

Trova
Prezzi

shoppydoo

KIRIVO

drezzy

cliss
HOBBY

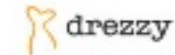


Test Driven Development e Pair Programming

Università degli Studi di Bologna
Dipartimento di Informatica

Bologna - 7 Dicembre 2017

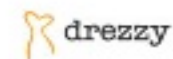
Marco Fracassi
Roberto Grandi



Chi siamo?

In 7Pixel 8 team di sviluppo che fanno XP:

- Nautilus
- **Nimbus -> Marco**
- Antani
- Iguana
- **Analytics & BI -> Roberto**
- Ninja
- Antani
- iOS



Cosa facciamo

Marketplace



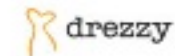
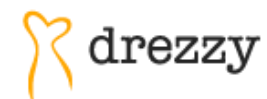
Motore di comparazione prezzi

Italia



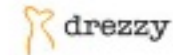
Motore di comparazione prezzi

Spagna



Agenda

- Parte I
 - SW Engineering
 - TDD: perché lo facciamo e cos'è
 - Refactoring
 - Simple Design
 - Tecnica del pomodoro
 - Pair Programming
- E se rimane tempo...
 - Strumenti
 - Esempio



Domande?

- Sono ben accette!
- Sentitevi liberi di interrompere :)
- Non siate timidi



Trova
Prezzi

KIRIVO

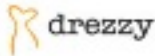
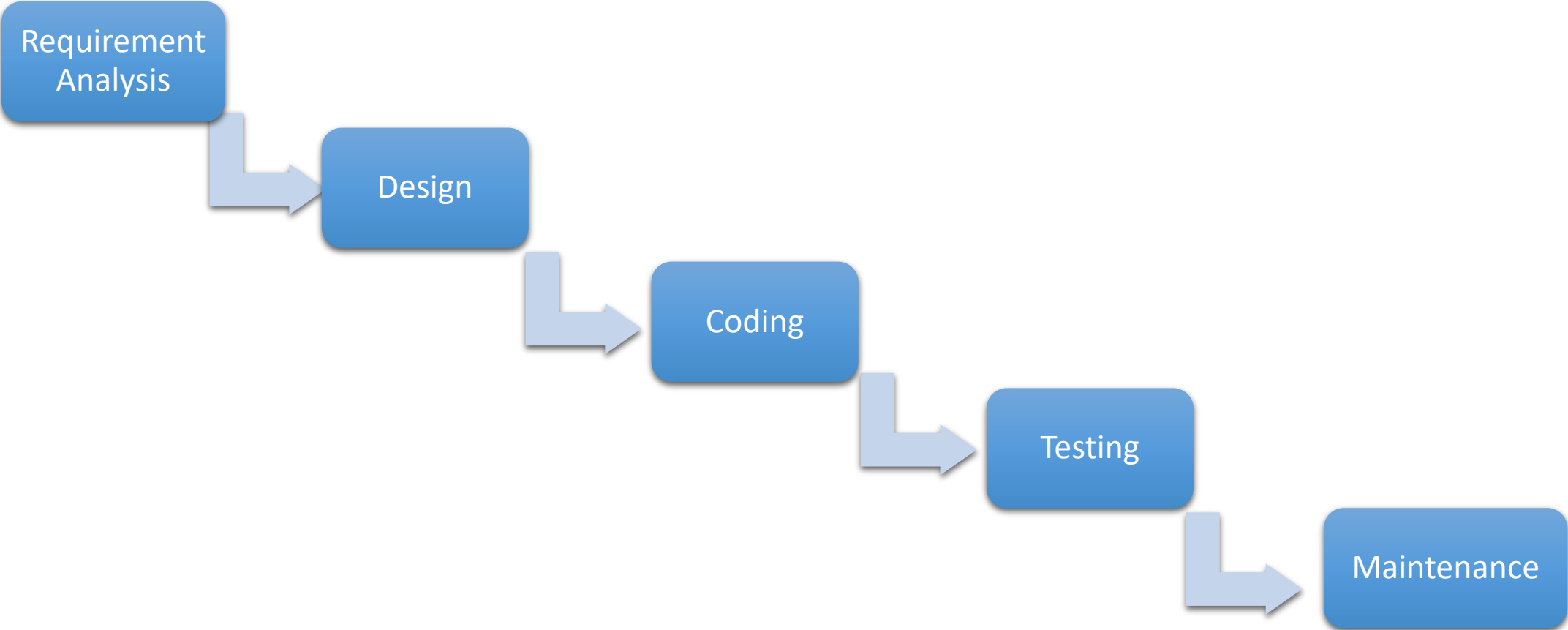
shoppydoo

