



Sviluppo software in gruppi di lavoro complessi¹

Mattia Monga

Dip. di Informatica
Università degli Studi di Milano, Italia
mattia.monga@unimi.it

Anno accademico 2024/25, II semestre

Svigruppo

Monga

Riassunto

Gruppi di lavoro 'agili'

Valori
Principi

I punti chiave delle metodologie agili

Punti controversi fuori dal mondo agile
Team Scrum
Tecniche di lavoro
Pair Programming
Codice condiviso
Refactoring
TDD
Velocity

¹ © 2025 M. Monga. Creative Commons Attribution — Condividi allo stesso modo 4.0 Internazionale. <http://creativecommons.org/licenses/by-sa/4.0/deed.it>



Svigruppo

Monga

Riassunto

Gruppi di lavoro 'agili'

Valori
Principi

I punti chiave delle metodologie agili

Punti controversi fuori dal mondo agile
Team Scrum
Tecniche di lavoro
Pair Programming
Codice condiviso
Refactoring
TDD
Velocity

Lezione VI: Metodologie agili



Riassumendo...

- Sviluppare sw in gruppo è difficile (*The Tar Pit*): poiché le attività sono difficilmente *separabili* la complessità del coordinamento aumenta col quadrato del numero di componenti ("Legge di Brooks")
- La soluzione 'gerarchica': la squadra di lavoro gioca per il progettista (*cattedrale/sala operatoria*)
- L'esistenza di progetti *open source* con migliaia di partecipanti sembra mettere in dubbio la legge di Brooks
- L'organizzazione di questi gruppi è molto decentrata e le idee progettuali vengono da più parti (*bazaar*)
- *Legge di Linus*: se c'è un obiettivo condiviso chiaro, le attività di verifica e manutenzione perfetta *sono* parallelizzabili almeno in parte
- In realtà però la *Tar Pit* e il *caos* sono sempre in agguato: più che dei *bazaar* disordinati, i progetti distribuiti tendono più ad assomigliare a dei *kibbutz*, con *valori tecnologici condivisi* e regole di partecipazione e sviluppo *forzate da strumenti software*

Svigruppo

Monga

Riassunto

Gruppi di lavoro 'agili'

Valori
Principi

I punti chiave delle metodologie agili

Punti controversi fuori dal mondo agile
Team Scrum
Tecniche di lavoro
Pair Programming
Codice condiviso
Refactoring
TDD
Velocity



Metodologie 'agili'

Negli anni '90 (dominati da "UML") nascono molti approcci legati soprattutto dal fastidio per l'eccessiva enfasi data alla *documentazione formale del processo di sviluppo*.

- *eXtreme Programming* (XP)
- *Scrum*
- DSDM, Adaptive Software Development, Crystal, Feature-Driven Development, Pragmatic Programming, ...

Meno rivoluzionari di quanto sembri a prima vista: i ricercatori dell'ingegneria del sw hanno sempre messo in evidenza la **crucialità del processo di sviluppo** e studiato cicli di vita *iterativi*.

Svigruppo

Monga

Riassunto

Gruppi di lavoro 'agili'

Valori
Principi

I punti chiave delle metodologie agili

Punti controversi fuori dal mondo agile
Team Scrum
Tecniche di lavoro
Pair Programming
Codice condiviso
Refactoring
TDD
Velocity

Il manifesto 'agile'



Svigruppo

Monga

Riassunto

Gruppi di lavoro 'agili'

Valori

Principi

I punti chiave delle metodologie agili

Punti controversi fuori dal mondo agile

Team Scrum

Tecniche di lavoro

Pair Programming

Codice condiviso

Refactoring

TDD

Velocity

Il termine agile è stato proposto da Martin Fowler. Nel 2001 viene pubblicato un *manifesto* dei 'valori' agili:

<http://agilemanifesto.org/>

- Gli individui e le interazioni più che i processi e gli strumenti
- Il *software* funzionante più che la documentazione esaustiva
- La collaborazione col cliente più che la negoziazione dei contratti
- Rispondere al cambiamento più che seguire un piano

