Il **prodotto software** è l'insieme di tutti gli artefatti che permettono l'utilizzo di un programma di un utente: codice, documentazione, prodotti intermedi quali casi di test, manuali tecnici ecc...

La **programmazione** è l'insieme di quelle attività che a partire da un problema concludono alla stesura di un programma la cui esecuzione da parte di un calcolatore ha come risultato la soluzione del problema dato.

Il primo problema è quello di capire il problema, il secondo è quello di progettare la soluzione del problema

Software:

- Intangibile
- Non definibile a priori
- Spesso costruito ad hoc
- Ad alta intensità di lavoro umano
- Soggetto al cambiamento

Seniority: frutto dell'esperienza maturata in campo ottenuta con il perfezionamento delle metodologie classiche

Ingegneria del software: L'istituzione e l'impiego di principi ingegneristici fondati, allo scopo di ottenere in modo economico software affidabili ed efficienti, su macchine vere

Motivazioni che portano all'utilizzo di metodologie dedicate:

- Affrontare lo sviluppo software in presenza di:
    - Requisiti non certi
    - Ambiente di sviluppo non certo
    - Ambiente di erogazione certo e non adattabile
    - Risorse umane creative e indipendenti
- Ridurre i costi di sviluppo
- Rispettare i tempi ed i costi di sviluppo
- Favorire il riuso dei componenti realizzati e della conoscenza acquisita
- Crescere assieme come team di sviluppo

Processo software: Insieme organizzato delle pratiche che sovrintendono alla costruzione del prodotto da parte del team di sviluppo utilizzando metodi, tecniche, metodologie e strumenti

Prodotto software può essere di due tipi:

- **Per un cliente** (ad hoc)
- **Per il mercato** (prodotto pacchettizzato [Commercial Off The Shelf])

La realizzazione di un progetto viene effettuata in un **arco temporale ben individuato**

I prodotti informatici possono essere suddivisi in quattro diverse tipologie:

- **Progetti commerciali**

- **Progetti di innovazione e investimento**
- **Progetti di miglioramento gestionale**
- **Progetti di riorganizzazione aziendale**

Applicazioni web: Progettazione basata su architettura Client-Server e adottano un modello strutturale su tre strati (three-tier):

- **Tier di presentazione**
- **Tier dell'applicazione**
- **Tier dei dati**

I vantaggi di questo tipo di struttura sono i seguenti:

- **Velocità di sviluppo elevata**
- **Miglior scalabilità**
- **Migliore sicurezza**

Ciclo di vita del software

Ciclo di vita del software: insieme di tutte le attività connesse alla produzione di un programma

In un qualsiasi progetto software le fasi di sviluppo possono essere ulteriormente suddivise in sottofasi:

1. **Pianificazione del sistema**: identifica la relazione tra l'ambiente ed il sistema individuando le parti software da realizzare
2. **Analisi**: Si decide che cosa il progetto dovrebbe realizzare senza alcun riferimento a come il programma realizzerà i suoi obiettivi. Viene prodotto un documento noto come specifiche dei requisiti e in questo documento sono presenti anche altre due parti:
    a. Il manuale per l'utente
    b. Le prestazioni richieste al sistema e le risorse utilizzate
3. **Progettazione**: incorpora tutte le attività connesse all'ideazione e alla definizione della strategia che porta alla soluzione nel modo più efficace ed efficiente. Si individuano i due obiettivi seguenti:
    a. Sviluppare un piano di realizzazione del sistema
    b. Si individuano le strutture dati e archivi necessari alla soluzione del problema, in caso di OOP si parlerà di classi e metodi
4. **Codifica**: si scrive e compila il codice sorgente eseguendo la codifica delle istruzioni in linguaggi di programmazione in due fasi:
    a. Si traducono in programma le informazioni ottenute dalla formalizzazione (requisiti)
    b. Il programma viene tradotto in linguaggio macchina
5. **Test e debug**: viene verificata la correttezza dei risultati tramite dati campione e l'assenza di errori (test) la cui eventuale presenza innesca la fase di debug dove si va a correggere l'errore
6. **Installazione, verifica e collaudo**:

    a. **Installazione**: consegna del software al cliente mediante installazione fisica sul sistema del cliente stesso
    b. **Verifica**: si verifica che il software sia corretto rispetto alle specifiche individuate dall'analista e quindi conforme a quanto specificato nelle richieste
    c. **Collaudo**: si verifica che eseguendo le prove con dati reali il programma funzioni correttamente
7. **Manutenzione**: fase permanente di supporto al sistema dopo la consegna all'utente che si concentra in tre aspetti:
    a. Correzione ed eliminazione ulteriori errori identificati
    b. Apportare ulteriori modifiche necessarie al software per poter essere utilizzato su altri sistemi informatici
    c. Aggiungere nuove funzionalità o migliorare quelle esistenti