drezzy

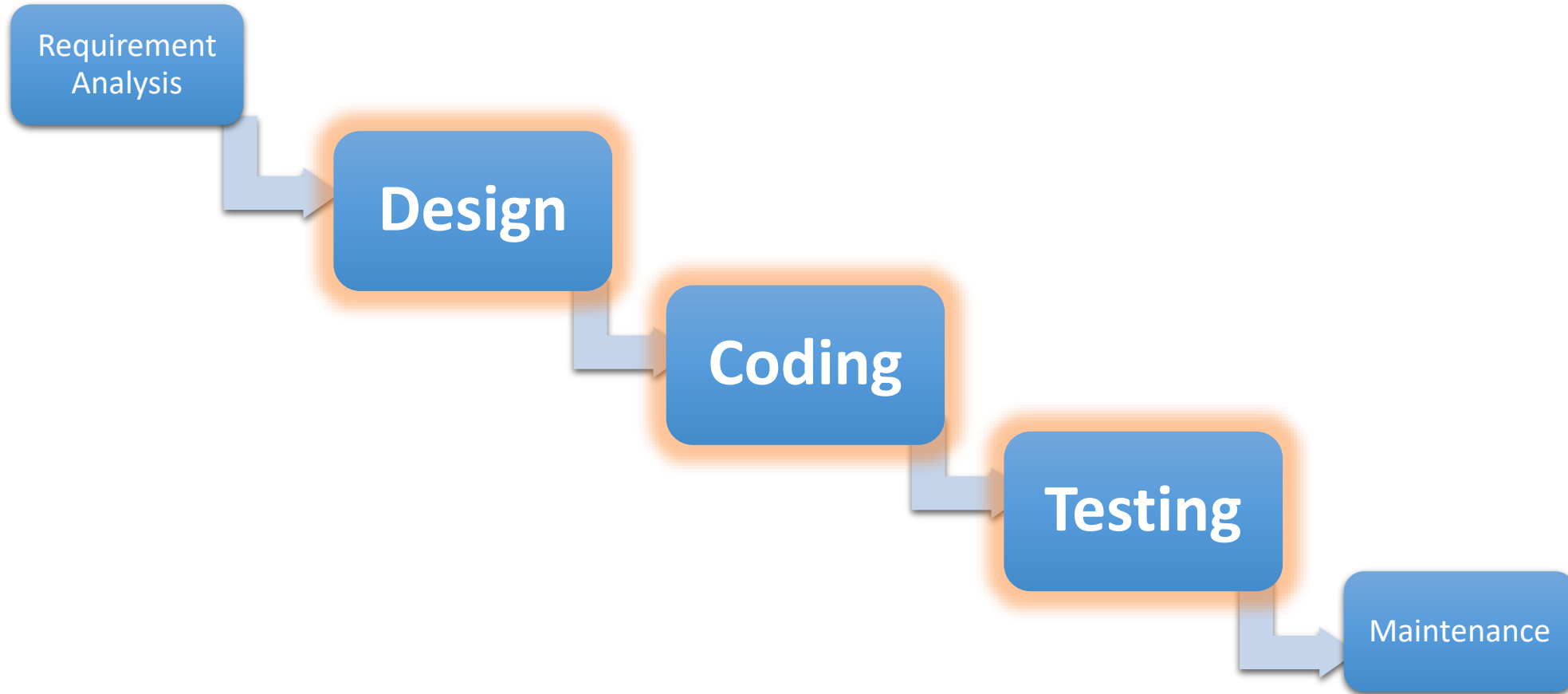
cliss
HOBBY

Encuentra
Precios

SW Engineering – Classical WaterFall Model



SW Engineering – Classical WaterFall Model

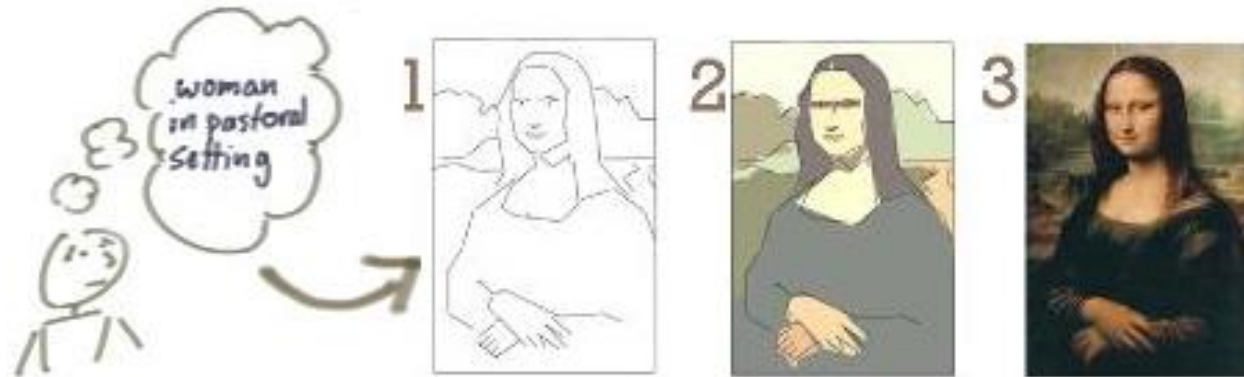


SW Engineering - Incrementale/Iterativo

Incremental



Iterative



Perché fare TDD

- Design evolutivo del codice
- Focus su un unico aspetto utile
- Sicurezza sul codice di produzione



Trova
Prezzi