36

I principi 'agili'



Svigruppo

Monga

Riassunto

Gruppi di lavoro 'agili'

Valori

Principi

I punti chiave delle metodologie agili

Punti controversi fuori dal mondo agile

Team Scrum

Tecniche di lavoro

Pair Programming

Codice condiviso

Refactoring

TDD

Velocity

- 1 Rilasciare *software* di valore, fin da subito e in maniera continua
- 2 Cambiamenti nei requisiti, anche a stadi avanzati
- 3 Consegniamo frequentemente *software* funzionante
- 4 Committenti e sviluppatori devono lavorare insieme quotidianamente
- 5 Individui motivati e ben supportati
- 6 Conversazione faccia a faccia
- 7 Il software funzionante è la principale misura di progresso
- 8 Sviluppo sostenibile: essere in grado di mantenere indefinitamente un ritmo costante
- 9 Eccellenza tecnica
- 10 La semplicità — l'arte di massimizzare la quantità di lavoro non svolto — è essenziale
- 11 Team che si auto-organizzano
- 12 A intervalli regolari il team riflette su come diventare più efficace

37

Lezione VIII: Gruppi di lavoro agili



Svigruppo

Monga

Riassunto

Gruppi di lavoro 'agili'

Valori

Principi

I punti chiave delle metodologie agili

Punti controversi fuori dal mondo agile

Team Scrum

Tecniche di lavoro

Pair Programming

Codice condiviso

Refactoring

TDD

Velocity

I veri punti chiave



Svigruppo

Monga

Riassunto

Gruppi di lavoro 'agili'

Valori

Principi

I punti chiave delle metodologie agili

Punti controversi fuori dal mondo agile

Team Scrum

Tecniche di lavoro

Pair Programming

Codice condiviso

Refactoring

TDD

Velocity

- *Team* piccoli e auto-organizzati, senza *manager* tradizionali, ma *facilitatori*
- Rifiuto di azioni e decisioni **big upfront**, sviluppo iterativo aperto alle variazioni in corso d'opera (rigorosamente regolate)
- Misura e controllo del processo di sviluppo, con pianificazioni con orizzonti temporali e funzionali ridotti
- Enfasi sul *testing*: non solo come *verifica & convalida*, ma come supporto alla progettazione, allo sviluppo e alla gestione delle variazioni

La parte più problematica è la *partecipazione della committenza*, che infatti è interpretati in maniera molto diversa dai vari approcci agili.

40

41

Semplicità e minimalismo



Svigruppo

Monga

Riassunto

Gruppi di lavoro 'agili'

Valori
Principi

I punti chiave delle metodologie agili

Punti controversi fuori dal mondo agile

Team Scrum
Tecniche di lavoro
Pair Programming
Codice condiviso
Refactoring
TDD
Velocity

- *You aren't gonna need it* (YAGNI): non pensare né implementare una funzionalità finché non è davvero necessaria; realizzare la cosa più semplice che può funzionare.
- È in esplicito contrasto con il principio dell'ingegneria del sw classica "*Design for change*": se il cambiamento/adattabilità non è adeguatamente progettato costerà troppo.

42

User stories e test



Svigruppo

Monga

Riassunto

Gruppi di lavoro 'agili'

Valori
Principi

I punti chiave delle metodologie agili

Punti controversi fuori dal mondo agile

Team Scrum
Tecniche di lavoro
Pair Programming
Codice condiviso
Refactoring
TDD
Velocity

"As a . . . I want *<business_functionality>* so that *<business_justification>*"

`assert_equal(fizzbuzz(1), 1)`

- Invece di *requisiti*, si usano storie d'uso, senza casi eccezionali, evidenza delle dipendenze. . .
- Invece di *specifiche*, si usano casi di test, con descrizioni estensive anziché intensive. . .

