

Ingegneria del Software

Angelo Passarelli

September 21, 2023



Appunti basati sulle lezioni e dispense della professoressa Laura Semini ¹

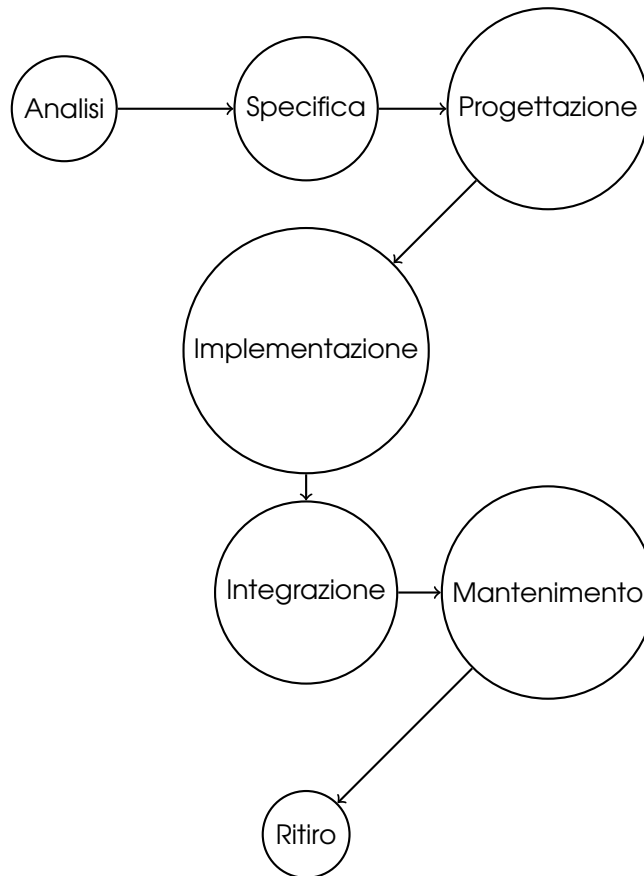
¹<http://didawiki.cli.di.unipi.it/doku.php/informatica/is-a/start>

Contents

0	Introduzione	3
1	Modelli di Ciclo di Vita	4
1.1	Modelli Sequenziali	4
1.1.1	Build-and-Fix	4
1.1.2	Modello a Cascata	5
1.1.3	Modello a V	5
1.2	Modelli Iterativi	6
1.2.1	Rapid Prototyping	6
1.2.2	Modello Incrementale	6
1.2.3	Modello a Spirale	8
1.3	Unified Process	9
1.4	Processi Agili	9

0 Introduzione

Fasi del progetto



Specificità del Software

1. **Fault tolerance**: capacità del software di essere tollerante ai guasti.
2. **Difetto latente**: difetto nascosto che si trova difficilmente in fase di testing; e anche nel caso comparisse è quasi impossibile da ritrovare.
3. **Robustezza**: capacità di funzionare anche con input non previsti e/o non testati.
4. Il software non presenta **costi materiali** e nemmeno **costi marginali**, ovvero il costo di un'unità del prodotto.

5. Infine il software non si consuma nel tempo, ma potrebbe diventare **obsoleto**.

La Manutenzione

Costi La fase di manutenzione è quella che richiede costi più alti. Per evitare uno spreco durante questa fase è necessario studiare bene l'analisi dei requisiti, in quanto un errore in questa fase si propagherà in modo esponenziale, in termini di costi, nelle fasi successive.

La manutenzione si divide in:

- **Manutenzione Correttiva**: rimuove gli errori, lasciando invariata la specifica.
- **Manutenzione Migliorativa**: consiste nel cambiare quella che è la specifica, e a sua volta può dividersi in:
 - **Perfettiva**: modifiche per migliorare e/o introdurre nuove funzionalità.
 - **Adattiva**: modifiche indotta da cambiamenti esterni, come leggi o modifiche all'hardware o al sistema operativo.

Stakeholders

- **Fornitore**: colui che sviluppa il software.
- **Committente**: chi lo richiede e paga.
- **Utente**: chi lo usa.

1 Modelli di Ciclo di Vita

Definizione (Processo Software). Con processo software si indica il percorso da seguire per sviluppare un prodotto o più nello specifico un software. Fanno parte del processo sia gli strumenti e le tecniche per lo sviluppo che i professionisti coinvolti.

1.1 Modelli Sequenziali

1.1.1 Build-and-Fix

Il prodotto è sviluppato senza alcuna fase di progettazione preliminare, lo sviluppatore scrive il software e poi lo modifica ogni volta che non soddisfa il committente.

Contro Diventa improponibile per progetti grandi e la manutenzione diventa difficile senza documentazione nè specifica.

1.1.2 Modello a Cascata

Questo modello è stato il primo a distinguere il processo software in più fasi, evidenziando l'importanza della progettazione e dell'analisi.