KIRIVO

shoppydoo

drezzy

cliss
HOBBY

Encuentra
Precios

Perché fare TDD

- Documentazione sempre aggiornata
- Esprimere creatività
- Dare un ritmo allo sviluppo



Trova
Prezzi

KIRIVO

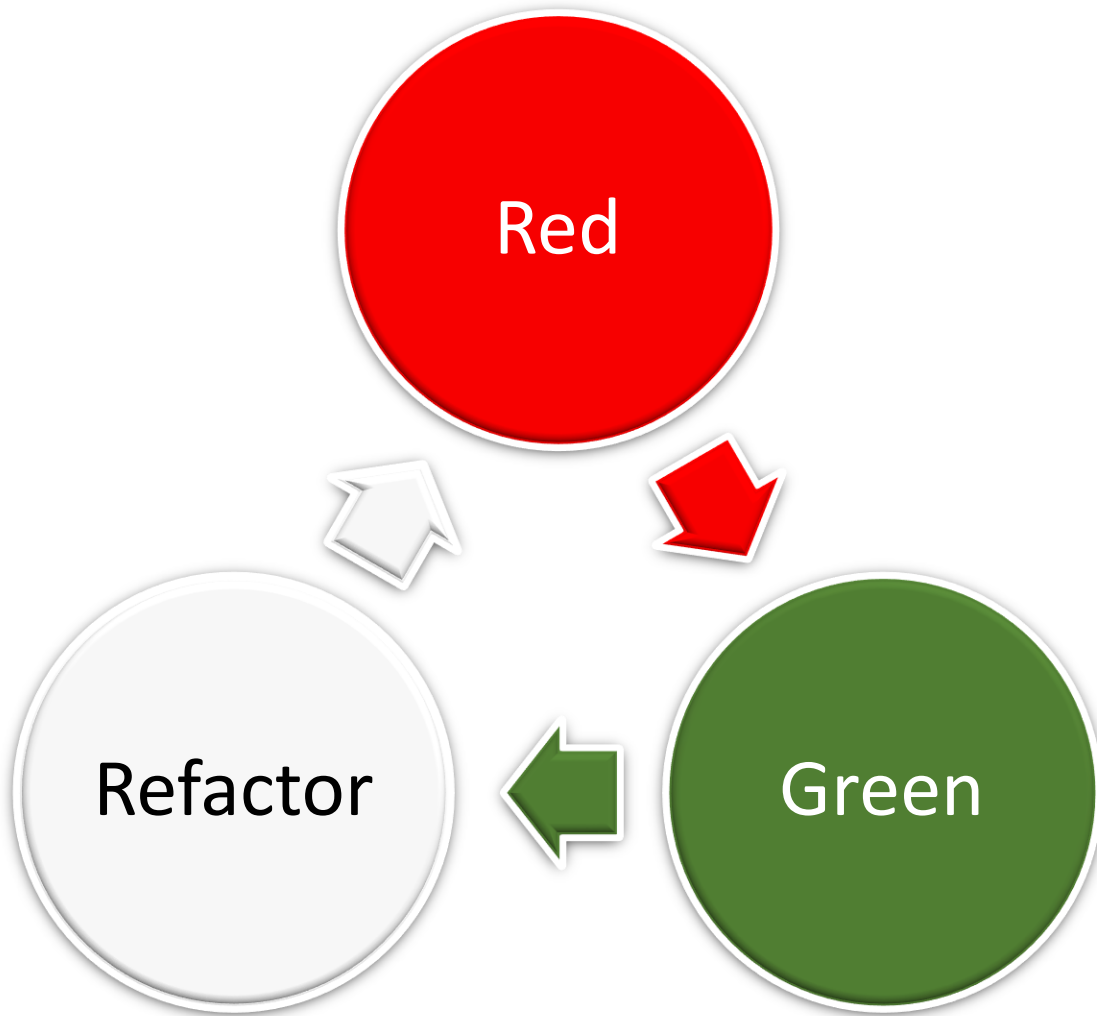
shoppydoo

drezzy

cliss
HOBBY

Encuentra
Precios

Come si fa TDD



Red	Scrivo un test che fallisce
Green	Scrivo il codice che fa passare il test
Refactor	Pulisco e il codice ed il test

