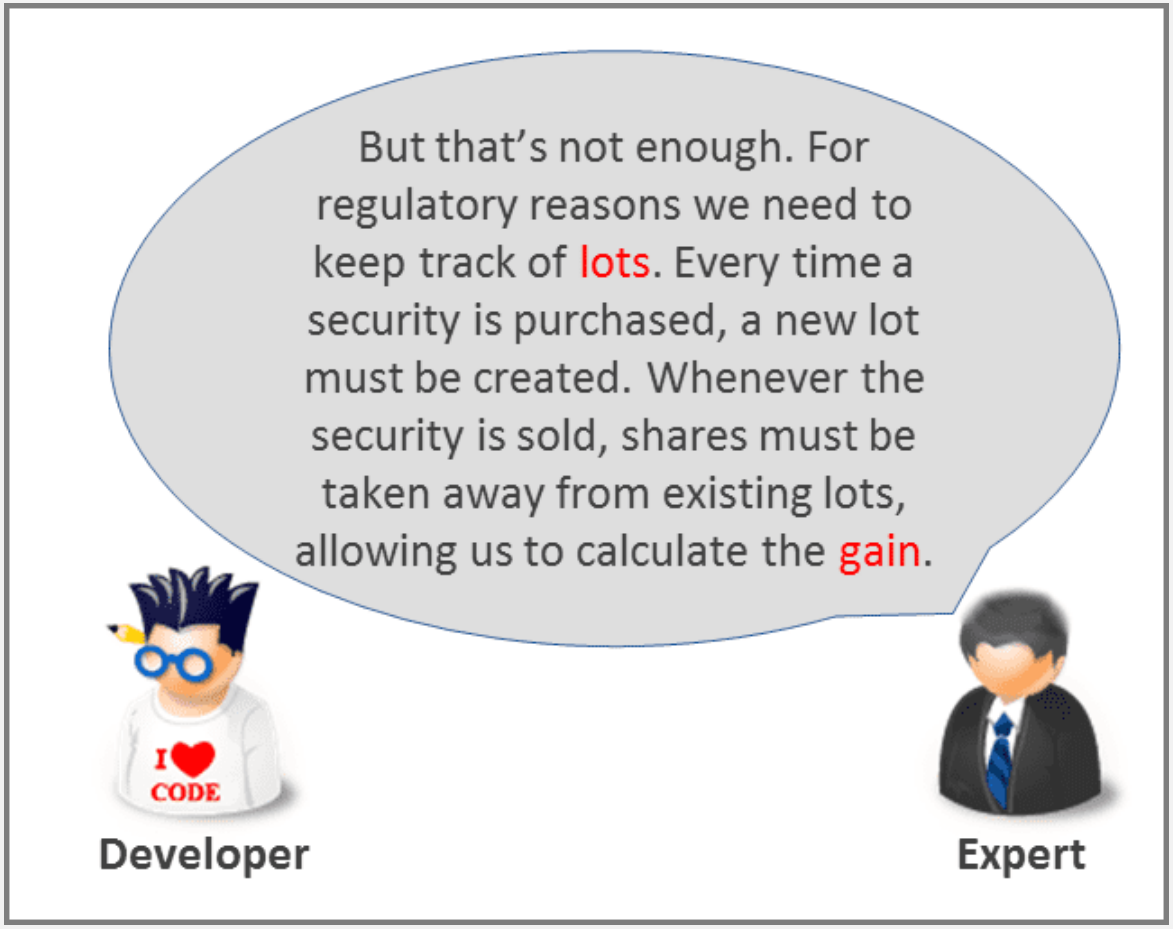
- Migliorare
- Condividere

InformAtelier

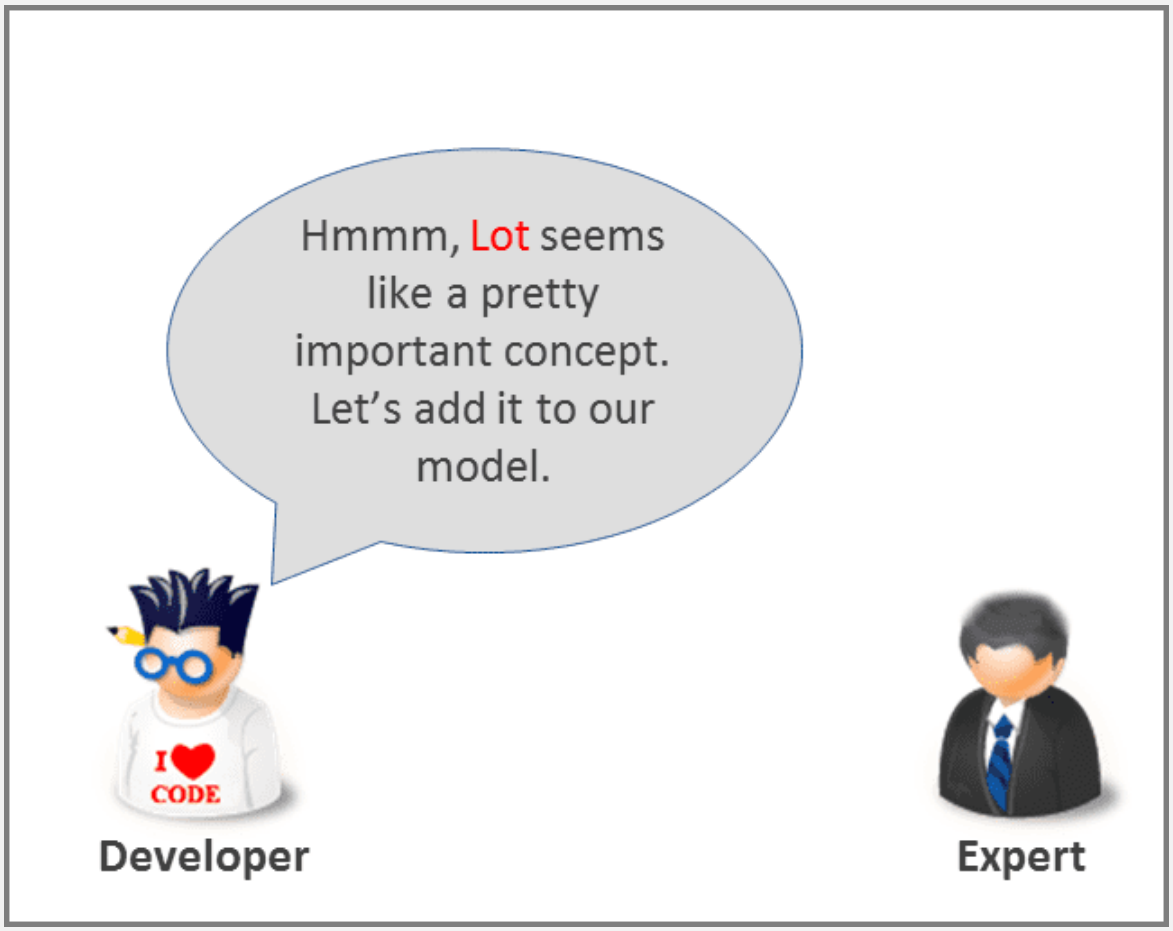
