Ingegneria del Software

Introduzione

L’ingegneria del software è l’insieme delle teorie, dei metodi e delle tecniche che si usano nello sviluppo industriale del software.

> *“disciplina tecnologica e manageriale che riguarda la produzione sistematica e la manutenzione dei prodotti software, [. . .] sviluppati e modificati entro i tempi e i costi preventivati”.*
>
> *— D. Fairley, Software Engineering Concepts*

L’ingegneria del software studia anche gli aspetti sociali e organizzativi sia dell’ambiente in cui viene sviluppato il software, sia di quello in cui il software viene applicato.

Alcuni ruoli:

- **Sviluppatore**
- **Produttore**
- **Committente**
- **Utente**

OSS: L’utente può anche essere uno sviluppatore, come del caso dell’utilizzo di librerie.

Una specifica è una descrizione precisa dei requisiti (di un sistema o di una sua parte). Una specifica descrive una certa entità “dall’esterno”, cioè dice quali servizi devono essere forniti o quali proprietà devono essere esposte da tale entità.

La specifica può contenere anche il costo massimo per la sua realizzazione.

Una *specifica* descrive *che cosa* deve fare un sistema ma non come deve essere costruito affinché mostri determinate proprietà.

Un progetto invece descrive come deve essere realizzato un sistema.

Dal punto di vista del progettista, un requisito è un obbligo imposto dall’esterno mentre l’implementazione è il risultato di una serie di scelte.

Alcune caratteristiche del progetto invece possono essere imposti dall’esterno, quindi in questo caso si parla di vincoli.

Data la potenziale ambiguità nel ruolo (requisito o vincolo o scelta di progetto) di certi aspetti di un sistema, è importante che la documentazione prodotta durante il suo sviluppo identifichi tali ruoli chiaramente.