Se voglio fare TDD devo rispettare le seguenti regole

- 1. Non scrivere codice di produzione prima di aver scritto un test rosso**
- 2. Non scrivere più di un test che fallisce, un errore di compilazione è come un fallimento**
- 3. Non scrivere più codice di produzione di quello necessario per far passare il test correntemente rosso**

Le 3A dei test

- **Arrange** -> Costruisco il contesto del test
- **Act** -> Eseguo il codice che voglio testare
- **Assert** -> Verifico quello che ottengo

Esempio:

Voglio testare le funzionalità base di una classe conto bancario (Java + JUnit)

- *BankAccount account = new BankAccount().WithBalance(10);*
- *account.addMoney(10);*
- *assertEquals(20, account.getBalance());*

- Con quale test partire?
 - **Happy Path:** il caso più semplice/favorevole che porta maggior valore
 - ..
 - ..
 - **Boundaries Condition:** i casi limite

Come scrivo i test - Esempio Boundaries

- *BankAccount* account = *new BankAccount().WithBalance(10);*
- *account.addMoney(0);*
- ***assertThat("Balance must be 0", account.getBalance(), is(0));***

- *account.addMoney(null);*
- ***assertThrown(account.getBalance());***

"...is a disciplined technique for restructuring an existing body of code, altering its internal structure without changing its external behavior"

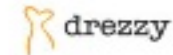
Ovvero: l'esecuzione di azioni di ristrutturazione del codice volte a **migliorarne la struttura interna**, abbassandone la complessità **senza modificarne le funzionalità**

Cos'è il Refactoring

- Si basa su una serie di **piccole trasformazioni** che **presevano il comportamento**; ogni trasformazione (refactoring) è limitata, ma una sequenza di queste può portare ad una significativa ristrutturazione.
- Poichè ogni trasformazione è limitata è poco verosimile abbia un impatto negativo ed elevato: è controllabile.
- Dopo ogni refactoring, grazie ai test soprattutto, il software mantiene inalterate le funzionalità per definizione (se abbiamo fatto un refactoring)
- Si reduce, grazie ai test, la possibilità di introdurre errori nel software durante le ristrutturazioni del codice

Benefici del Refactoring

- Non far degradare il design del SW
- Eliminare codice duplicato (!!!)
- Rendere il codice più facile da comprendere (a me e ai miei colleghi)
- Trovare Bug (se un test fallisce sono già a metà dell'opera)
- Velocizzare lo sviluppo



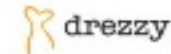
Code Smell

Da dove parto a fare refactoring? Dai Code Smell!

"Insieme di caratteristiche che il codice sorgente può avere e che sono generalmente riconosciute come probabili indicazioni di un difetto di programmazione"

Non sono bug! (il software funziona correttamente)

Sono "debolezze" del codice che ne riducono la qualità

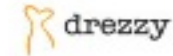


Smell

- *Elenco dei principali smells:*
 - *Codice duplicato ("Don't Repeat Yourself" - DRY)*
 - *Metodo troppo lungo*
 - *Commenti*
 - *Classe troppo grande*
 - *Lista di parametri troppo lunga*
 - *Feature envy*
 - *Switch*
 - *Dead code*
 - *Generalità speculativa ("You Aren't Gonna Need It" o YAGNI)*
 - *Ecc...*

Mosse

- *Extract method*
- *Extract class*
- *Rename*
- *Replace temp with query*
- *Move method*
- *Decompose conditionals*
- *Introduce Parameter Object*
- *Preserve Whole Object*
- *Substitute Algorithm*



Simple design

- **Tutti i test sono verdi**
- **Non c'è duplicazione**
- **L'intento del programmatore è espresso chiaramente**
- **Il numero di classi e metodi è minimo**



