Luca Cabibbo

Analisi e Progettazione del Software

Sviluppo iterativo ed evolutivo

Capitolo 2

marzo 2024

Lo sviluppo iterativo dovrebbe essere utilizzato solo per i progetti che si desidera vadano a buon fine.

Martin Fowler

2.2 Processi per lo sviluppo del software

Un **processo** (o **metodo**) **per lo sviluppo del software** – o **processo software** – definisce un approccio disciplinato per la costruzione, il rilascio e la manutenzione del software

- **chi** fa **che cosa**, **quando** e **come** – per raggiungere un certo obiettivo
- ruoli, attività, organizzazione temporale delle attività e metodologie

Esistono numerosi processi software

- il processo a cascata, UP, Scrum, XP, ...
- tutti organizzati attorno ad alcune attività fondamentali comuni

Attività nei processi software

Attività comuni nei processi software

- *specifica* (o *analisi*) dei *requisiti*
- *analisi*
- *progettazione*
- *implementazione*
- *validazione e verifica*
- *rilascio/installazione*
- *manutenzione/evoluzione*
- *gestione del progetto*

Differenziazione tra processi software

Perché esistono diversi processi software? In cosa si differenziano?

- nel “che cosa”?
- nel “chi”?
- nel “come”?
- nel “quando”?

- i processi si differenziano soprattutto nel “quando”
  - nell’ordine delle attività
  - nei criteri di transizione da un’attività alla successiva
- ovvero, nel modo di rispondere a queste domande
  - per quanto tempo continueremo a fare questa cosa?
  - che cosa faremo dopo?

* Disclaimer – Processi vs. Persone

I principi, le tecnologie e i processi sono importanti – ma sono più importanti le persone che li mettono in pratica [Cockburn]

- i processi e le tecnologie hanno un impatto secondario nei risultati di un progetto
- le persone – con le loro individualità, sentimenti e qualità – sono molto più importanti

Le persone sono più importanti dei processi [Booch]

- persone buone che adottano un buon processo lavoreranno meglio, in ogni caso, di buone persone senza processo

Per realizzare qualcosa di valore, le persone devono mettere "anima e cuore" in tutto quello che fanno

- i "processi" sono solo per migliorare il modo in cui le persone lavorano insieme

2.1 Che cos'è UP

Unified Process (**UP**) è un processo **iterativo** per lo sviluppo di software OO

- l'OOA/D si applica al meglio nei processi *iterativi* e *agili*
- UP è aperto e flessibile – incoraggia l'adozione di pratiche da altri processi e metodi
  - ad es., TDD, refactoring, integrazione continua, …
  - da Scrum, Extreme Programming (XP), Agile Modeling, …
- in ogni caso, le idee fondamentali del corso sono indipendenti da ogni particolare processo – e possono essere applicate anche nel contesto di altri processi

2.3 Il processo "a cascata"

Il **processo a cascata** è un processo software "classico" – definito negli anni '60/'70 – ma purtroppo ancora diffuso

- attività svolte in modo sequenziale
  - pianificazione – con stime dettagliate per tutte le attività
  - analisi dei requisiti del software
  - progettazione
  - implementazione del codice
  - collaudo
- inoltre
  - ciascuna attività produce documenti dettagliati
  - in accordo con il piano iniziale, il lavoro procede da un'attività alla successiva solo con l'approvazione dei documenti dell'attività precedente
  - le diverse attività sono spesso svolte da team differenti

Il processo "a cascata"

Il **processo a cascata** è un processo software "classico" – definito negli anni '60/'70 – ma purtroppo ancora diffuso

Che origini ha?

Può funzionare per lo sviluppo del software?