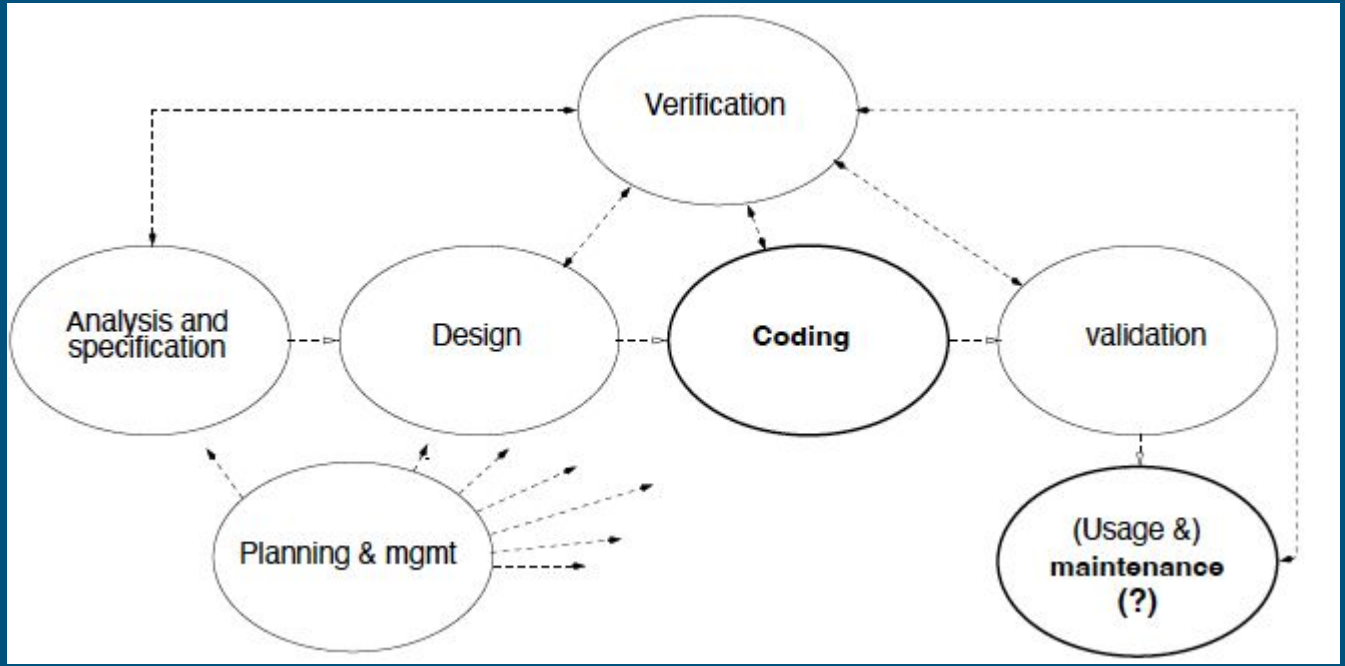
Ciclo Vita

Ogni prodotto industriale ha un ciclo di vita che, a grandi linee, inizia quando si manifesta la necessità di un nuovo prodotto e prosegue con l'identificazione dei suoi requisiti, il progetto, la produzione, la verifica, e la consegna al committente.

Dopo la consegna, il prodotto viene usato ed è quindi oggetto di manutenzione e assistenza tecnica, e infine termina il ciclo di vita col suo ritiro.

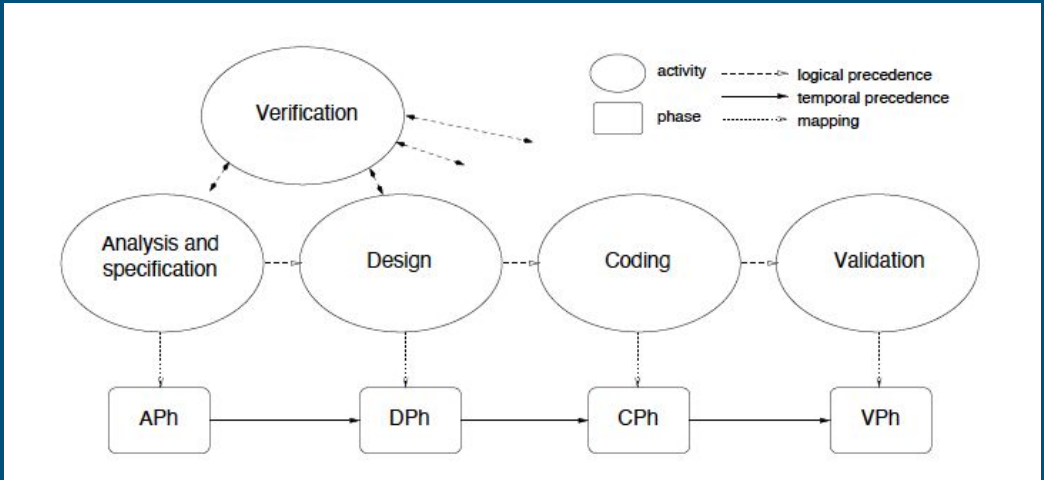
A queste attività se ne aggiungono altre, che spesso vi si sovrappongono, come la pianificazione e la gestione del processo di sviluppo, e la documentazione.

Ciclo vita di un Software



Modello a Cascata - Waterfall

Il modello a cascata **(waterfall)** prevede una successione di fasi consecutive. Ciascuna fase produce dei semilavorati (deliverable), i quali vengono ulteriormente elaborati dalle fasi successive.

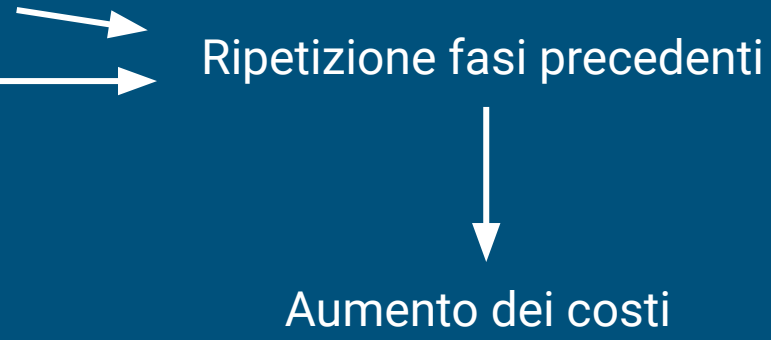


Presupposti :

