Ente Ospedaliero Cantonale

Test Driven Development

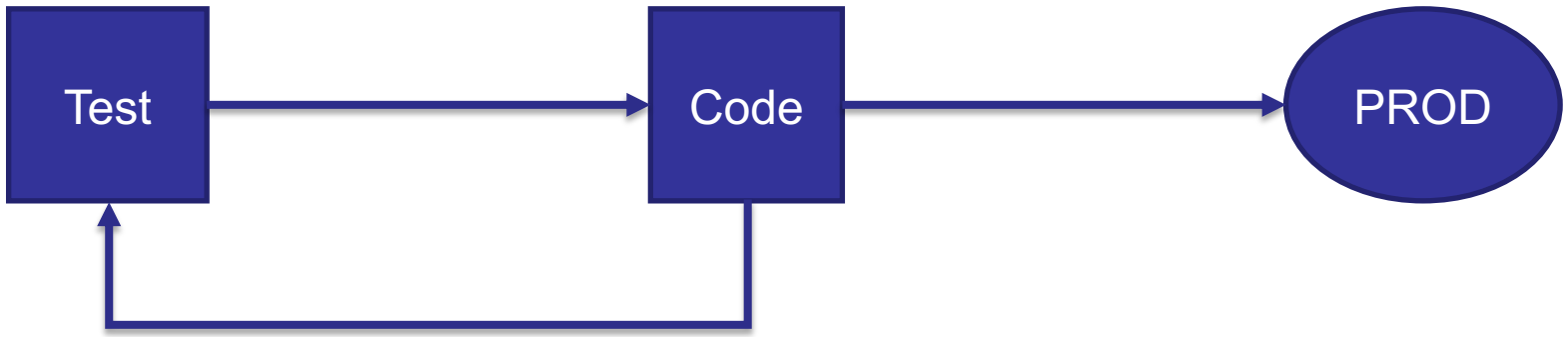
Valerio Lavio

Indice

- Introduzione
- Concetti chiave
- Come funziona
- Tecniche
- Conclusioni

Introduzione

- Sviluppo guidato dai test
- Pattern ciclico
- Produrre valore il più presto possibile
  → «Valuable software sooner»
- Deploy in prod del «Walking Skeleton»

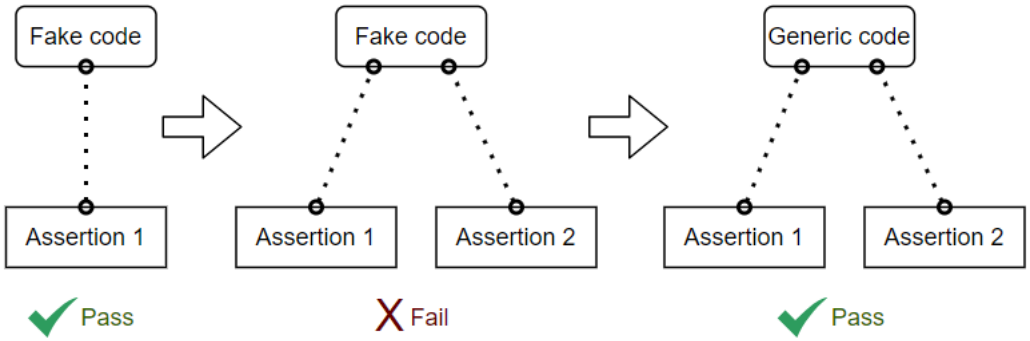


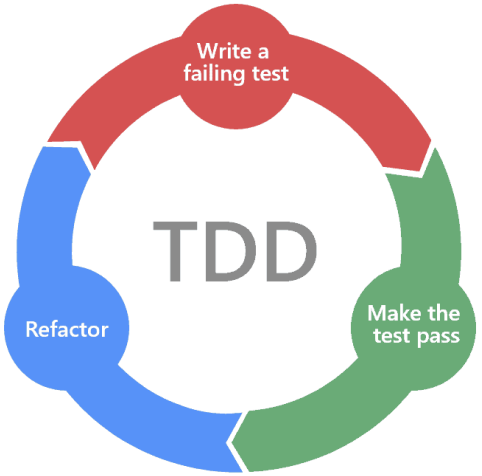
Concetti chiave

- Testare il comportamento e non l’implementazione
- Testiamo quello che porta valore all’utente finale
- Approccio iterativo

Come funziona

- RED – Scriviamo un test che fallisce
- GREEN – codice sufficiente per far passare il test
  - Fake implementation
  - Obvious implementation
  - Triangulation
- REFACTOR – miglioramento del codice





Tecniche

- Refactoring continuo
- Non rendere generico il codice troppo presto
- Cominciare dal test più semplice
- Esplorare i gradi di libertà (uno per volta)
- Nomi «parlanti» per i test
  → MyClassShould → doSomenthing
- Struttura Given When Then
- Transformation Priority Premise per il refactoring

Tabella TPP

#	TRANSFORMATION	STARTING CODE	FINAL CODE
1	{} => **nil**		`return nil`
2	nil => **constant**	`return nil`	`return “1”`
3	constant => **constant+**	`return “1”`	`return “1” + “2”`
4	constant => **scalar**	`return “1” + “2”`	`return argument`
5	statement => **statements**	`return argument`	`return arguments`
6	unconditional => **conditional**	`return arguments`	`if(condition)return arguments`
7	scalar => **array**	`dog`	`[dog, cat]`
8	array => **container**	`[dog, cat]`	`{dog = “DOG”, cat = “CAT”}`
9	statement => **recursion**	`a + b`	`a + recursion`
10	conditional => **loop**	`if(condition)`	`while(condition)`
11	recursion => **tail recursion**	`a + recursion`	`recursion`
12	expression => **function**	`today - birthday`	`CalculateAge()`
13	variable => **mutation**	`day`	`var day = 10; day = 11;`
14	**switch case**		

Conclusioni

- **I test fanno da documentazione**
- Il design emerge dai test
- Ottimo se il problema non é chiaro dall'inizio
- Scomponiamo il problema in piccole parti
- La TPP può aiutarci a scoprire i pattern
  → convergiamo alla soluzione in modo sistematico

Conclusioni

Grazie per l'attenzione

- Fonti:
  - Agile technical practices distilled
    (P.M. Santos, M. Consolaro, A. Di Gioia)
  - https://blog.cleancoder.com/
  - https://it.wikipedia.org/wiki/Test_driven_development
  - https://dmitripavlutin.com/triangulation-test-driven-development/
- Immagini:
  - https://www.dmep.it/hubfs/11_domande_utili_da_fare_per_creare_una_buyer_persona.jpg
  - https://marsner.com/wp-content/uploads/test-driven-development-TDD.png
  - https://res.infoq.com/articles/test-driven-design-java/en/headerimage/test-driven-development-design-technique-logo-big-1557225038423.jpg
  - https://dmitripavlutin.com/3660ee8da9d79232bba059f801a680e3/triangulation-3.svg

Contatti

https://github.com/vale247

https://www.linkedin.com/in/valerio-lavio-38815916a/