Trova
Prezzi

KIRIVO

shoppydoo

drezzy

cliss
HOBBY

Encuentra
Precios

Cos'è la duplicazione

- **Codice duplicato**
- **Logica duplicata**
- **Algoritmi interscambiabili**



Trova
Prezzi

KIRIVO

shoppydoo

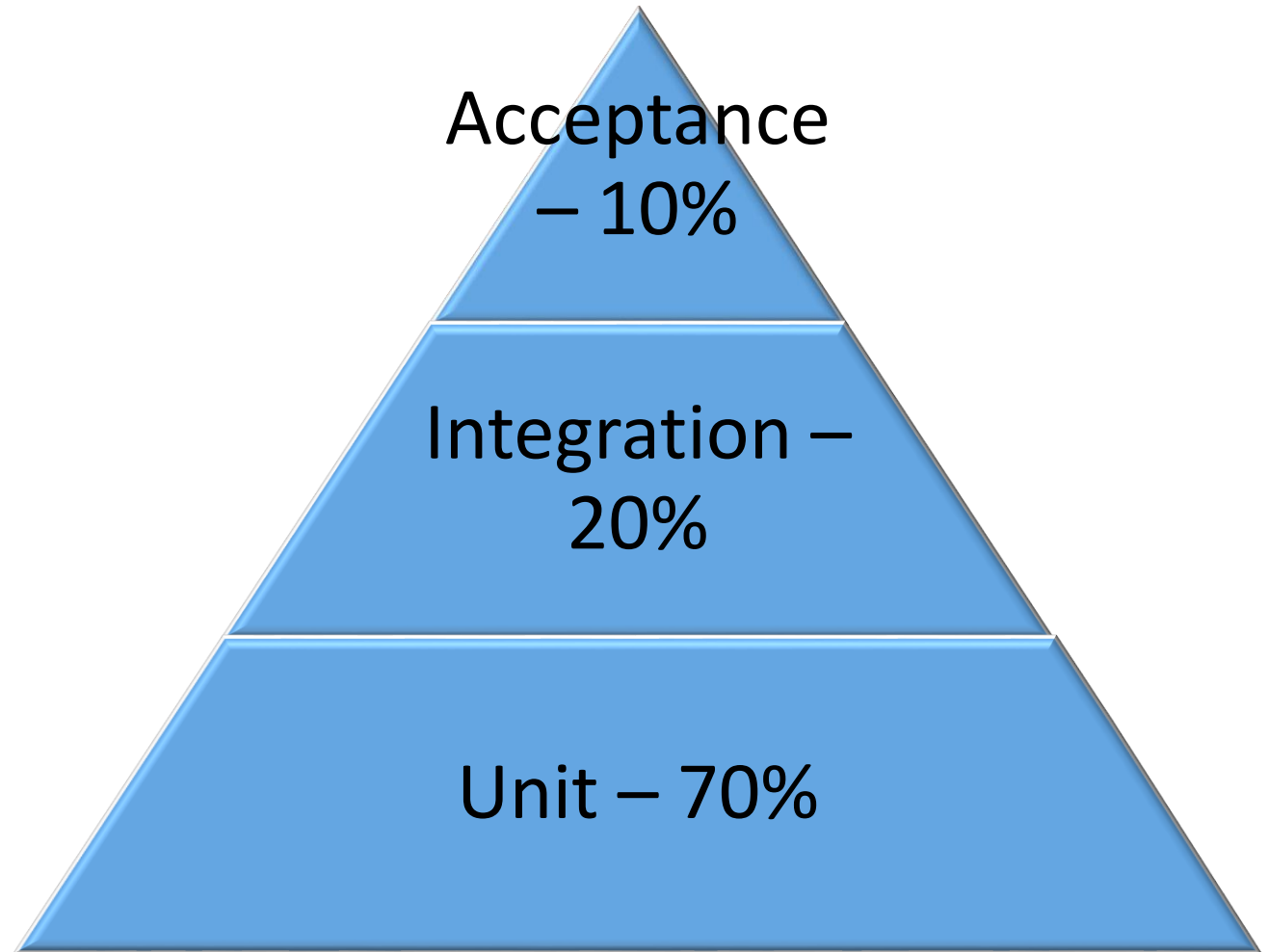
drezzy

cliss
HOBBY

Encuentra
Precios

Test

- Test.... sono tutti uguali?
 - Unit Test
 - Integration Test
 - Acceptance Test
 -
 - Regression Test
 - Smoke Test



Come dovrebbe essere un test?

- **Veloce**
 - Li dobbiamo lanciare centinaia di volte! Domanda: Tutti?
- **Ripetibile**
- **Indipendente** (dagli altri test)
 - Altrimenti si diventa pazzi!
- **Isolato** (dal codice di produzione)
 - Separazione progetti
 - Anche se molti tool non partono così (test e src vicini di casa)!
 - A volte promuoviamo classi di test a classi applicative

Come dovrebbe essere un test?