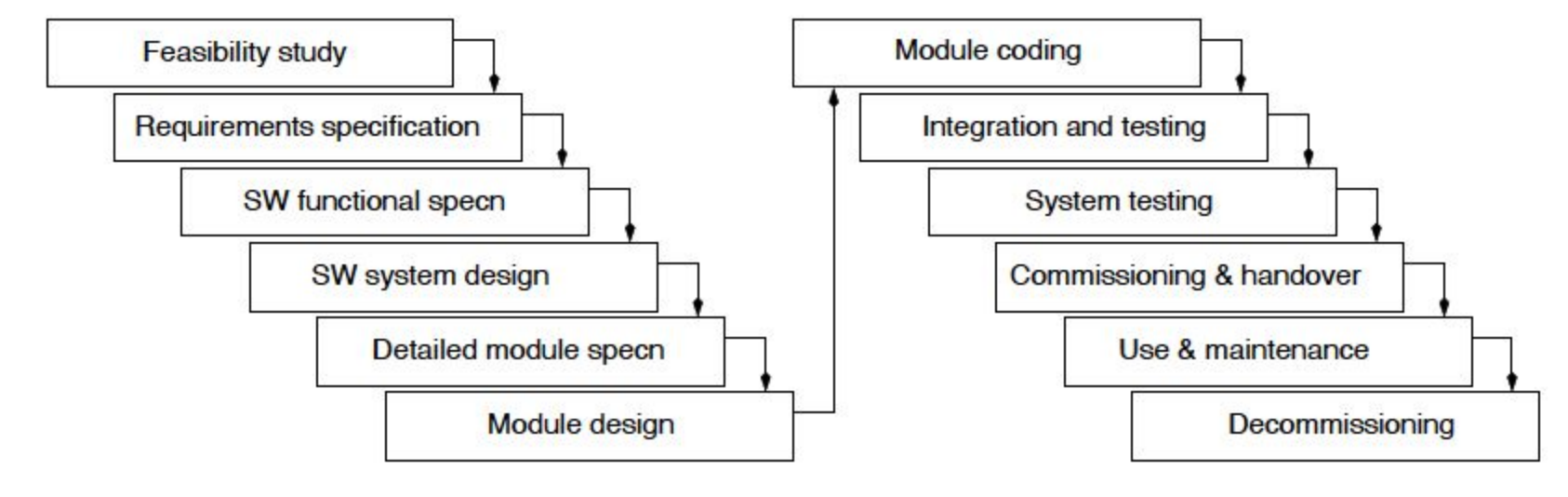
- ogni fase deve essere conclusa prima dell’inizio della fase successiva;
- requisiti iniziali devono essere chiari e dettagliati fin dall’inizio;
- sviluppo prevedibile del prodotto.

Problematiche:

- errori nel progetto o nelle specifiche
- introduzione di nuovi requisiti



Esempio modello a cascata



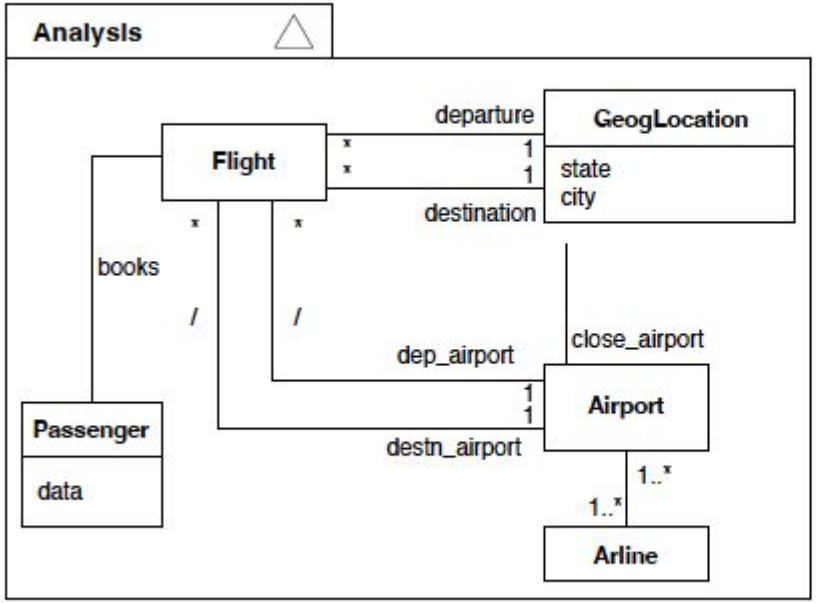
Fasi di sviluppo

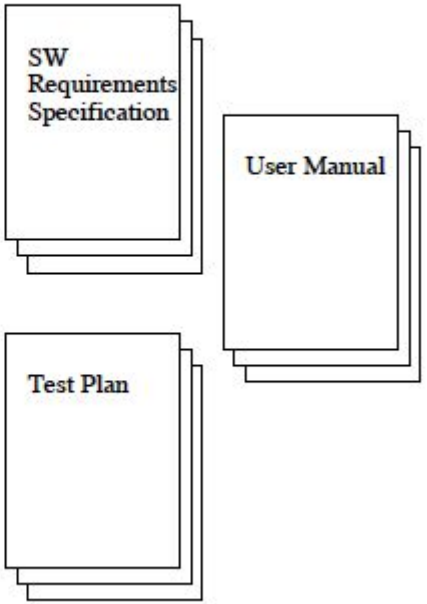
- studio di fattibilità;
- *analisi e specifica dei requisiti*, suddivisa in
  - analisi dei requisiti dell'utente;
  - specifica dei requisiti del software (**funzionali e non funzionali**)
- *progetto*, suddiviso in
  - progetto architetturale;
  - progetto in dettaglio;
- programmazione e test di unità;
- integrazione e test di sistema;
- manutenzione.

Attività di *supporto* svolte in contemporanea

- gestione;
- controllo di qualità;
- documentazione.

Esercizio aeroporto





Esercizio Desktop PC

Analisi dei requisiti dell'utente

Definizione dei requisiti dell'utente

1 L'applicazione deve permettere la rappresentazione e l'elaborazione di file creati da altre applicazioni (detti file esterni).

Specifica dei requisiti dell'utente

1.1 L'applicazione deve permettere all'utente di definire i tipi dei file esterni.
1.2 Ad ogni tipo di file esterno corrisponde un programma esterno ed opzionalmente un'icona che viene usata per rappresentare il file. Se al tipo di un file non è associata alcuna icona, viene usata un'icona default non associata ad alcun tipo.
1.3 L'applicazione deve permettere all'utente di definire l'icona associata ai tipi di file esterni.
1.4 La selezione di un'icona rappresentante un file esterno causa l'elaborazione del file rappresentato, per mezzo del programma associato al tipo del file stesso.

Specifica dei requisiti del software

1.1.1 L'utente può definire i tipi dei file esterni sia per mezzo di menú che di finestre di
dialogo. È opzionale la possibilità di definire i tipi dei file esterni per mezzo di file di
configurazione modificabili dall'utente.
1.2.1 L'utente può associare un programma esterno ad un tipo di file esterno sia per mezzo
di finestre di dialogo che di file di configurazione.
1.2.2 L'utente può associare un'icona ad un tipo di file esterno per mezzo di una finestra
di selezione grafica (chooser).
1.3.1 L'applicazione deve comprendere una libreria di icone già pronte ed uno strumento
grafico che permetta all'utente di crearne di nuove.
1.4.1 La selezione di un'icona rappresentante un file esterno può avvenire sia per mezzo
del mouse che della tastiera.

Progetto

In questa fase si stabilisce come deve essere fatto il sistema definito dai documenti di specifica. (DSP - Documento delle Specifiche di Progetto)

In generale, esistono diversi modi di realizzare un sistema che soddisfi un insieme di requisiti, l'attività del progettista consiste essenzialmente in una serie di scelte fra le soluzioni possibili, guidate da alcuni principi e criteri

Il risultato del progetto è una architettura software, cioè una scomposizione del sistema in elementi strutturali, detti moduli, dei quali vengono specificate le funzionalità e le relazioni reciproche.

La fase di progetto può essere suddivisa nelle sottofasi di progetto architetturale e di progetto in dettaglio.
La distinzione fra queste due sottofasi spesso non è netta specialmente nelle metodologie più moderne

Programmazione

In questa fase i singoli moduli definiti nella fase di progetto vengono implementati e collaudati singolarmente. Vengono scelte le strutture dati e gli algoritmi, che di solito non vengono specificati nel DSP.