Potrebbe non funzionare?
In quali casi? Perché?
Si può fare di meglio?

tività

Il processo a cascata è poco efficace

Il processo a cascata è spesso poco efficace

- elevata percentuale di fallimenti, bassa produttività, percentuali di difetti alte

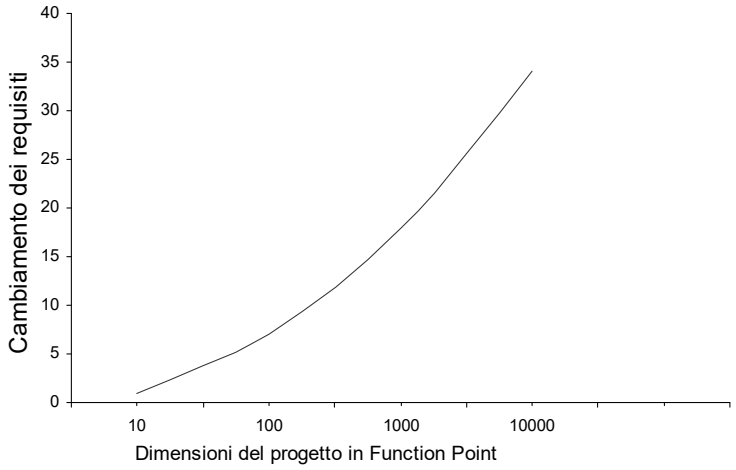
Perché è poco efficace?

- un'ipotesi fondamentale è poter definire correttamente e "congelare" i requisiti prima di procedere con le fasi successive
  - questa ipotesi non è realistica
- tutte le "buone idee" dovrebbero venire all'inizio del progetto
  - non consente l'introduzione di buone idee "ritardatarie"
- non consente una gestione efficace dei rischi
  - perché i rischi vengono affrontati *tardi* nel ciclo di vita
- l'uso della prototipazione per validare i requisiti mitiga un pochino questo problema – ma non lo risolve

Il processo a cascata è poco efficace

Studi empirici hanno dimostrato che

- i requisiti cambiano del 25% nei progetti software di medie dimensioni – ma anche del 35%-50% nei progetti grandi e/o innovativi

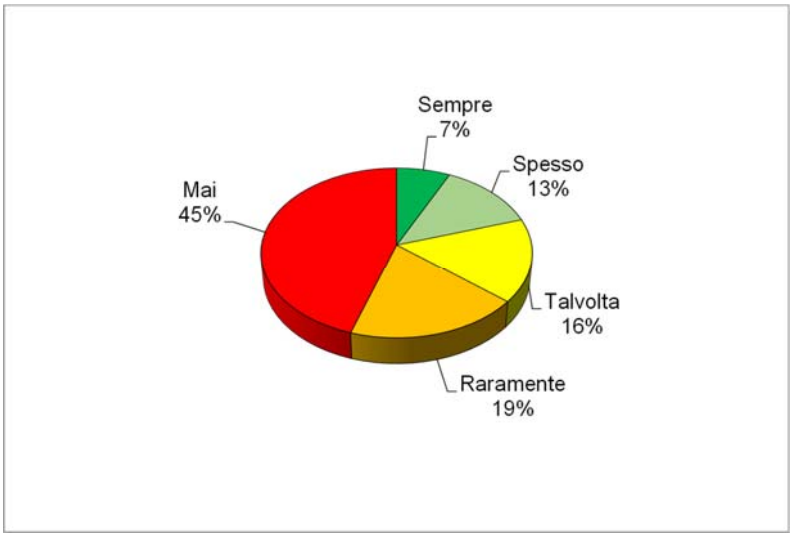


- la mancata gestione del cambiamento è una causa comune del fallimento dei progetti software

Requisiti che cambiano

Un'altra indagine

- se viene fatta un'analisi "a cascata" dei requisiti, quante caratteristiche identificate inizialmente saranno effettivamente utili nel prodotto finale?



Il processo a cascata è poco efficace

Il **processo a cascata** è un processo software "classico" – definito negli anni '60/'70 – ma purtroppo ancora diffuso

Processo a cascata:
si può fare di meglio?