Viene chiamato anche modello *document driven* dato che ogni fase produce un documento, e per passare alla successiva occorre aver approvato il documento della fase precedente.

Contro Troppo pesante da seguire, inoltre non si può tornare indietro, e mancando l'interazione con il cliente, se non è soddisfatto, va tutto ripetuto dall'inizio.

1.1.3 Modello a V

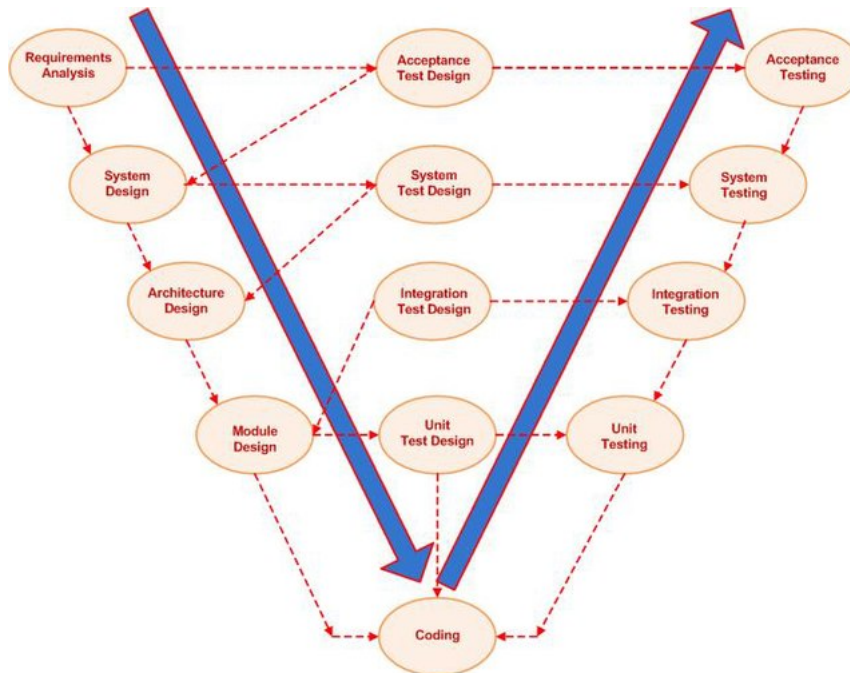


Figure 1: Le frecce blu rappresentano il *tempo*, mentre quelle tratteggiate le *dipendenze*

Questo modello evidenzia come sia possibile progettare i **test** durante le fasi di sviluppo (quelle a sinistra, prima della fase di *coding*). Mentre sulla destra sono presenti i test veri e propri che devono verificare e convalidare l'attività in corrispondenza sulla sinistra.

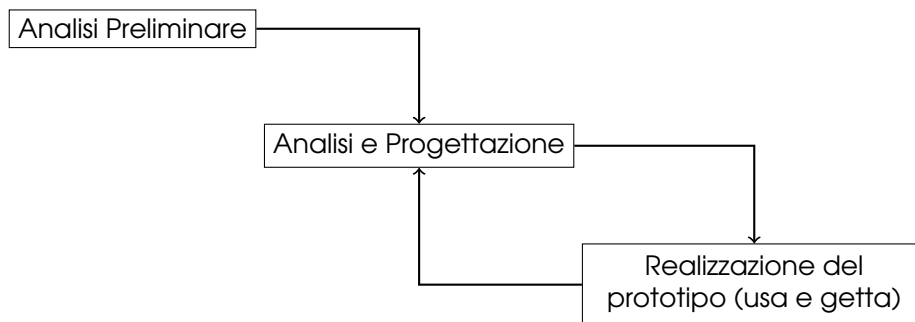
Standard SQA Questo modello è uno degli standard SQA (Software Quality Assurance), usato per descrivere le attività di test durante il processo di sviluppo.

1.2 Modelli Iterativi

1.2.1 Rapid Prototyping

L'obiettivo è quello di costruire rapidamente un prototipo del software per permettere al committente di sperimentarlo.

Questo modello diventa utile quando i requisiti non sono chiari, quindi ogni prototipo aiuterà il cliente a descriverli meglio.



1.2.2 Modello Incrementale

Il software viene costruito in modo iterativo, aggiungendo di volta in volta nuove funzionalità.

I requisiti e la progettazione vengono definiti inizialmente, per questo è possibile applicarlo solo in caso di requisiti stabili.

Contro Se non viene realizzata una buona progettazione, questo modello sfocia in un *Build-and-Fix*.