Ciclo gestionale

Si differenzia dal ciclo di vita del software ed è composto da tre fasi:

- **Avvio**
- **Pianificazione**
- **Esecuzione**

Questa logica iterativa permette di avere una gestione più efficace in quanto le informazioni per pianificare una fase sono disponibili solamente alla fine della fase che la precede limitando la propagazione di errori

Modelli di sviluppo

Per esplorazione

I progettisti/lavorano con i clienti dall'inizio alla fine con il cliente che man mano che il software prende vita aggiunge successive richieste

Build and Fix

Si affronta il progetto senza alcuna documentazione realizzando un prodotto approssimativo che viene presentato al cliente e viene modificato in base alle obiezioni mosse da quest'ultimo

A cascata (Waterfall)

Ogni singola attività viene completata prima dell'attività successiva. Il modello di realizzazione viene scomposto in varie fasi che si susseguono l'un l'altra e ogni singola fase viene eseguita solamente se le fasi precedenti sono state completate correttamente. Si può schematizzare nel seguente modo:

- **Studio di fattibilità**
- **Analisi e specifica dei requisiti**
- **Progettazione**
- **Codifica delle singole unità software**

- **Programmazione e test delle unità**
- **Installazione**
- **Manutenzione**

Il problema principale di questo modello di sviluppo è legato al come ci si renda conto spesso troppo tardi che in alcune situazioni sarebbe necessario a tornare alla fase precedente in cui è stato trascurato qualche elemento prima ritenuto superfluo ed ora rivelatosi indispensabile per procedere con il progetto

Modello a prototipazione rapida-RAD

Questo modello produttivo è caratterizzato da un **processo iterativo** che raffina di volta in volta il risultato fino al raggiungimento di un prodotto che soddisfa sia l'utente finale che lo sviluppatore. Si inizia con la raccolta dei requisiti poi successivamente si crea un prototipo che viene consegnato all'utente finale e sulla base delle considerazioni espresse da quest'ultimo si itera sul prototipo fino al punto da arrivare ad un prodotto che soddisfi entrambe le parti.

I problemi che si possono riscontrare sono i seguenti:

1. La **valutazione del prototipo** spesso non riguarda gli aspetti che il prototipo voleva evidenziare ma altre caratteristiche di seconda importanza lontane dallo scopo del prototipo
2. Il prototipo viene utilizzato come **base di sviluppo** e non come "chiarificatore dei requisiti"

Modello incrementale

È considerabile come la fusione dei due modelli precedenti. Si segue il modello a cascata dove una o più fasi sono eseguite per incrementi successivi al fine di creare versioni successive dello stesso prodotto, ogni versione soddisfa una parte dei requisiti (sottosistema) implementando prima i quelli critici fino a quelli marginali. Ogni sottosistema soddisfa solo una parte dei requisiti, infatti vengono realizzati in modo incrementale (una parte per volta di tutto il sistema complessivo).

Le conseguenze sono che per ogni sottosistema si vanno ad applicare la fase di codifica e di test poi l'integrazione dei vari sottosistemi e un test complessivo del sistema intero che rappresentano i problemi a cui è soggetto questo tipo di modello. Non presenta una fase di manutenzione perché l'aggiunta di un incremento a un sottosistema corrisponde ad una manutenzione e dunque le modifiche possono essere facilmente implementabili.

Modello a spirale

Nasce per ovviare ai modelli precedenti nel 1988 e scompone il processo di sviluppo in quattro fasi multiple ciascuna ripetuta più volte. Sono le stesse del modello a cascata ma con tempi più ristretti e dalla fase di testing si torna a quella di pianificazione per applicare le dovute correzioni al risultato dello sviluppo. Ogni iterazione è un modello a cascata ripetuto che procede per affinazioni dove viene pianificato il numero di iterazioni. Le quattro fasi sono le seguenti:

1. **Pianificazione**: si determinano obiettivi, vincoli e alternative. Committente e sviluppatori interagiscono allo scopo di definire in maniera univoca cosa deve essere realizzato e come

2. **Analisi dei rischi**: si identificano ed analizzano i problemi ed i rischi legati al progetto al fine di elaborare strategie per controllarli. Rientrano tempi, costi e variazione delle specifiche
3. **Sviluppo**: si procede alla fase vera e propria della realizzazione ed i tempi per quest'ultima sono tra i più lunghi presenti all'interno del ciclo di vita del prodotto software
4. **Valutazione**: il committente valuta se il prodotto soddisfa le sue esigenze o ha bisogno di definire ulteriori requisiti non realizzati (conseguenza di un prodotto che non supera la fase di validazione)