- **Rifattorizzato**...non è codice di serie B
 - A volte testo le primitive di test
- **Non distruttivo**
 - Preserva le condizioni iniziali (No Drop table)
- **Self verified**
 - E' chiaro perché fallisce? Se fallisce devo capire al volo come mai
 - Piuttosto metto un messaggio nelle asserzioni
- **Timely**
 - Eseguito in tempo ragionevole



Trova
Prezzi

KIRIVO

shoppydoo

drezzy

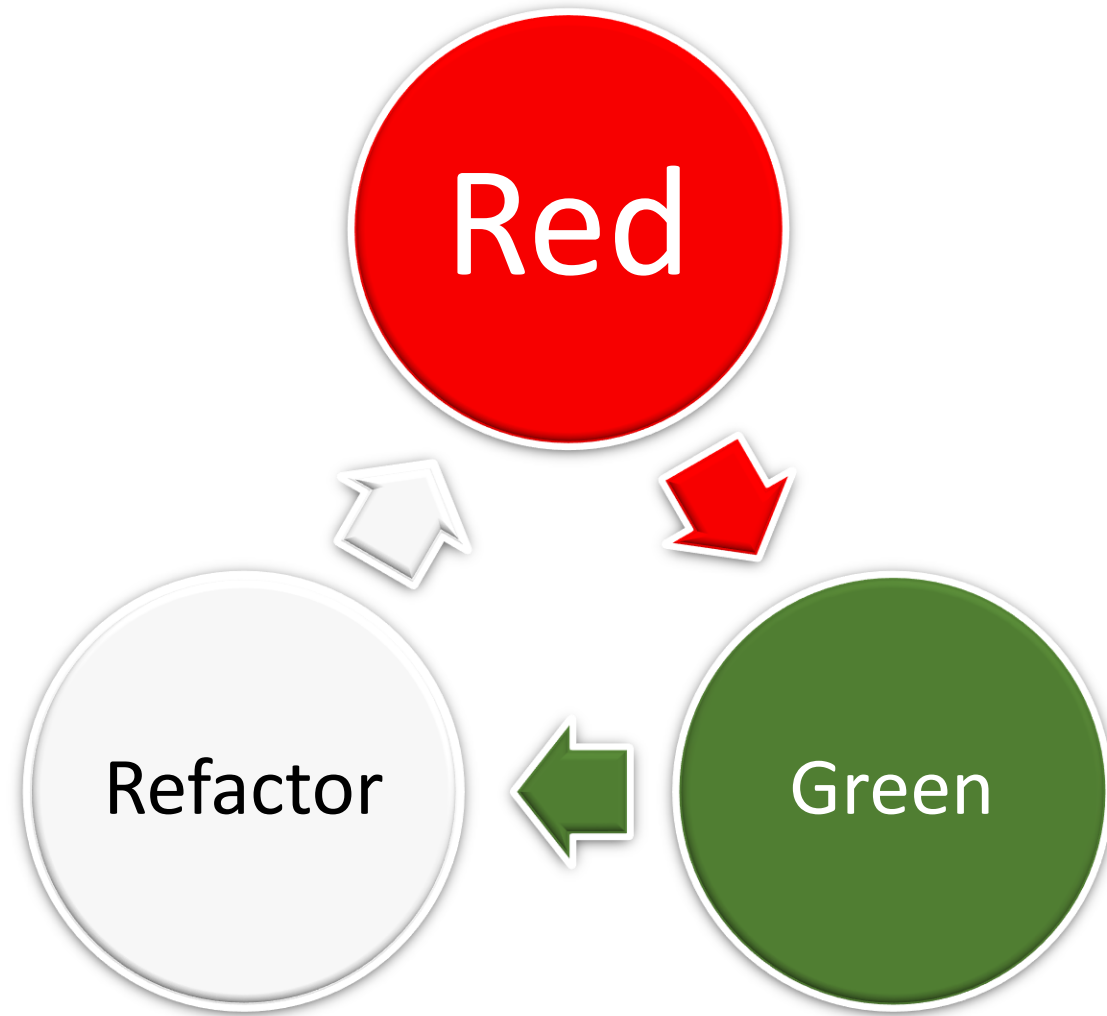
cliss
HOBBY

Encuentra
Precios

Chi testa il test?

- Il QA?
- Il mio compagno di pair?
- Un tester del test?
- Un test automatico del test?

RED verifica il test!



Il ritmo dello sviluppo

- Come possiamo introdurre un il concetto di **ritmo** nella giornata dello sviluppatore?
- Possiamo usare uno strumento che ci permetta anche di valutare il tempo speso?
- Quale unità di misura scegliere?