2.4 Lo sviluppo evolutivo ed iterativo

L'idea fondamentale dello **sviluppo evolutivo**

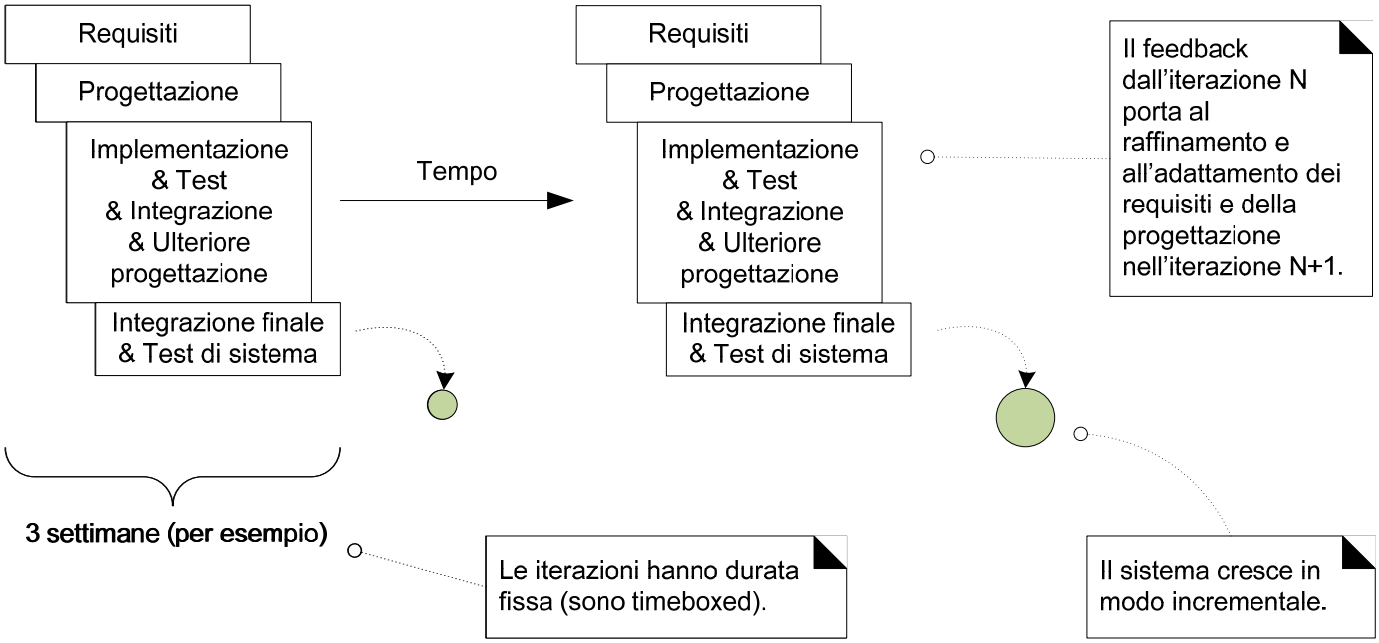
- realizzare un'implementazione iniziale del sistema, esporla agli utenti per ottenere un feedback e poi raffinarla attraverso diverse versioni, finché non si ottiene un sistema adeguato
- l'evoluzione è guidata da esperienze pratiche dell'utilizzo del sistema e dal relativo feedback
  - *sì, è più o meno quello che volevo – ma in effetti volevo qualcosa di un po' diverso…*

Lo sviluppo iterativo

Nello **sviluppo iterativo** lo sviluppo del software è organizzato in una serie di mini-progetti brevi, di durata fissa (ad es., 2-4 settimane) chiamati **iterazioni**

- in ogni iterazione vengono svolte attività di analisi dei requisiti, analisi, progettazione, implementazione, verifica, …
- il risultato di ciascuna iterazione è un ***sistema software eseguibile***, integrato e testato, anche se ***parziale***
- il sistema cresce in modo **incrementale** – e viene adattato ai requisiti, in modo **evolutivo**, sulla base del feedback delle iterazioni precedenti
- il software converge verso un sistema completo dopo varie iterazioni