Come emerge un UL



https://tigosoftware.com/what-ubiquitous-language-examples

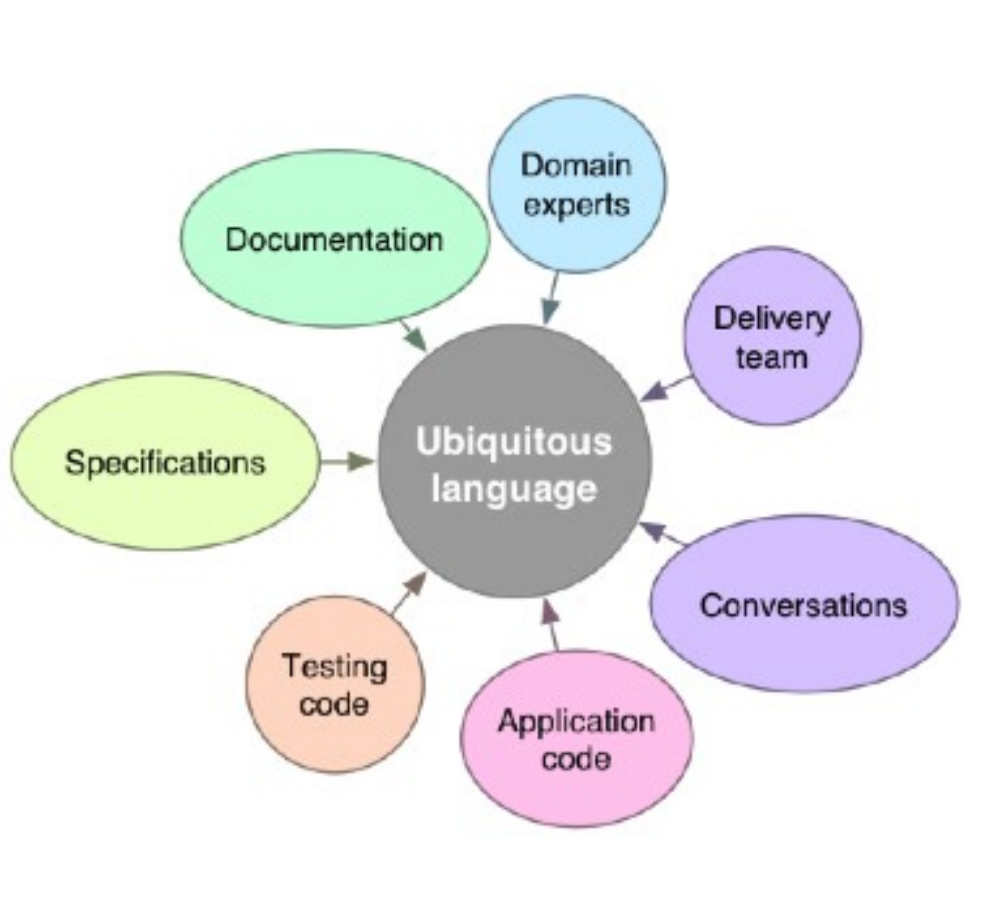


https://tigosoftware.com/what-ubiquitous-language-examples

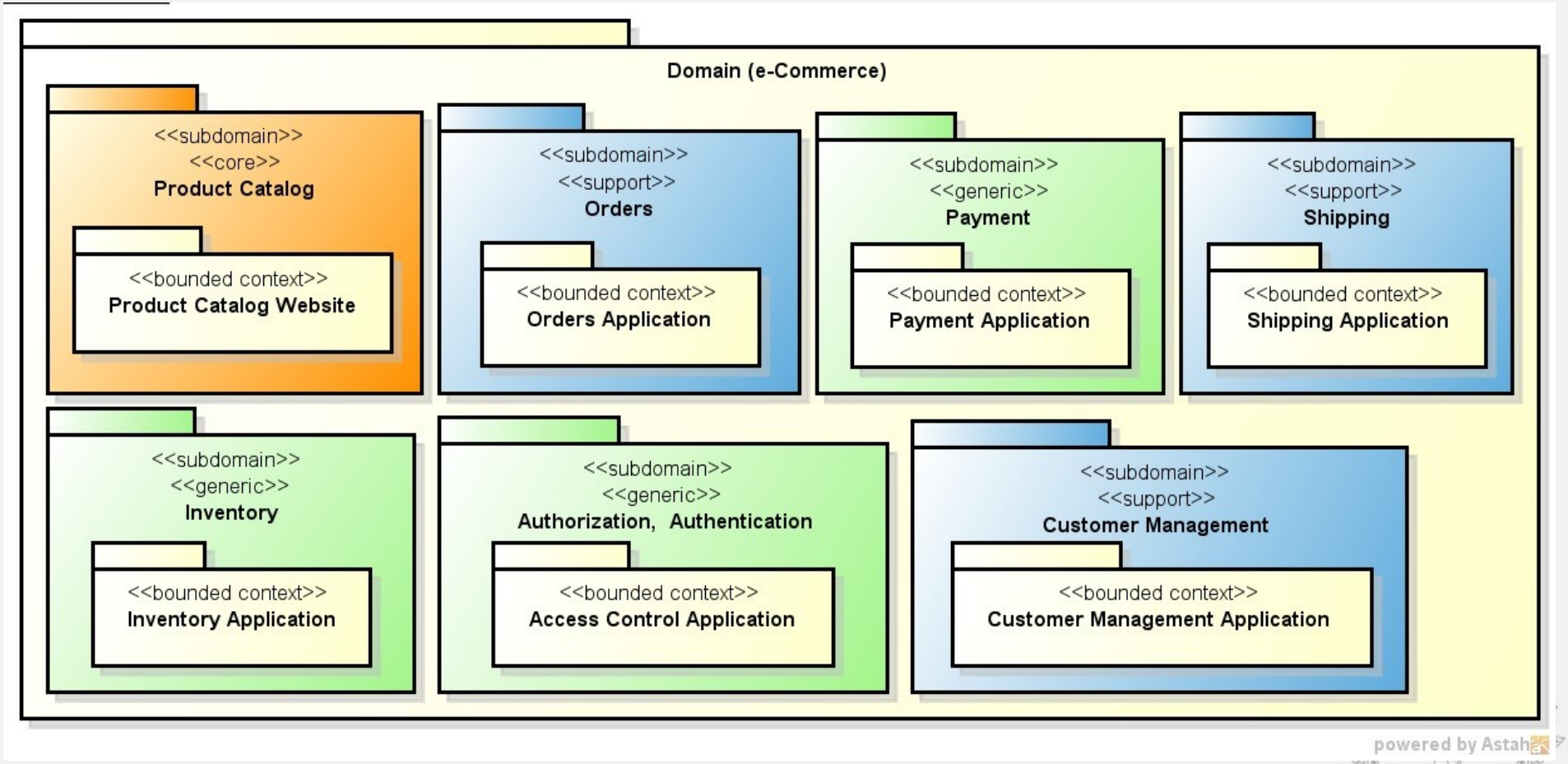