Strumenti per un programmatore:

- gestione delle versioni
- configurazione e compilazione automatica (IDE)
- documentazione del codice
- notifica e archiviazione di malfunzionamenti
- test di unità

Integrazione e test di sistema

Nel corso dell'integrazione vengono assemblati i vari sottosistemi a partire dai moduli componenti, effettuando parallelamente il test di integrazione, che verifica la corretta interazione fra i moduli. Dopo che il sistema è stato assemblato completamente, viene eseguito il test di sistema

Manutenzione

La cosiddetta manutenzione del software è in realtà la correzione di errori presenti nel prodotto consegnato al committente, oppure l'aggiornamento del codice allo scopo di fornire nuove versioni

Modelli evolutivi

Pensati per far fronte alla necessità di cambiamenti in corso d'opera, affinché possano essere introdotti e sviluppati tempestivamente

OSS: In questi processi, il software viene prodotto in modo incrementale, in passi successivi. Ogni passo produce, a seconda delle varie strategie possibili, una parte nuova oppure una versione via via più raffinata e perfezionata del sistema complessivo.

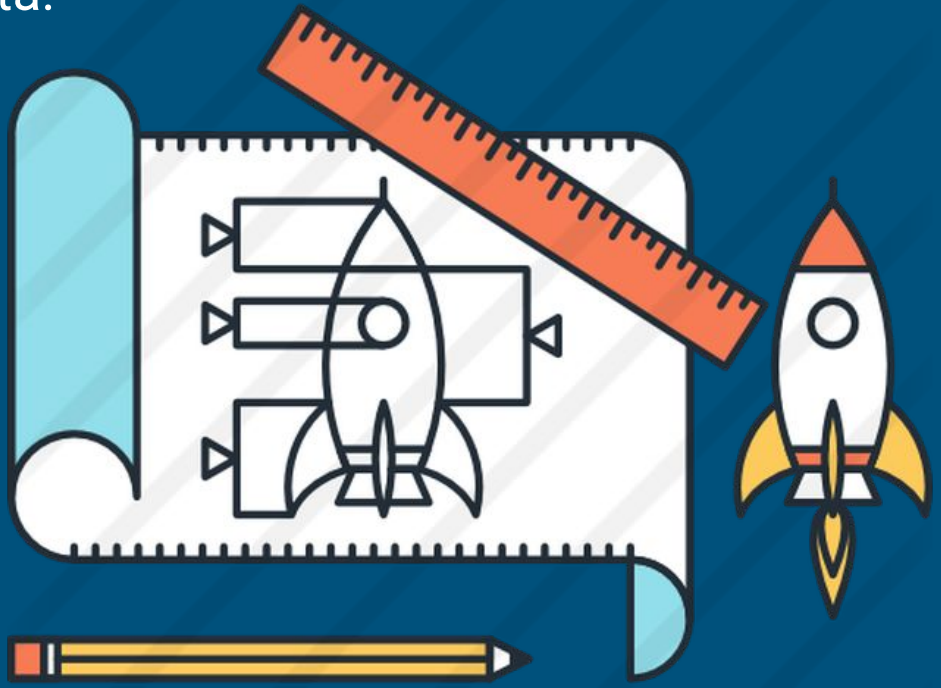
Il prodotto di ciascun passo viene valutato ed i risultati di questa valutazione determinano i passi successivi, finché non si arriva ad un sistema che risponda pienamente alle esigenze del committente, almeno finché non si sentirà il bisogno di una nuova versione.

Prototipazione

Un prototipo è una versione approssimata, parziale, ma funzionante, dell'applicazione che viene sviluppata.

Esistono varie tipologie di prototipi:

- esplorativo (throw-away);
- sperimentale;
- evolutivo;



Unified Process (UP)

Lo Unified Process è un processo evolutivo per lo sviluppo di software orientato agli oggetti, concepito dagli ideatori del linguaggio UML. Composizione

• processo di sviluppo suddiviso in quattro fasi successive;