Trova
Prezzi

KIRIVO

shoppydoo

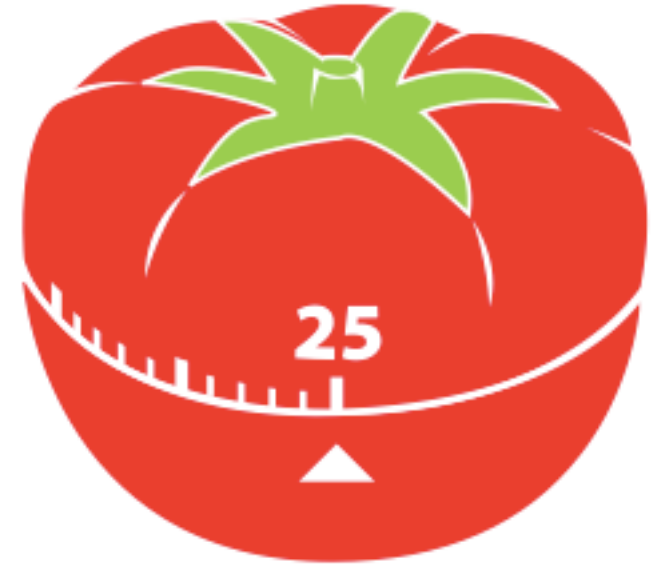
drezzy

cliss
HOBBY

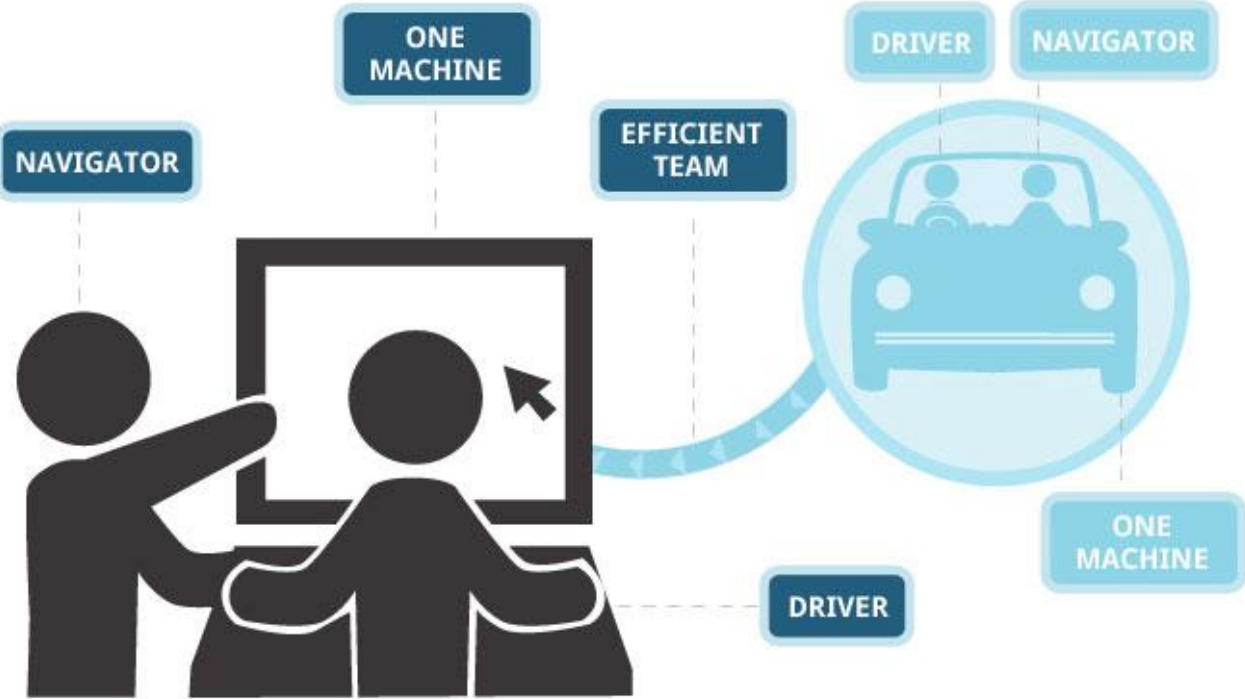
Encuentra
Precios

La tecnica del pomodoro

```
{  
  "name": "tecnica del pomodoro",  
  "youtube": https://www.youtube.com/watch?v=CT70iCaG0Gs,  
  "website": "http://pomodorotechnique.com/"  
}
```



Pair Programming



Alcune buone regole per un buon rapporto... di coppia

- *Non interrompere*
- *Suggerire («cosa ne pensi?»)*
- *Appuntarsi i problemi emersi – checklist*
- *Igiene personale*

Pilota

- *Ha mouse e tastiera*
- *Scrive i test ed il codice*
- *Ha una visione di breve periodo*

Navigatore

- *Ha una todo-list*
- *Appunta i test mancanti*
- *Appunta refactoring possibili*
- *Ha una visione di lungo periodo*

Anche il miglior pilota ha bisogno di riposo

Anche il miglior navigatore vuole adrenalina

Ovvero: cambiare ogni 2-3 pomi, seguire user story diverse, affrontare le cose che si conoscono meno