https://tigosoftware.com/what-ubiquitous-language-examples

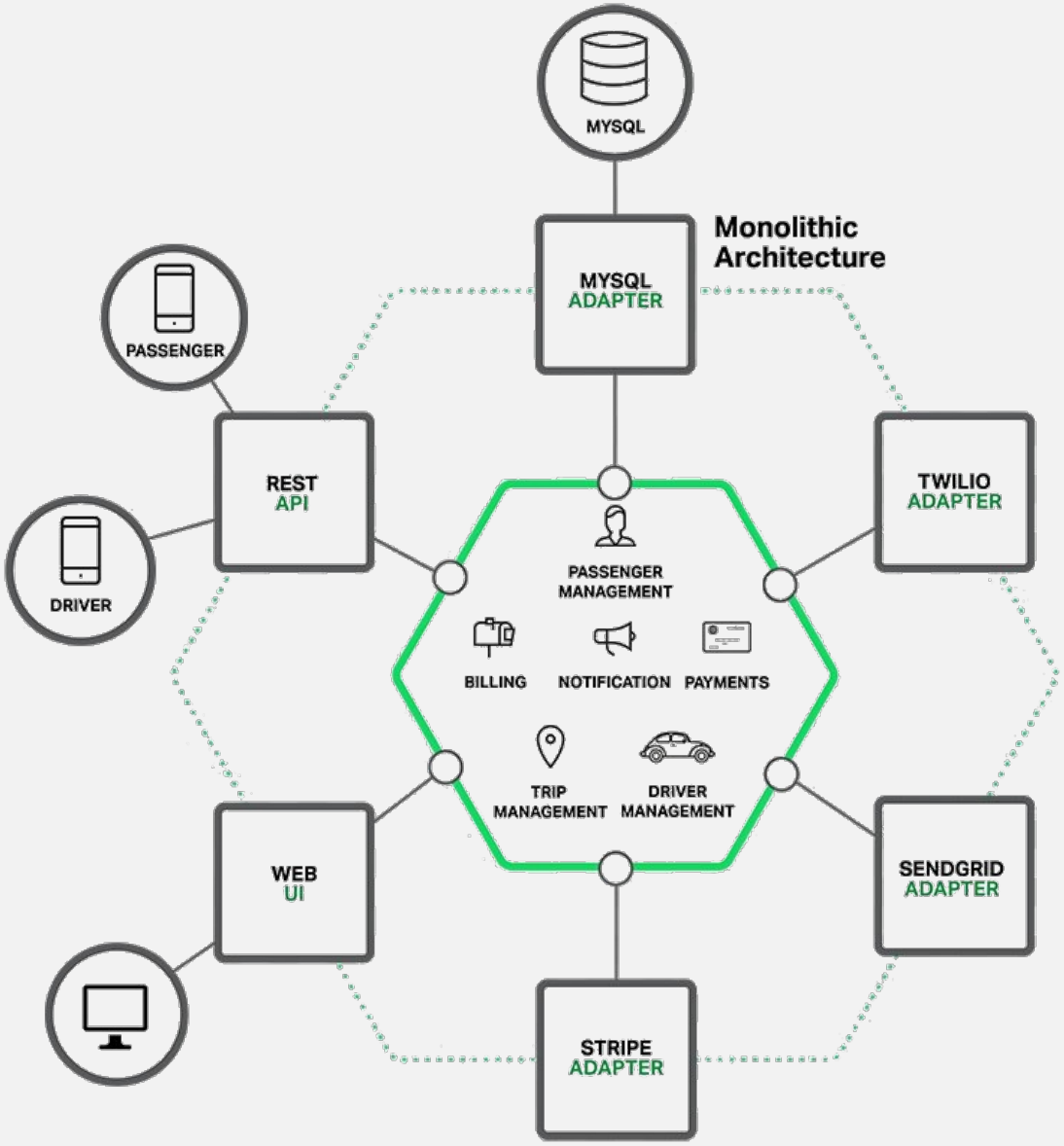