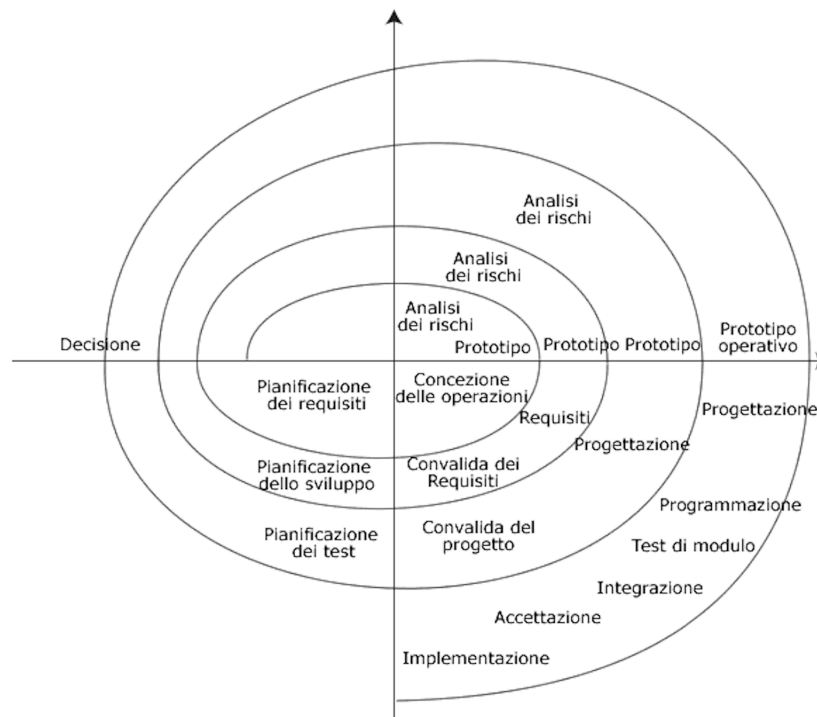


1.2.3 Modello a Spirale

In questo caso ogni iterazione è formata da 4 fasi che corrispondono ai quadranti del piano:

1. *Quadrante in alto a sinistra*: definizione degli obiettivi e dei vincoli.
2. *Quadrante in alto a destra*: analisi e risoluzione dei rischi.
3. *Quadrante in basso a destra*: sviluppo e verifica del prossimo livello.
4. *Quadrante in basso a sinistra*: pianificazione della fase successiva.

Questo modello viene anche chiamato *risk driven* in quanto è incentrato principalmente sull'analisi dei rischi. Inoltre si ispira profondamente al metodo iterativo *plan-do-check-act cycle*²



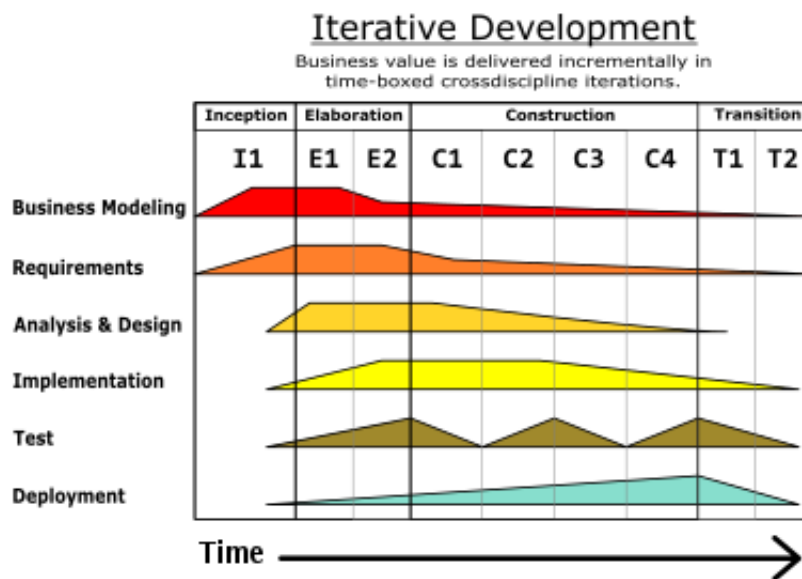
²https://it.wikipedia.org/wiki/Ciclo_di_Deming

1.3 Unified Process

In questo modello vengono distinte quattro fasi chiamate *Inception*, *Elaboration*, *Construction* e *Transition*. Ogni fase può presentare un numero variabile di iterazioni anche in base alla dimensione del progetto.

Questo modello viene definito *iterativo incrementale*, *incrementale* perchè alla fine di ogni iterazione si ottiene un rilascio del sistema con funzionalità in più o migliorate rispetto al rilascio precedente.

Inoltre viene data molta importanza all'architettura del sistema, infatti già dalle prime fasi ci si concentra soprattutto sull'architettura anche se a livello molto superficiale, lasciando i dettagli alle fasi successive. In questo modo è molto facile avere una visione generale del sistema che sarà facilmente modellabile sulla variazione dei requisiti. Per questo, piuttosto che dai requisiti, ci si fa guidare principalmente dai *casi d'uso* e dall'*analisi dei rischi*.



1.4 Processi Agili