Sviluppo iterativo e incrementale



Sviluppo iterativo

Lo sviluppo iterativo è **incrementale** ed **evolutivo**

- il sistema cresce in modo incrementale, iterazione per iterazione
- le specifiche e il progetto evolvono, in base a feedback ed adattamento

L'idea di fondo dello sviluppo iterativo

- in ciascuna iterazione, considerare solo un piccolo insieme di requisiti
- anticipare nel tempo alcune attività e rimandarne altre
- massimizzare i benefici e minimizzare i rischi

Cambiamento e adattamento

Nello sviluppo del software, i cambiamenti sono inevitabili

- lo sviluppo iterativo accetta e “abbraccia” il cambiamento e l’adattamento, scegliendoli come guide essenziali
  - ma non vuol dire lavorare in modo caotico (“feature creep”)
- ma come vengono gestiti i cambiamenti?
  - il lavoro procede mediante una serie di iterazioni di costruzione-feedback-adattamento
  - i cambiamenti più significativi dovrebbero essere sollecitati soprattutto nelle iterazioni iniziali

Feedback e adattamento

Feedback e adattamento nello sviluppo iterativo

- feedback proveniente dalle attività di sviluppo – in relazione ai requisiti e alla loro comprensione da parte del team di sviluppo
- feedback proveniente dai test
- feedback proveniente dagli sviluppatori che raffinano i modelli e il progetto
- feedback per raffinare le stime dei tempi e dei costi
- feedback proveniente dai clienti e dal mercato – sulla priorità da assegnare alle caratteristiche da sviluppare
- feedback sull’applicazione del processo software

Vantaggi (e rischi) dello sviluppo iterativo

Vantaggi dello sviluppo iterativo

- minor probabilità di fallimento del progetto, miglior produttività, percentuali più basse di difetti
- riduzione precoce anziché tardiva dei rischi maggiori
- visibilità del progresso
- feedback precoce, impegno degli utenti, adattamento
- gestione della complessità
- miglioramento continuo del processo stesso

Ovviamente ci sono anche dei rischi – ma conoscendoli è possibile affrontarli in modo opportuno

- ad esempio
  - come gestisco il tempo?
  - in che ordine considero i requisiti?
  - come gestisco in pratica i cambiamenti?

Progetti iterativi vs. pensare a cascata

Un rischio comune

- dire di adottare un processo iterativo… ma in realtà pensare a cascata
- purtroppo visto anche in aziende, tirocini e tesi ☹ ma anche in alcuni libri di ingegneria del software

In questo caso

- il pensiero a cascata ha inficiato il progetto – e non si tratta di un progetto iterativo sano

Timeboxing

Timeboxing – una buona pratica dello sviluppo iterativo – ogni iterazione deve avere una durata *prefissata*

- iterazioni *brevi* consentono un adeguato feedback e adattamento ai cambiamenti
- le iterazioni sono **timeboxed**
  - la durata di un'iterazione non può cambiare
  - ogni iterazione deve comunque produrre un sistema eseguibile – nel caso, è meglio ridurre i requisiti di un'iterazione