Ubiquitous



Sub-domains e.g.

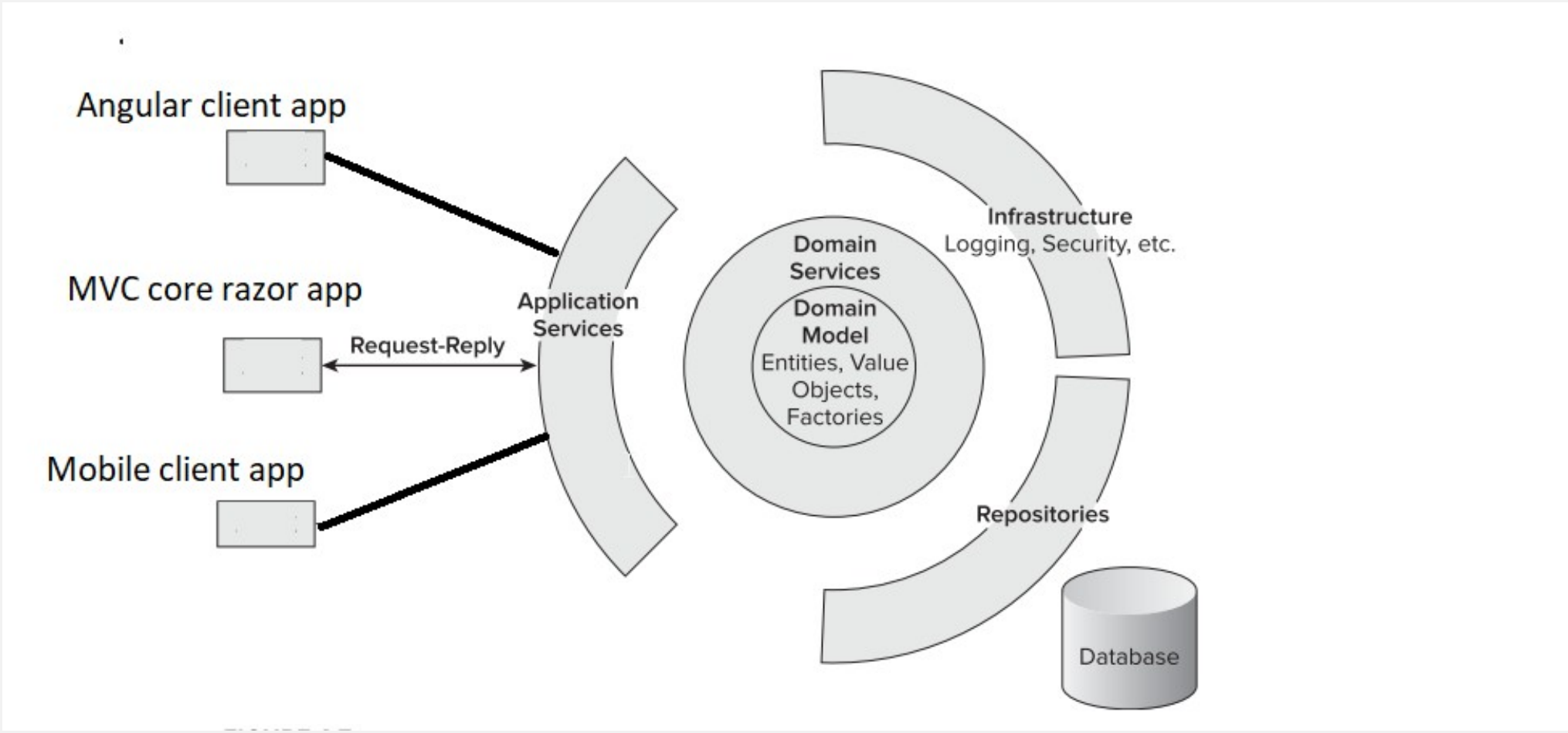


Domain Adapters e.g.