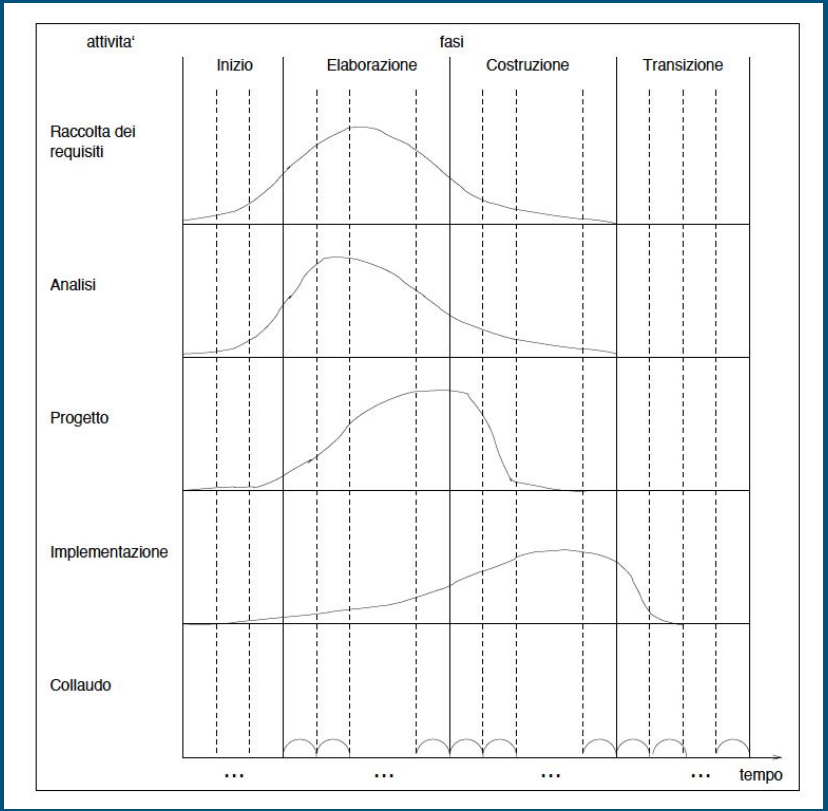
• ogni fase ha un obiettivo e produce un insieme di semilavorati chiamato milestone

• ogni fase è suddivisa in un numero variabile di iterazioni;

• nel corso di ciascuna iterazione possono essere svolte tutte le attività richieste (analisi, progetto. . .), anche se, a seconda della fase e degli obiettivi dell'iterazione, alcune attività possono essere predominanti ed altre possono mancare;

• ciascuna iterazione produce una versione provvisoria (baseline) del prodotto, insieme alla documentazione associata.

Fasi dello UP - workflow



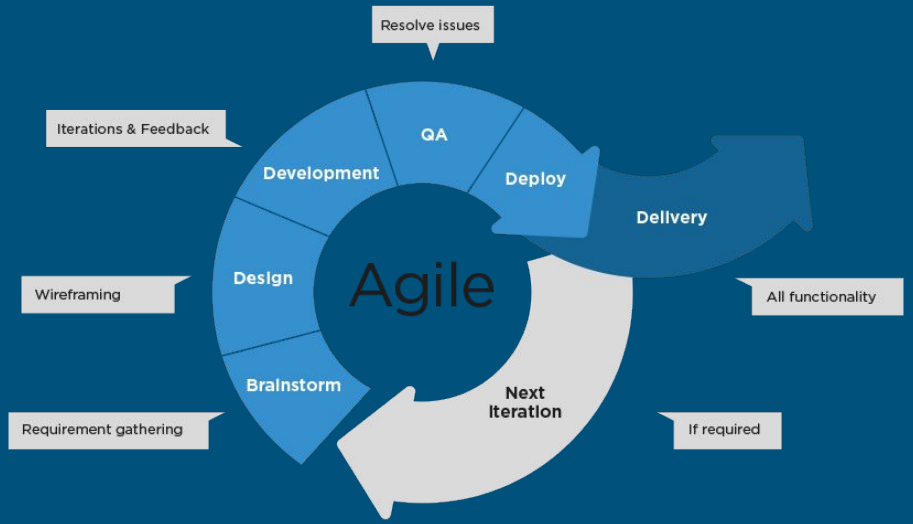
Nello UP si riconoscono cinque attività, dette workflow (o flussi di lavoro):
raccolta dei requisiti (requirement capture),
analisi (analysis),
progetto (design),
implementazione (implementation),
e collaudo (test).

Le prime due attività corrispondono all' analisi e specifica dei requisiti

Processi Agili

I processi agili sono una famiglia di processi evolutivi caratterizzati da iterazioni veloci, enfasi sulla produzione e collaudo di codice, e interazione col committente/utente.