Sviluppo iterativo e qualità del codice

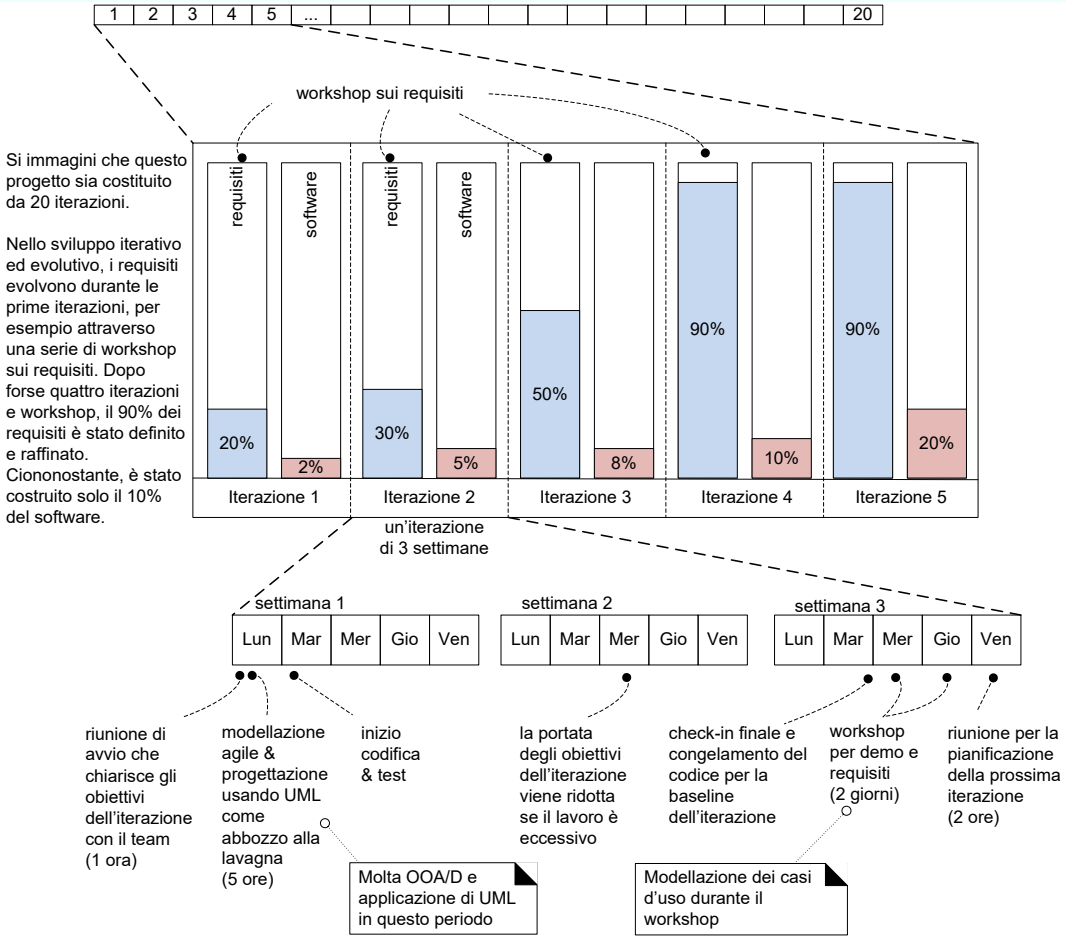
Nello sviluppo iterativo, è bene che il software possegga alcune qualità

- la struttura del software deve essere **flessibile** – in modo tale che l'impatto dei cambiamenti sul sistema sia basso
  - a tal fine, il codice (e il progetto sottostante) deve essere facilmente **modificabile**
  - inoltre, come prerequisito, il codice (e il progetto sottostante) deve essere facilmente **comprensibile**

Come sostenere queste qualità?

- applicando principi di progettazione opportuni – LRG, modularità, separazione degli interessi, ...
- utilizzando strumenti (metodologici) opportuni – tecnologie OO, refactoring, TDD, tracciatura dei requisiti e gestione delle modifiche, pianificazione iterativa, ...

2.5 OOA/D iterativa ed evolutiva



2.6 Pianificazione iterativa

La **pianificazione iterativa** – una pratica fondamentale dello sviluppo iterativo

- che cosa fare nella prossima iterazione?

Pianificazione iterativa **guidata dal rischio** e **guidata dal cliente**

- nelle prime iterazioni
  - identificare e affrontare i rischi principali
  - implementare caratteristiche visibili a cui il cliente è particolarmente interessato
- è compatibile con lo sviluppo **centrato sull'architettura**
  - le prime iterazioni si concentrano sulla costruzione, il test e la stabilizzazione del nucleo dell'architettura

A P S * Pianificazione iterativa e backlog

Il **backlog** ("lavoro arretrato") – uno strumento per l'organizzazione del lavoro che deve essere ancora svolto nello sviluppo iterativo