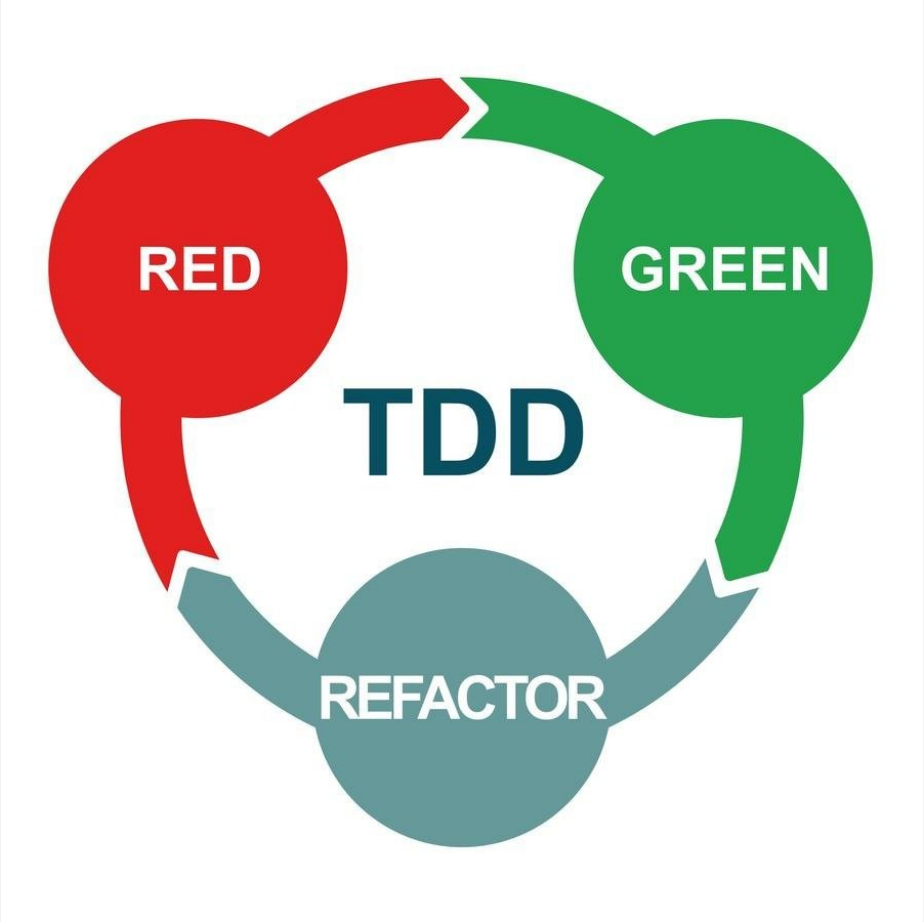


InformAtelier

Domain Services & Model e.g.



Ciclo TDD



3 Regole TDD

1) Non scrivere alcun codice di produzione finché non hai scritto prima un test unitario che fallisca.