Limiti dei processi di sviluppo tradizionali

Essendo modelli degli anni Settanta ad oggi risultano inefficaci perché ad oggi ci sono tecnologie completamente diverse e quindi i limiti sono i seguenti:

- **Fase di test tardiva**: è rischioso fare i test alla fine dopo aver scritto migliaia di righe di codice ed effettuare ciò poco prima della messa in produzione
- **Burocrazia**: la specifica dei requisiti vincola il prodotto da sviluppare e ciò non sempre soddisfa le esigenze del cliente
- **Attività svolte per obbligo**: nessuno sviluppatore svolge volentieri le attività burocratiche, test e scrittura della documentazione
- **Regressione dell'errore**: introduzione di modifiche al termine dello sviluppo che portano alla comparsa di ulteriori anche nelle parti non direttamente collegate alla modifica effettuata
- **Linearità**: La correzione dell'errore viene fatta a ritroso percorrendo linearmente tutto il ciclo di sviluppo
- **Rigidità**: nei modelli a cascata ogni passaggio di fase è ben definito e dopo la parte iniziale il cliente non viene coinvolto più fino alla fine e ciò può portare alla scoperta di errori e incomprensioni solo a progetto ultimato

Nuovi modelli di sviluppo

Sviluppo "agile" o iterativo incrementale

Sono metodologie di sviluppo definite "leggere" in contrapposizione a quelle pesanti (waterfall, iterativa). Si focalizzano sul raggiungimento di un risultato alla volta, piccolo e ben definito costruendo con un processo iterativo il sistema completo.

La loro caratteristica non è di essere predittive come quelle vecchie ma di essere adattive, ovvero di adattarsi all'evoluzione dei requisiti utente prima che del software in sé per sé.

Si basano su fasi di progettazione, sviluppo e test molto più brevi e le caratteristiche principali sono le seguenti:

- **Iteratività e incrementali**: le fasi elencate precedentemente avvengono in tempi molto più ridotti concentrandosi nella risoluzione dei pochi problemi, piccoli e ben definiti.

- **Rilasci frequenti**: procedendo a piccoli passi i rilasci vengono prodotti in modo molto più rapido
- **Testing**: Si applica al codice ed ai dati ed è una delle pietre miliari delle metodologie agili

Non sono metodologie sempre applicabili a tutti i progetti, infatti sono indicate in quei progetti dove i requisiti utente cambiano continuamente e dove il team di sviluppo segue praticamente in tempo reale la loro evoluzione. La metodologia agile per eccellenza è XP (eXtreme Programming)

XP (eXtreme Programming)

Definisce le best practice per sviluppare un software nelle condizioni ottimali ed alla base ci sono le seguenti regole:

- **Pianificazione realistica**: i clienti decidono le funzionalità, i tecnici le decisioni tecniche aggiornando il piano di sviluppo se sono in conflitto con la realtà
- **Piccoli stati di avanzamento**: fornire velocemente un sistema utilizzabile e fornire aggiornamenti in tempi brevi
- **Metafora**: Tutti i programmatori dovrebbero condividere un racconto che illustri il sistema in fase di realizzazione
- **Semplicità**: Progettare ogni cosa in modo che sia la più semplice possibile
- **Collaudo**: Sia i programmatori che i clienti devono preparare casi di prova, il sistema deve essere collaudato continuamente
- **Riprogettazione**: i programmatori devono continuamente ristrutturare il sistema per migliorare il codice su unico calcolatore
- **Programmazione a coppie**: i programmatori devono lavorare a coppie su un unico calcolatore
- **Proprietà collettiva**: tutti i programmatori devono poter modificare porzioni di codice quando necessario
- **Integrazione continua**: non appena un problema è risolto, bisogna mettere insieme l´intero sistema e collaudarlo
- **Settimana di 40 ore**: Usare piani di lavoro realistici
- **Cliente a disposizione**: Un vero utilizzatore deve essere sempre a disposizione della squadra di progettazione
- **Standard per la scrittura del codice**: I programmatori devono utilizzare standard di codifica che si concentrano sul codice auto documentato

Alcuni di questi criteri arrivano dal buon senso altri sono sorprendenti: Beck (il creatore di ciò) afferma che è la sinergia tra queste regole a definire la potenza di XP

RUP (Rational Unified Process)

È un metodo iterativo che può essere utilizzato come metodologia agile e specifica soprattutto la composizione dei team di sviluppo e il calendario delle fasi, nonché alcuni modelli di documenti. Si struttura in 5 fasi (**avvio**, **elaborazione**, **costruzione**, **transizione** e **produzione**) con iterazioni che hanno durata da 2 a 6 settimane