- un elenco di voci – requisiti da implementare e problemi da risolvere
- la gestione è iterativa e guida la pianificazione e l'evoluzione del processo di sviluppo
- prima di iniziare una nuova iterazione
  - le voci del backlog vengono aggiornate – anche sulla base del feedback
  - a ciascuna voce viene assegnata (o riassegnata) una priorità
  - per l'iterazione che sta per iniziare viene scelto come compito l'implementazione di voci tra quelle che hanno priorità maggiore

A P S Iterazioni e requisiti bloccati

Durante un'iterazione, i requisiti su cui operare vengono prefissati (stabiliti all'inizio dell'iterazione) e bloccati (non possono cambiare durante l'iterazione)

- in questo modo, il team di sviluppo, nell'ambito di ciascuna iterazione, può lavorare al suo meglio
- i committenti possono interagire con il team al termine di ciascuna iterazione (che sono brevi)
- infatti, consentire la variazione degli obiettivi di un'iterazione durante l'iterazione ha in genere conseguenze negative

2.7 Altre pratiche fondamentali di UP

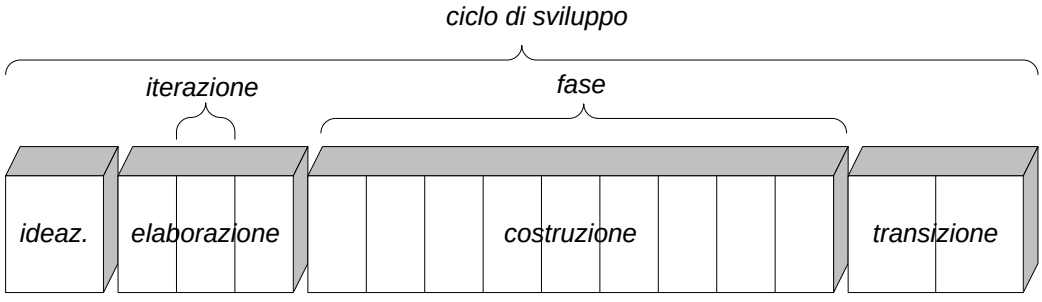
UP promuove diverse best practice

- sviluppare software in modo iterativo, evolutivo ed adattivo
- sviluppo guidato dal rischio
- sviluppare il cuore dell'architettura nelle prime iterazioni
- coinvolgimento continuo degli utenti
- verifica continua delle qualità – testare presto, spesso, e in modo realistico
- gestire attentamente i requisiti
- applicare i casi d'uso – se appropriato
- uso delle tecnologie a oggetti (OOA/OOD/OOP)
- modellare in modo visuale
- gestire le richieste di cambiamento e le configurazioni
- sviluppo basato su componenti

2.8 Fasi di UP

Quattro **fasi** principali nell'organizzazione di un progetto UP

- **ideazione**, **elaborazione**, **costruzione** e **transizione**



milestone

La fine di un'iterazione in cui si verifica una decisione o una valutazione significativa.

release

Un sottoinsieme stabile ed eseguibile del prodotto finale. La fine di ogni iterazione produce una release minore.

incremento

La differenza (delta) tra le release di due iterazioni successive.

release produzione finale

A questo punto il sistema viene rilasciato e consegnato ai clienti.

2.9 Discipline di UP

In UP, le attività lavorative (compiti) sono organizzate in discipline

- **disciplina** – le attività e gli elaborati relativi a una determinata area
- **elaborato** (*artifact*) – termine generico che indica un prodotto di lavoro