ScuolaXP

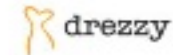
Come diventare Agile Developers in 5 giorni

Rivolta a:

- Studenti
- Professionisti
- Interni Junior

Riferimenti e materiale www.scuolaxp.it

Seguici su [@ScuolaXP](https://twitter.com/ScuolaXP) per i prossimi eventi e seminari



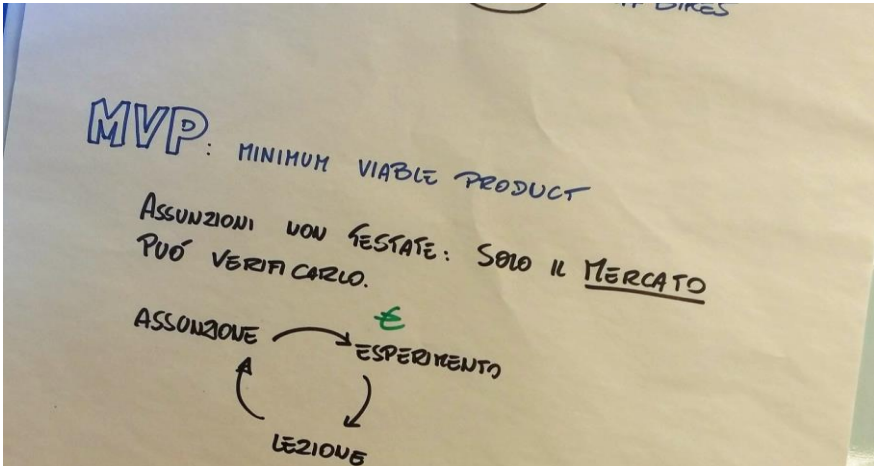


MVP: MINIMUM VIABLE PRODUCT

ASSUNZIONI NON TESTATE: SONO IL MERCATO
PUÒ VERIFICARLO.

ASSUNZIONE → ESPERIMENTO
↳ LEZIONE

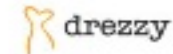
ScuolaXP - Edizione 2016



ScuolaXP - Edizione 2017

Grazie!

- Domande?
- Il vostro Feedback
- Contattateci: qualsiasi cosa (no cash \$\$\$), tesi, stage, cv
- marco.fracassi@trovaprezzi.it - @fracassi_marco
- roberto.grandi@trovaprezzi.it - @grandirob



Gli strumenti del mestiere

- *Un sistema di versionamento: **Git** o **Svn***
- *Un ambiente di sviluppo: **Eclipse** o **NetBeans***
- *Un Test Framework: **JUnit***

Obiettivo

- *Vogliamo costruire una calcolatrice in grado di sommare i numeri presenti una stringa di testo*
- *La stringa può avere più numeri separati dalla virgola*
- *Se non ho numeri voglio ottenere 0*
- *....ho dimenticato qualcosa?*

Esempio - String Calculator 1

```
@Test
public void parseOneNumber()
{
    Calculator calculator = new Calculator();

    int result = calculator.parse("1");

    assertThat(result, is(1));
}
```

```
@Test
public void parseTwoNumbers()
{
    Calculator calculator = new Calculator();

    int result = calculator.parse("1,2");

    assertThat(result, is(3));
}
```

Esempio - String Calculator 2

```
@Test
public void parseManyNumbers()
{
    Calculator calculator = new Calculator();

    int result = calculator.parse("1,2,3,4");

    assertThat(result, is(10));
}
```

```
@Test
public void parseEmptyString()
{
    Calculator calculator = new Calculator();

    int result = calculator.parse("");

    assertThat(result, is(0));
}
```

Esempio - String Calculator 3

```
@Test
public void parseNotANumberWithNumber()
{
    Calculator calculator = new Calculator();

    int result = calculator.parse("1,ABC");

    assertThat(result, is(1));
}
```

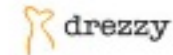
```
@Test
public void parseNotANumber()
{
    Calculator calculator = new Calculator();

    int result = calculator.parse("A,ABC");

    assertThat(result, is(0));
}
```

Riferimenti

- Martin Fowler et al. (1999), **Refactoring: Improving the Design of Existing Code**, Addison-Wesley
- Robert C. Martin (2008), **Clean Code: A Handbook of Agile Software Craftsmanship**, Prentice Hall
- Kent Beck (2000), **Extreme Programming Explained: Embrace Change, 2nd Edition**, Addison-Wesley



Riferimenti

marco.fracassi@trovaprezzi.it

roberto.grandi@trovaprezzi.it

cv@trovaprezzi.it



Trova
Prezzi

KIRIVO

shoppydoo

drezzy

cliss
HOBBY

Encuentra
Precios