



Svigruppo

Monga

Tecniche di lavoro  
Pair Programming  
Codice condiviso  
Refactoring  
TDD  
Velocity

# Sviluppo software in gruppi di lavoro complessi<sup>1</sup>

Mattia Monga

Dip. di Informatica  
Università degli Studi di Milano, Italia  
[mattia.monga@unimi.it](mailto:mattia.monga@unimi.it)

Anno accademico 2022/23, II semestre

<sup>1</sup> 2023 M. Monga. Creative Commons Attribuzione — Condividi allo stesso modo 4.0 Internazionale.  
<http://creativecommons.org/licenses/by-sa/4.0/deed.it>



Svigruppo

Monga

Tecniche di lavoro  
Pair Programming  
Codice condiviso  
Refactoring  
TDD  
Velocity

## Lezione VI: Tecniche di lavoro agile



Svigruppo

Monga

Tecniche di lavoro

Pair Programming

Codice condiviso

Refactoring

TDD

Velocity

Ogni metodologia agile ne ha di specifiche, le più famose sono:

- *Pair programming*
- Codice condiviso (di proprietà collettiva)
- *Refactoring*
- *Test Driven Development* (TDD)
- *Velocity tracking*



Svigruppo

Monga

Tecniche di lavoro

Pair Programming

Codice condiviso

Refactoring

TDD

Velocity

Si programma a coppie, con una sola tastiera.

- Obbliga a rendere espliciti i ragionamenti
- Aiuta a mantenere il focus sull'obiettivo
- Diffonde la conoscenza totale della *codebase* (riducendo anche i rischi in caso di assenza di un collaboratore)

Questa (e TDD) è fra le tecniche maggiormente studiate sperimentalmente: nessuna evidenza che faccia differenza sulla qualità dei prodotti. La produttività, apparentemente dimezzata, rimane simile.



Svigruppo

Monga

Tecniche di lavoro  
Pair Programming  
Codice condiviso  
Refactoring  
TDD  
Velocity

Tutto il *team* è responsabile di **tutto** il codice e può modificarlo a piacimento.

- Un'unica *codebase*
- Si lavora tutti sulla stessa *branch* senza specifici momenti di *merge*
- *Continuous integration* (possibile grazie a TDD)
- Il codice è una forma di comunicazione *broadcast*

La proprietà collettiva **non** è una buona ragione per rinunciare all'*information hiding*



Svigruppo

Monga

Tecniche di lavoro

Pair Programming

Codice condiviso

**Refactoring**

TDD

Velocity

refactoring

Martin Fowler, 2000: *“is a disciplined technique for restructuring an existing body of code, altering its internal structure without changing its external behavior.”*

Sono piccole trasformazioni che non cambiano la semantica del codice, spesso attuabili automaticamente con un *editor* “consapevole” del linguaggio di programmazione.

Fowler mantiene un catalogo: <http://refactoring.com/catalog/>



Svigruppo

Monga

Tecniche di lavoro

Pair Programming

Codice condiviso

**Refactoring**

TDD

Velocity

- Fattorizzazione di codice ripetuto in una funzione/metodo
- Campi attributo in metodi *getter/setter*
- Eliminazione di condizionali, sostituendoli con opportuni collegamenti dinamici (sottoclassi)
- Fattorizzazioni di comportamenti complessi in superclassi (eventualmente astratte)



Svigruppo

Monga

Tecniche di lavoro

Pair Programming

Codice condiviso

Refactoring

TDD

Velocity

Il *test* di unità viene scritto prima dell'unità stessa, servendo come “specifica” (ma senza la **necessaria** generalità!)

- 1 Aggiungi un *test*
- 2 Ripeti tutti i *test* assicurandoti che il nuovo *test* fallisca
- 3 Scrivi il codice dell'unità
- 4 Ripeti i *test* (questa volta dovrebbero passare)
- 5 *Refactoring* mantenendo il superamento dei *test*
- 6 Da capo

Ogni *bug* dovrebbe essere esaminato attentamente e diventare un nuovo caso di *test*



# Supporto al test: testing frameworks



Librerie “xUnit” (JUnit, Kent Beck, 2002)

```
import org.junit.jupiter.api.Test;
import static org.assertj.core.api.Assertions.assertThat;

public class GroceryListTest {
    final int MAX_FOOD = Main.MAX_FOOD;

    @Test
    public void testGroceryList() {
        GroceryList groceryList = new GroceryList(MAX_FOOD);
        assertThat(groceryList.size()).isEqualTo(0);
    }
}
```