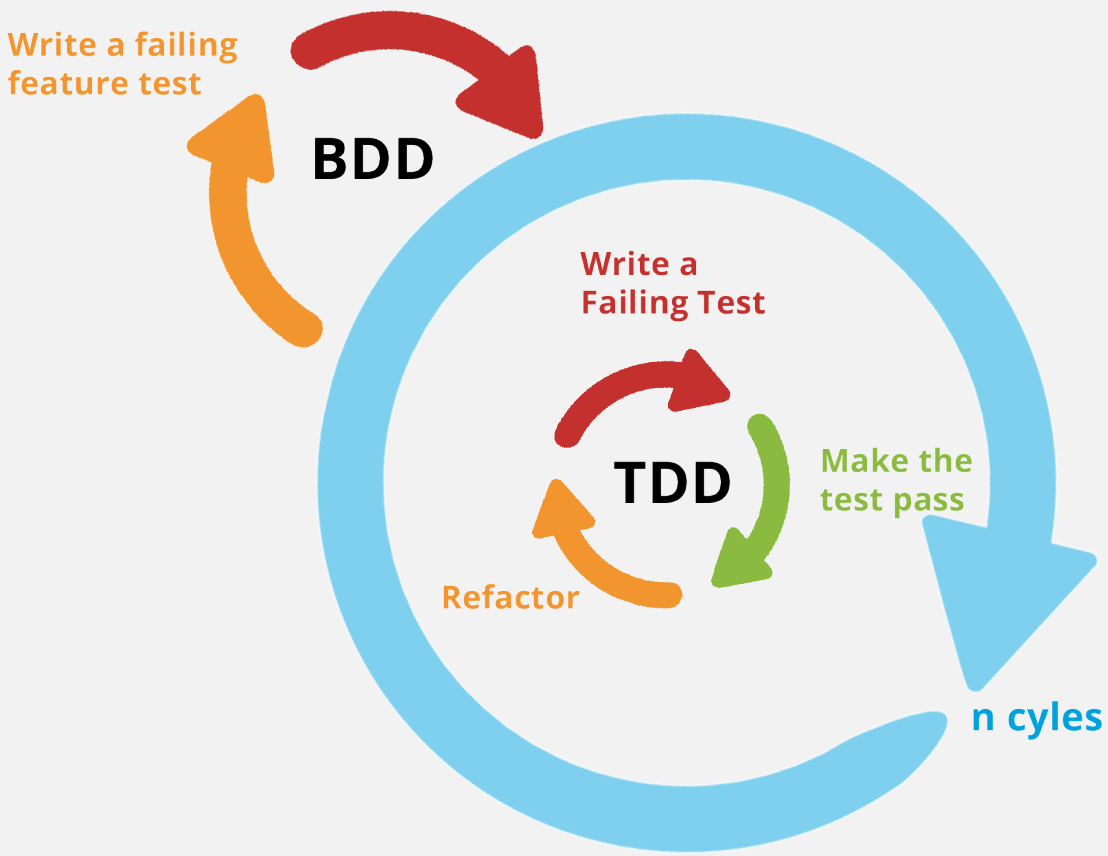
2) Non puoi scrivere più di un test che minimamente fallisca .

3) Non puoi scrivere codice di produzione più di quanto sia sufficiente a far passare il test che attualmente fallisce.

Behavior-Driven Development (BDD)

- Behavior-driven development è un estensione del test-driven development.
- Aggiunge un DSLs per convertire natural language statements strutturati in test eseguibili.
- Il risultato è una relazione più stretta tra criteri di accettazione di una data funzione e i test usati a validare la funzionalità.

Ciclo BDD



InformAtelier

User Story - Scenaries

Title: Items back to inventory.

As a shop woner,

I want to add items back to inventory when they are returned or exchanged,

so that I can track inventory.

Scenario 1: Items returned for refund should be added to inventory.

Given that a customer previously bought a black sweater from me

and I have three black sweaters in inventory,

when they return the black sweater for a refund,

then I should have four black sweaters in inventory.

Gherkin & Cucumber

- Cucumber è un software di test open-source che supporta il BBD.

https://cucumber.io/tools/cucumber-open/

- Gherkin è il linguaggio che gli sviluppatori usano per definire i test in Cucumber.

Esempio Gherkin

Feature: Google Searching

As a web surfer, I want to search Google, so that I can learn new things.

Scenario: Simple Google search

Given a web browser is on the Google page

When the search phrase "panda" is entered

Then results for "panda" are shown

And the related results include "Panda Express"

But the related results do not include "pandemonium"

Definizione singolo Step

```
Scenario: Some cukes
  Given I have 48 cukes in my belly
```

```
package com.example;
import io.cucumber.java.en.Given;

public class StepDefinitions {
    @Given("I have {int} cukes in my belly")
    public void i_have_n_cukes_in_my_belly(int cukes) {
        System.out.format("Cukes: %n\n", cukes);
    }
}
```

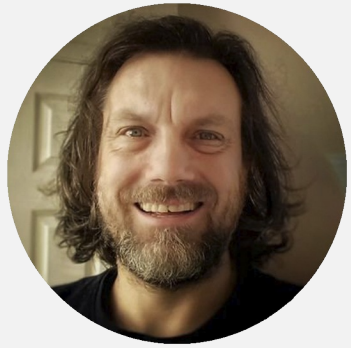
End to end functional tests that can run in milliseconds | Nat Pryce | CukenFest London 2017