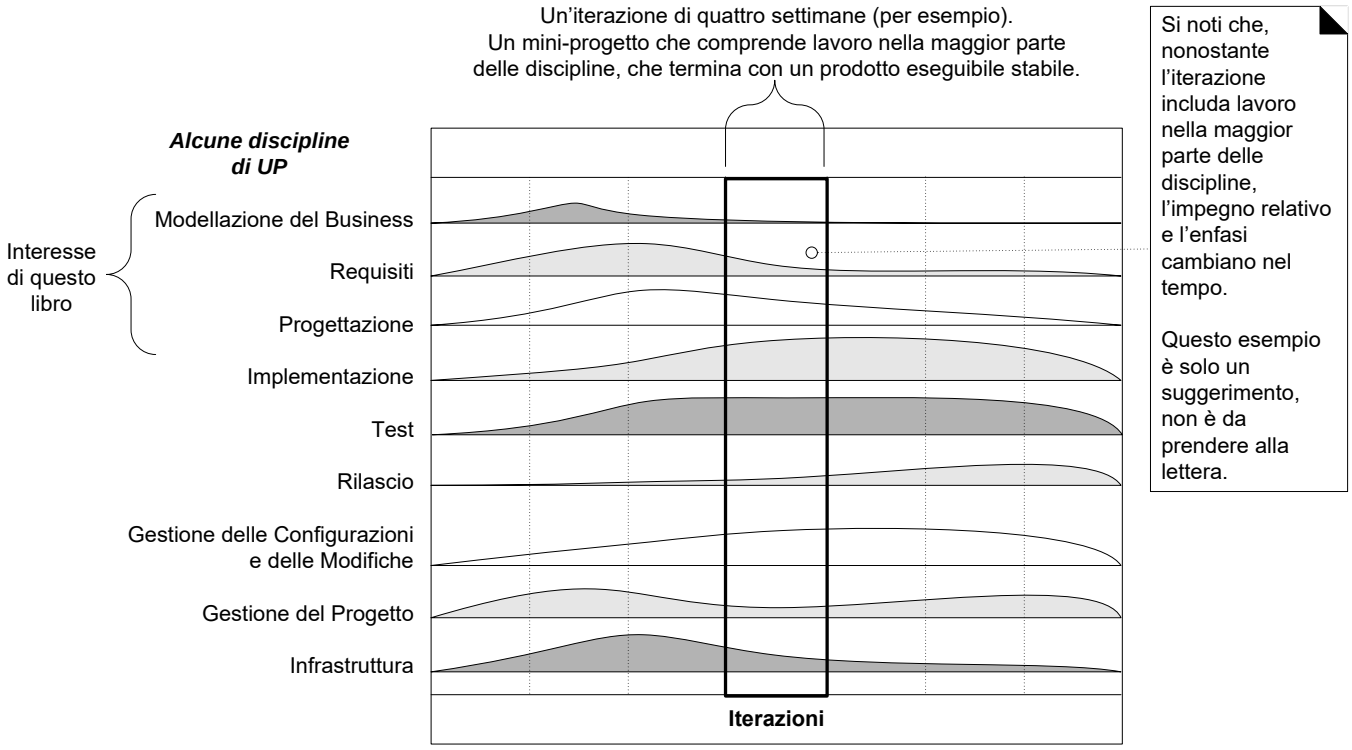
Tre discipline di UP sono di interesse per questo corso

- **requisiti**
- **modellazione del business** (business modeling)
- **progettazione**