Svigruppo

Monga

Tecniche di lavoro

Pair Programming

Codice condiviso

Refactoring

TDD

Velocity

# Supporto al test: *mock objects*



Librerie che permettono di fare *behavior verification*, con oggetti “collaboratori”.

```
import org.junit.jupiter.api.Test;
import static org.assertj.core.api.Assertions.assertThat;
import static org.mockito.Mockito.*;

public class GroceryListTest {

    @Test
    public void testGroceryAddNotification() {
        GroceryList groceryList = new GroceryList(MAX_FOOD);
        @SuppressWarnings("unchecked")
        Observer<GroceryList> mockObserver =
            (Observer<GroceryList>) mock(Observer.class);
        groceryList.addObserver(mockObserver);
        groceryList.add("milk", 6);
        verify(mockObserver).onChange(groceryList);
    }
}
```

Svignippo

Monga

Tecniche di lavoro

Pair Programming

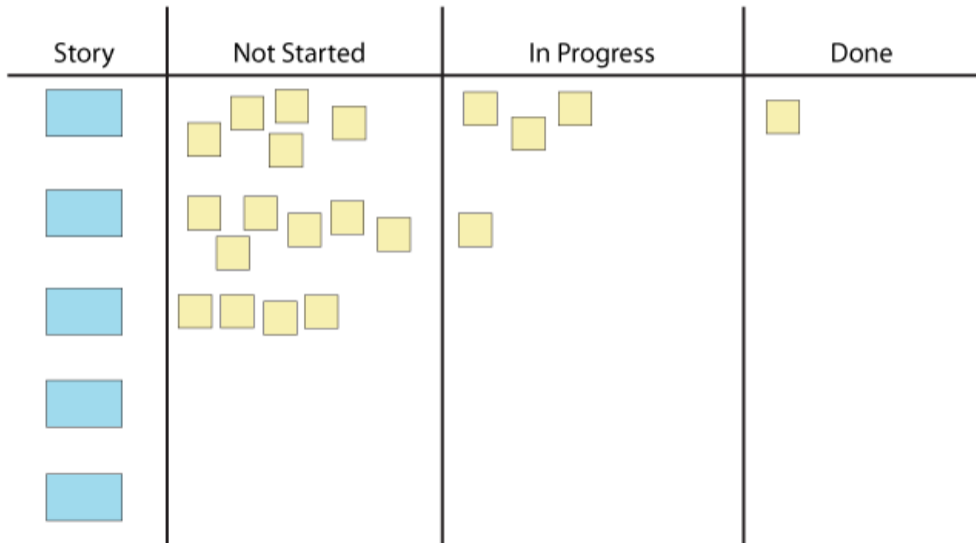
Codice condiviso

Refactoring

TDD

Velocity

# La task-board



Svigruppo

Monga

- Tecniche di lavoro
- Pair Programming
- Codice condiviso
- Refactoring
- TDD
- Velocity



Non è veramente una velocità, semmai uno “spazio percorso” in un tempo dato per fisso (lo *sprint*).

- In una iterazione (*sprint*) è la somma degli *item* in stato *Done*
- Se ne tiene traccia giornaliera con la *burn down chart*
- Inizialmente stimata riferendosi a  $\frac{1}{3}$  del tempo a disposizione (in giorni/persona); con 6 programmatori e uno *sprint* di 2 settimane:  
$$6 \times 5 \times 2 \cdot \frac{1}{3} = 20$$

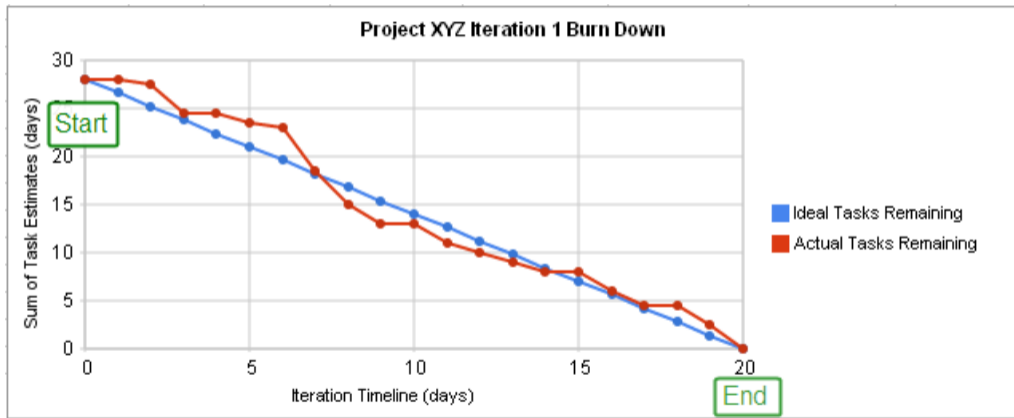
# Burn down chart



Svigruppo

Monga

- Tecniche di lavoro
- Pair Programming
- Codice condiviso
- Refactoring
- TDD
- Velocity



(By l8abug - Own work, CC BY-SA 3.0, <https://commons.wikimedia.org/w/index.php?curid=15511814>)