Il ciclo di vita del software(3)

Ciclo di vita di un software: Evolve nel tempo e raggiunge **stati** tramite **transizioni** scatenate da attività che hanno il fine di far avanzare il sw. Si divide in 4 fasi:

- **Concezione**, quando qualcuno pensa che ci sia (o abbia) bisogno di qualcosa;
- **Sviluppo**;
- **Utilizzo**;
- **Ritiro**.

Fase: `Periodo di tempo contiguo che cattura transizioni o segmenti di ciclo di vita`
La scelta del ciclo di vita pone vincoli sulla pianificazione del progetto, è indipendente da metodi e strumenti di sviluppo.

Tipi di ciclo di vita:

- **Cascata:** successione di fasi rigidamente sequenziali con dipendenze causali tra di loro, non ammette ritorno a fasi precedenti.(Pre e Post solide e definite all'origine, modello che non spreca, al tempo t so dove sono). Modello **Document driven** (documenti che spiegano la fiducia nelle scelte fatte), non sono previsti prototipi.
  Per fasi qui si intende durate temporali con dipendenze causali tra loro. Difetti: eccessivamente rigido, non ammette cambiamenti ai requisiti in corso d'opera, il committente vede solo l'opera finita.
- **Incrementale:** in questo modello procedo per fasi che possono anche essere completamente separate le une dalle altre, ma a differenza del waterfall posso integrare anche le piccole parti. Produco valore ad ogni incremento, ed ogni incremento riduce il rischio di fallimento; le funzionalità essenziali inoltre sono sviluppate nei primi incrementi. Qui si prevedono rilasci multipli e successivi(base stabile ed aggiungo successivamente). Non si ripetono mai l'analisi e la progettazione architetturale. Non ritorno mai sul progetto generale.

- **Iterativo:** comporta il rischio di non convergenza all'obbiettivo, tuttavia consentono maggior capacità di adattamento. Ogni iterazione comporta un ritorno all'indietro, nella direzione opposta rispetto al tempo. Iterazione e incremento coincidono quando la sostituzione raffina ma non ha impatto sul resto.
- **Evolutivo:** insegue il futuro non ritirando il passato, ho tante attività concorrenti. Questo modello lo attua chi può sostenere molte versioni in parallelo (e quindi ha buone capacità finanziarie), infatti ogni fase ammette iterazioni multiple e parallele. Si può anche modificare la base ATT.
- **Spirale:** è composto da cicli interni rapidi e ripetuti dedicati ad analisi e sviluppi prototipali, mentre quelli esterni aderiscono a qualsiasi altro modello std. Comporta un miglior controllo dei rischi di progetto, infatti pone molta attenzione sugli aspetti gestionali (pianificazione delle fasi, analisi dei rischi), è un modello **risk driven** e prevede una forte iterazione tra committente e fornitore. Prevede 4 attività principali:
  - Definizione degli obbiettivi (requisiti, rischi, strategia di gestione)
  - Analisi dei rischi (studio delle conseguenze, valutazione delle alternative)
  - Sviluppo e validazione (realizzazione del prodotto)
  - Pianificazione (decisione circa il proseguimento, pianificazione del proseguimento)

Utilizzato solo per progetti innovativi

- **Componenti:** si basa sul riuso dei componenti già fatti, utilizzo meno risorse ed ho un ecosistema(librerie e strumenti) già pronti.
- **Metodi Agili:** basati su 4 principi fondamentali:

  1. Mettere in primo piano persone e iterazioni piuttosto che processi e strumenti. Gli individui sono importanti ed è importante il modo in cui interagiscono tra loro;
  2. Ciò che importa e che il sw funzioni non la documentazione;
  3. Avere un buon rapporto con il customer, coinvolgerlo;
  4. Reattività piuttosto che pianificazione. Capacità di adattamento a cambiamenti delle situazioni.

  L'idea base e di costruire degli user story che sono definiti da

  - Un documento di descrizione;
  - La minuta della conversazione con il cliente;
  - Piano dei test per confermare che la strategia funziona.

  Questi tipi di metodi permettono dunque di dimostrare costantemente ciò che e stato fatto, verificare l'avanzamento tramite progresso reale, assicurare che l'intero prodotto sw è ben verificato e integrato. **Scrum**, **Kanban** e **Scrumban** sono degli esempi.

Scrum: regole interne molte collaborative. Ci sono 3 tipi di fase:

- Backlog: "le cose da fare", requisiti e funzionalità del prodotto;
- Spirit Backlog: insieme di storie del prossimo spirit
- Spirit: fase operativa di sviluppo, durata 2-4 settimane, alla fine abbiamo un prodotto potenzialmente vendibile.

Ad ogni periodo di tempo si prendono le cose più necessarie e si esegue uno spirit, non appena è terminatose manca qualcosa si ritorna al backlog aggiungendo *to do*. Questo ciclo di vita e iterativo in quanto non ho un numero casuale di spirit e questi producono quasi sempre un avanzamento.
SEMAT:
I *Requirements* hanno 6 aggettivi:

- **Conceived:** i requisiti vanno concepiti, nascono nel luogo dove stanno gli stakeholder, ovvero tutti coloro che danno forma ad un prodotto. Non significa "*completamente formato*", i requisiti vanno fatti emergere;
- **Bounded:** so cosa voglio/non voglio requisiti ben definiti;
- **Coherent:** elenco dei bisogni che ho trovato, anche qui gli stakeholder sono gli interlocutori;
- **Acceptable:**designa lo stato di maturità e dice che i requisiti sono ragionevoli per tutti;
- **Addressed:** ho la soluzione per i requisiti marcati come *Acceptable*, sono capace di dimostrare che soddisfo i requisiti questo va dimostrato nella progettazione;
- **Fullfilled**, "*compiere*". Ho un prodotto che fa esattamente ciò che mi aspettavo.

Anche il *Work le cose da fare* hanno degli stadi ben predefiniti:

- **Initiated:** avere un piano di lavoro;

- **Prepared:** avere un calendario;
- **Started:** partire;
- **Under Control:** sappiamo dove siamo;
- **Concluded:** possiamo dire di aver finito;
- **Close:** risposta degli stakeholders *"va bene"*.

Il *Team* deve essere:

- **Seeded:** si comincia a identificare di chi ho bisogno, non come persone ma come competenze. Dobbiamo spersonalizzare, importa il ruolo che ho. Nessuno è un fuoriclasse (perché il fuoriclasse non è disciplinato e sistematico, quindi dannoso), anche se in una buona organizzazione può aiutare (altrimenti fa danno);
- **Formed:** le persone ce le ho e posso lavorare; istituisco dunque tecniche e strumenti per collaborare;
- **Collaborating:** le forme di collaborazione vanno studiate per evitare perdite di tempo;
- **Performed:** una volta acquisite le tecniche siamo efficienti ed efficaci;
- **Adjourned:** "*liberi tutti*", avendo acquisito l'esperienza del lavoro fatto.