43

I team Scrum



Svigruppo

Monga

Riassunto

Gruppi di lavoro 'agili'

Valori
Principi

I punti chiave delle metodologie agili

Punti controversi fuori dal mondo agile

Team Scrum
Tecniche di lavoro
Pair Programming
Codice condiviso
Refactoring
TDD
Velocity

<https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-US.pdf>

- 7 ± 2 membri, *product owner*, *scrum master*
- Riunioni periodiche con scopi diversi, *daily stand-up*
- Il *product owner*: interfaccia col cliente/committente, fissa le priorità in base opportunità e rischi di *business*, gestisce il *backlog*
- Lo *scrum master*: cura il supporto al lavoro del gruppo, elimina gli impedimenti, fa rispettare le regole
- Gli altri: stimano la complessità del lavoro, identificano i rischi, dimostrano il progresso del prodotto

44

Le modalità di lavoro



Svigruppo

Monga

Riassunto

Gruppi di lavoro 'agili'

Valori
Principi

I punti chiave delle metodologie agili

Punti controversi fuori dal mondo agile

Team Scrum
Tecniche di lavoro
Pair Programming
Codice condiviso
Refactoring
TDD
Velocity

- Il lavoro è frazionato in epoche, fatte di storie, rilasciate con **sprint** di 1-3 settimane
- **closed window rule**: durante uno *sprint* non si possono aggiungere funzionalità (se proprio è necessario, lo *sprint* ricomincia)
- Nelle riunioni di pianificazione i membri stimano la complessità con il *planning poker*, facilitato dallo *scrum master*, usando una 'storia di riferimento' come unità di misura: $\frac{1}{2}, 1, 2, 3, 5, 8, 13, 20, 40, 100$
- Nelle riunioni si identificano *pigs* (direttamente coinvolti) e *chicken* (solo interessati) che danno pareri solo se richiesto dai *pigs*

45



Svigruppo

Monga

Riassunto

Gruppi di lavoro 'agili'

Valori

Principi

I punti chiave delle metodologie agili

Punti controversi fuori dal mondo agile

Team Scrum

Tecniche di lavoro

Pair Programming

Codice condiviso

Refactoring

TDD

Velocity

Daily stand-up (15 min.) Cosa abbiamo fatto ieri, cosa facciamo oggi, ci sono impedimenti?

Planning (1-5 giorni) Pianificazione di uno *sprint*, definizione dello sprint backlog con la stima per ogni epopea/storia

Retrospektiva (30 min.) Alla fine di uno *sprint*, per migliorare

Review (1 ora) Alla fine di uno *sprint*, presentazione del lavoro agli *stakeholder*

46



Svigruppo

Monga

Riassunto

Gruppi di lavoro 'agili'

Valori

Principi

I punti chiave delle metodologie agili

Punti controversi fuori dal mondo agile

Team Scrum

Tecniche di lavoro

Pair Programming

Codice condiviso

Refactoring

TDD

Velocity

Increment

An Increment is a concrete stepping stone toward the Product Goal. Each Increment is additive to all prior Increments and thoroughly verified, ensuring that all Increments work together. In order to provide value, the Increment must be usable. Multiple Increments may be created within a Sprint.

Definition of Done

The Definition of Done is a formal description of the state of the Increment when it meets the quality measures required for the product. If it is not an organizational standard, the Scrum Team must create a Definition of Done appropriate for the product. The Developers are required to conform to the Definition of Done.

Esempi: l'incremento supera i test di unità e di integrazione, oppure, *deployed on users' server*

47



Svigruppo

Monga

Riassunto

Gruppi di lavoro 'agili'

Valori

Principi

I punti chiave delle metodologie agili

Punti controversi fuori dal mondo agile

Team Scrum

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*

48



Svigruppo

Monga

Riassunto

Gruppi di lavoro 'agili'

Valori

Principi

I punti chiave delle metodologie agili

Punti controversi fuori dal mondo agile

Team Scrum

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.