Alcune ulteriori discipline di UP

- *implementazione*
- *test*
- *gestione del progetto*

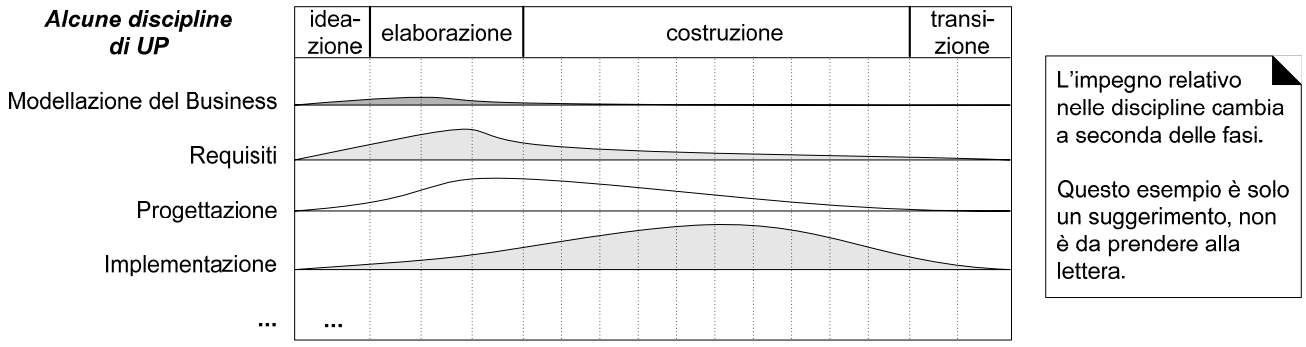
Discipline di UP e sviluppo iterativo



Discipline di UP e sviluppo iterativo

Nello sviluppo iterativo

- ciascuna iterazione coinvolge molte o tutte le discipline
- lo sforzo richiesto da ciascuna disciplina cambia durante le varie fasi e iterazioni



L'impegno relativo nelle discipline cambia a seconda delle fasi.

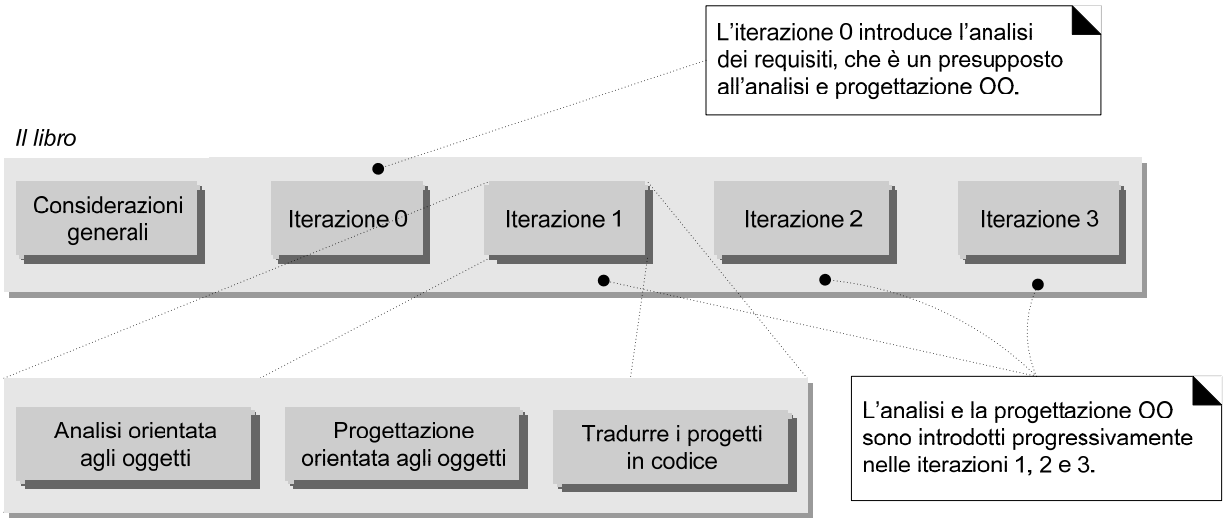
Questo esempio è solo un suggerimento, non è da prendere alla lettera.

Struttura del corso, fasi e discipline

Questo corso enfatizza

- soprattutto la fase di elaborazione
- alcune attività ed elaborati delle discipline di modellazione del business, requisiti, progettazione

che sono la fase e le discipline in cui vengono principalmente applicate l'analisi dei requisiti, l'OOA, l'OOD, i pattern, UML



2.10 Personalizzare UP

Alcuni principi e pratiche di UP sono indispensabili

- ad eccezione del codice, tutte le attività e gli elaborati sono opzionali
- ciascun progetto va basato sugli elaborati e le attività che sono effettivamente di valore per quel particolare progetto

La scelta degli elaborati per un progetto viene descritta nello **Scenario di sviluppo** (development case)

- un breve elaborato nella disciplina di infrastruttura (environment)