Si contrappongono ai modelli Waterfall o altri modelli classici



Manifesto Agile - Principi Generali

(https://agilemanifesto.org/iso/it/manifesto.html)

Persone e interazioni - - - sono più importanti dei processi e degli strumenti;
Un software funzionante - - - è più importante della documentazione;
Collaborare con i clienti - - - è più importante del contratto;
Aderire ai cambiamenti - - - è più importante che aderire al progetto.

In senso lato il termine "agile" indica tutte quelle metodologie di sviluppo leggere e flessibili, che rompono con la precedente tradizione di ingegneria del software (modello a cascata, modello a spirale, etc.) basata su una raccolta delle specifiche e su una strutturazione sequenziale dello sviluppo software.

Esempi di metodologie e framework agili:

- Agile Unified Process,
- Crystal (famiglia),
- Extreme programming (XP),
- Scrum.

Risultati

• soddisfazione del cliente;
• accogliere favorevolmente i cambiamenti anche se s
è avanti con lo sviluppo;
• usare scale temporali brevi (feedback piú rapido
possibile);
• team che si autoregolano;
• lavorare insieme quotidianamente (giocare per vincere
=> Gamification);
• semplicità del progetto e software funzionante;
• iterazioni regolari e valutazione dell'attività;

Una motivazione importante per l'adozione di tecnologie agili è la rapidità con cui
permettono di scoprire difetti nel prodotto e nei semilavorati