49



Svigruppo

Monga

Riassunto

Gruppi di lavoro 'agili'

Valori
Principi

I punti chiave delle metodologie agili

Punti controversi fuori dal mondo agile

Team Scrum

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*

50



Svigruppo

Monga

Riassunto

Gruppi di lavoro 'agili'

Valori
Principi

I punti chiave delle metodologie agili

Punti controversi fuori dal mondo agile

Team Scrum

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/>

51



Svigruppo

Monga

Riassunto

Gruppi di lavoro 'agili'

Valori
Principi

I punti chiave delle metodologie agili

Punti controversi fuori dal mondo agile

Team Scrum

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)

52



Svigruppo

Monga

Riassunto

Gruppi di lavoro 'agili'

Valori
Principi

I punti chiave delle metodologie agili

Punti controversi fuori dal mondo agile

Team Scrum

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*

53

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);
    }
}
```

54

Svigruppo

Monga

Riassunto

Gruppi di lavoro 'agili'

Valori

Principi

I punti chiave delle metodologie agili

Punti controversi fuori dal mondo agile

Team Scrum

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).onChanges(groceryList);
    }
}
```

55

Svigruppo

Monga

Riassunto

Gruppi di lavoro 'agili'

Valori

Principi

I punti chiave delle metodologie agili

Punti controversi fuori dal mondo agile

Team Scrum

Tecniche di lavoro

Pair Programming

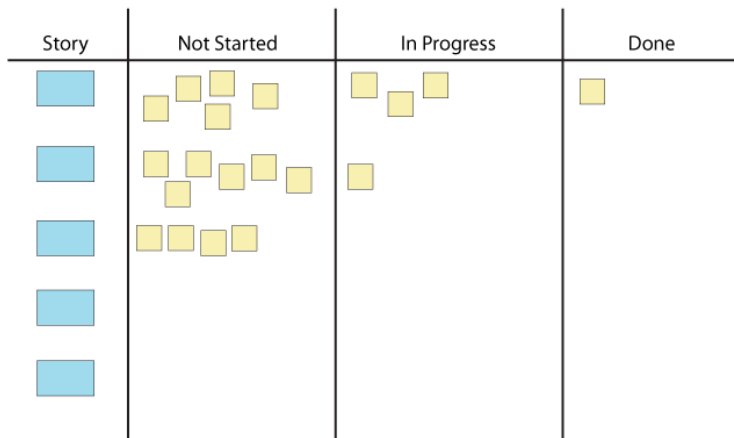
Codice condiviso

Refactoring

TDD

Velocity

La task-board



56

Svigruppo

Monga

Riassunto

Gruppi di lavoro 'agili'

Valori

Principi

I punti chiave delle metodologie agili

Punti controversi fuori dal mondo agile

Team Scrum

Tecniche di lavoro

Pair Programming

Codice condiviso

Refactoring

TDD

Velocity

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: $6 \times 5 \times 2 \cdot \frac{1}{3} = 20$

57

Svigruppo

Monga

Riassunto

Gruppi di lavoro 'agili'

Valori

Principi

I punti chiave delle metodologie agili

Punti controversi fuori dal mondo agile

Team Scrum

Tecniche di lavoro

Pair Programming

Codice condiviso

Refactoring

TDD

Velocity

Burn down chart



Svigruppo

Monga

Riassunto

Gruppi di lavoro 'agili'

Valori
Principi

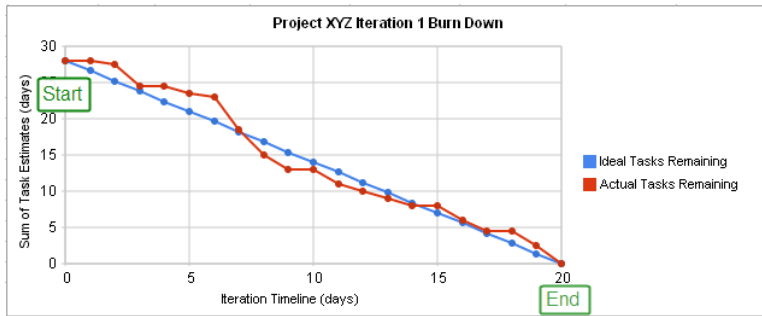
I punti chiave delle metodologie agili

Punti controversi fuori dal mondo agile

Team Scrum
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>)