- Hanno sfruttato l'architettura "Porte e adattatori" della nostra applicazione per scrivere i test in un modo diverso.
  - Invece di lavorare "outside-in", iniziando guidando la GUI, ora scrivono test funzionali che esercitano il modello di dominio in isolamento.
  - Poi possono eseguire gli stessi test direttamente sul modello di dominio, sulle interfacce di servizio e tramite l'interfaccia utente nei test di sistema end-to-end.
- https://www.youtube.com/watch?v=Fk4rCn4YLLU

Domain Driven Tests

- Scrivere test di comportamento di uno o più bounded context con i soli termini di Ubi Lang.
  - Comprensibile da tutti gli addetti ai lavori.
  - Indipendente dall’adapter.

Pesticide
di Uberto Barbini



“Pesticide is a library to describe our requirements as stories composed by a list of interactions between **actors** (domain personas) and one or more **interpreters** of our system.”

“The idea is to have a single interface (the interpreter) for two or more representations (**protocols**) of our application — for example, DomainOnly and Http. “

https://github.com/uberto/pesticide

https://www.youtube.com/watch?v=BP2ArQ2BCRc&t=1454s

Il processo di sviluppo con Pesticide

1) Scrivere il test DDT per HTTP
2) Implementare l’adapter HTTP
3) Definire il modello, in funzione dell’adapter
4) Implementare l’adapter in memoria
5) Completare le (minime) differenze tra memoria e HTTP

Benefici Pesticide

Eseguendo lo stesso test utilizzando diverse implementazioni dell'interprete otteniamo questi vantaggi:

1) Ci si assicura che la funzionalità funzioni sia end-to-end che nel dominio in memoria.
2) Si documenta la nostra funzionalità utilizzando una lingua vicina all'azienda.
3) Rimozione dell'interfaccia utente o dettagli tecnici dai test.
4) Ci si assicura che non ci siano logiche di business nel livello di infrastruttura e viceversa.

Entry Level

- No Framework oltre a JUnit
- Solo disciplina
  - Scrivere scenari
  - Un caso d’uso alla volta
  - Solo termini di dominio

InformAtelier

Ingredienti Kotlin per la leggibilità nei termini

- Typealias
- Inline classes
- Internal DSL

InformAtelier

Typealias

```
typealias IBAN = String
typealias MoneyAmmount = BigDecimal
typealias Pass = String

typealias UserCode = String
typealias Password = String
typealias Credentials = Pair<UserCode,Password>
```

Inline classes

```
@JvmInline
value class IbanCharge(val ibanValue:IBAN) {}

@JvmInline
value class IbanCredit(val ibanValue:IBAN) {}
```

Scenario

```
@Test
fun test_money_transfer_scenario() {

    // given
    val pass : Pass =
        Steps.` user authenticates with #credentials (user code and password)`(Credentials("userCode","passWord"))
    val user : User =
        Steps.` user asks authorizations giving the pass`(pass)
    val ibanCharge : IbanCharge =
        Steps.` #user inserts charge IBAN IT123 #ibanCharge`(user, "IT...")
    val ibanCredit: IbanCredit =
        Steps.` #user inserts credit IBAN ES123 #ibanCredit`(user, "ES...")
    val transferableAmount: MoneyAmmount =
        Steps.` #user inserts #money 15 euro`(user, "15,00")

    // when
    val moneyTransaction : Result<MoneyAmmount> =
        Steps.` #user has confirmed money transfer for #Amount 15€ from #ibanCharge to #ibanCredit`(
            transferableAmount,
            ibanCharge,
            ibanCredit
        )

    // then
    assertTrue(moneyTransaction.isSuccess)
    assertEquals(BigDecimal(15),moneyTransaction.getOrNull())
}
```

Domain Specific Language (as type safe builders)

```
val fac = socketFactory {
        keyManager {
            open("certsandstores/clientkeystore") withPass "123456" beingA "jks"
        }
        trustManager {
            open("certsandstores/myTruststore") withPass "123456" beingA "jks"
        }
        sockets {
            cipherSuites =
            listOf("TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256",
                    "TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384",
                    "TLS_DHE_RSA_WITH_AES_128_CBC_SHA",
                    "TLS_DHE_RSA_WITH_AES_256_CBC_SHA")
            timeout = 10_000
        }
    }
```

https://kotlinexpertise.com/create-dsl-with-kotlin/