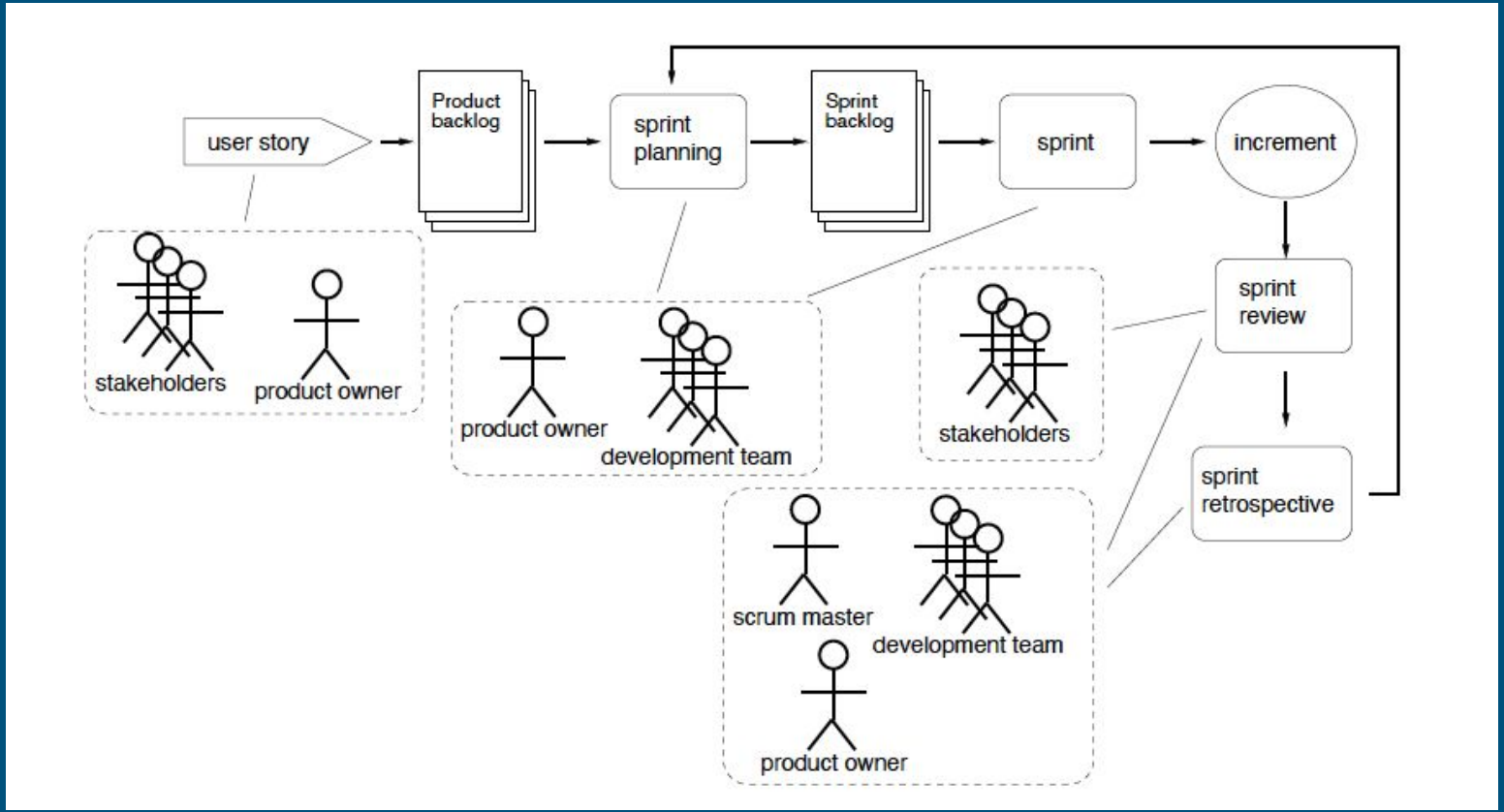
Figure Principali del Processo

Basato sul framework *Scrum* il processo prevede, dopo una fase iniziale, una serie di cicli chiamati *Sprint*.

I partecipanti sono:

- *stakeholder*: rappresentanti del committente
- *product owner*: rappresentante del committente che partecipa a tutte le attività insieme agli sviluppatori
- *scrum master*: collegamento tra il team e lo stakeholder

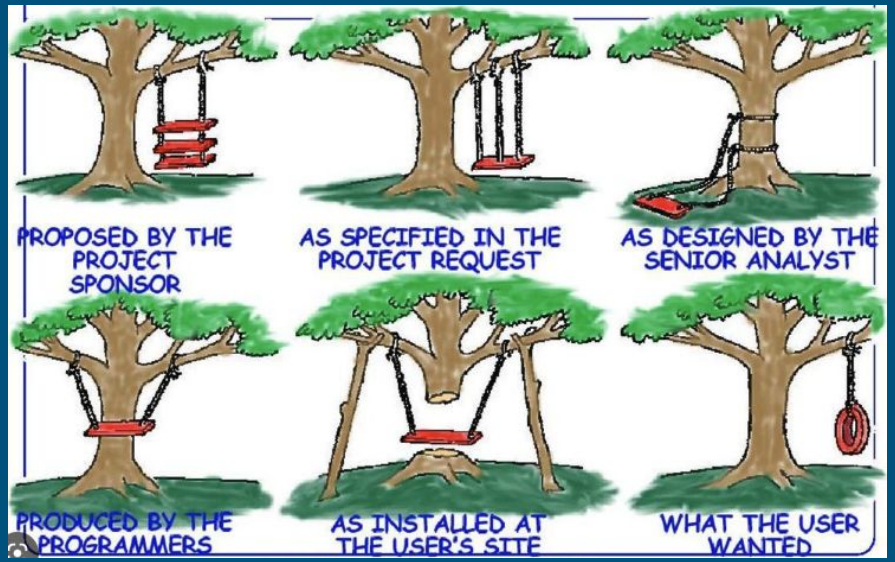
Momenti Salienti



Requisiti e Specifiche

I requisiti descrivono ciò che l'utente si aspetta dal sistema. I requisiti si distinguono in *funzionali e non funzionali* .

Il termine requisiti si riferisce ai comportamenti e proprietà del sistema come espressi nei documenti di specifica.



I requisiti

I requisiti *funzionali* descrivono cosa deve fare il sistema. Specificano le relazioni tra i dati in ingresso e i dati in uscita, oppure fra stimoli (ambiente -> sistema) e risposte.

I requisiti *non funzionali* esprimono dei vincoli o delle caratteristiche di qualità.

- **Sicurezza (*Safety*)**
- **Robustezza**
- **Riservatezza (*Security*)**
- **Prestazioni**
- **Disponibilità**
- **Usabilità**
- **Interoperabilità**
- **Tracciabilità e collaudabilità**

Classificazioni

- Requisiti Temporali legati alle prestazioni
  - sequenziali (*no vincoli di tempo*)
  - concorrenti (*processi sincronizzati*)
  - in tempo reale

- Tipo di Elaborazione
  - orientato ai dati
  - orientato alle funzioni
  - orientato al controllo

- Software di base o applicativo
  Applicazione Middleware o software applicativo