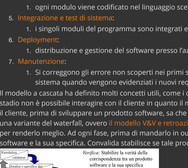
Con il termine "Software Life Cycle" si intende il periodo di tempo che inizia quando viene concepito un prodotto software e termina quando il prodotto non è più disponibile per l'uso. Il ciclo di vita del software comprende in genere una fase concettuale, fase dei requisiti, fase di progettazione, fase di implementazione, fase di test, fase di installazione e verifica, fase di funzionamento e manutenzione, e fase di pensionamento, che è in contrasto con il termine "Software Development Cycle".
Quest'ultimo sarebbe il periodo di tempo che inizia con la decisione di sviluppare un prodotto software e termina alla consegna del prodotto. Questo ciclo include in genere una fase dei requisiti, una fase di progettazione, una fase di implementazione, una fase di test e una fase di installazione e verifica.
Un modello del ciclo di vita del software rappresenta la scomposizione dell'attività in tanti sottoattività tra loro coordinate, il cui risultato finale è la realizzazione del sistema software stesso e tutta la documentazione a esso associata. Esso consentono agli sviluppatori di software di gestire la complessità dei sistemi software.
Le fasi di un CV sono:

1. Definizione: si occupa del cosa
    1. Determinazione dei requisiti, quali funzioni vuole il cliente stesso, interfacce
2. Sviluppo: si occupa del come
    1. Definizione del progetto, dell'architettura software, della strutturazione dei dati e delle interfacce, che tipo di linguaggio di programmazione.
3. Manutenzione: si occupa delle modifiche
    1. correzioni, adattamenti, miglioramenti, prevenzione

Ci sono vari modelli di ciclo di vita che al giorno d'oggi vengono usati, come il modello a cascata, il modello a V, ecc..
Cominciamo con il modello a cascata (Waterfall):
Prevede l'esecuzione lineare di una precisa sequenza di fasi, ciascuna delle quali genera un output utilizzato come input dalla fase successiva (da qui l'origine del termine "a cascata"). In linea di principio, occorre pianificare tutte le attività del processo prima di iniziare lo sviluppo del software. Ogni output che vengono prodotti ad ogni stadio per poi passarlo al prossimo stadio in forma di input, sono dei documenti approvati con su scritti delle varie descrizione dell'andamento. Quindi finché non viene prodotto un documento in uno stadio, non si passa alla fase successiva. Inoltre, i prodotti di uno stadio vengono "congelati", ovvero non sono più modificabili se non irrevocando un processo di modifica..
I principali stadi del modello a cascata riflettono direttamente le attività di sviluppo fondamentali del software:

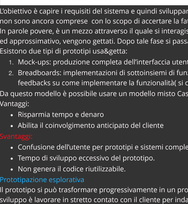
1. Studio di fattibilità:
    1. Stabilire se avviare il progetto, individuare le possibili opzioni e le scelte più adeguate, valutare le risorse umane e finanziare necessarie. Alla fine di questo stadio, viene prodotto un "documento di fattibilità" dove vi è una definizione preliminare del problema, scenari, costi, tempi e modalità di sviluppo per ogni alternativa.
2. Analisi dei requisiti:
    1. Si determinano i servizi del sistema, i suoi vincoli e obiettivi attraverso incontri con gli utenti del sistema. Si preoccupa nel cosa rappresentare all'interno di questo sistema. Viene prodotto un documento di specifica dei requisiti ed un manuale utente.
3. Progettazione del sistema e del software:
    1. Suddividie il sistema in componenti e moduli e stabilisce l'architettura generale del sistema. Si preoccupa nel come implementare i vari moduli. Si distingue in due parti: architectural design che definisce la struttura complessiva e detailed design che definisce dei dettagli interni a ciascun componente. Definisce anche le relazioni fra i moduli. Viene prodotto un documento di specifica di progetto dove vi è scritto l'uso di linguaggio per la progettazione,ecc..
4. Implementazione e test di unità:
    1. ogni modulo viene codificato nel linguaggio scelto e testato in isolamento
5. Integrazione e test di sistema:
    1. i singoli moduli del programma sono integrati e testati come un sistema completo per accertare che i requisiti del software siano soddisfatti.
6. Deployment:
    1. distribuzione e gestione del software presso l'azienda
7. Manutenzione:
    1. Si correggono gli errore non scoperti nei primi stadi del ciclo di vita, si migliora l'implementazione delle unità di sistema e si incrementano i servizi del sistema quando vengono evidenziati i nuovi requisiti.

Il modello a cascata ha definito molti concetti utili, come i documenti prodotti dai vari stadi ma rappresenta dei svantaggi, in particolar modo durante una fase di uno stadio non è possibile interagire con il cliente in quanto il modello stesso prevede la comunicazione soltanto all'inizio e alla fine di una fase. Ma non solo, non sempre il cliente, prima di sviluppare un prodotto software, sa che funzionalità vuole nel suo sistema creando di fatto dei requisiti spesso imprecisi. Per ovviare a ciò, è nato una variante del waterfall, ovvero il modello V&V e retroazione dove ad ogni fase, nel caso in cui i requisiti sono imprecisi è possibile tornare indietro e modificarlo per renderlo meglio. Ad ogni fase, prima di mandarlo in output, viene fatta una verifica e convalida. Verifica stabilisce la verità della corrispondenza tra un prodotto software e la sua specifica. Convalida stabilisce se tale prodotto esegue delle funzionalità previste alla sua missione operativa.

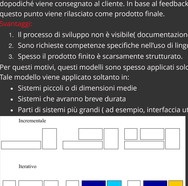


Modello a V

Il modello a V rappresenta un'estensione del modello a cascata quindi con la caratteristica che i semilavorati all'uscita di un fase vengono congelati e non più modificabili. Invece di scendere verso il basso in maniera lineare, le fasi del processo sono piegate verso l'alto per formare la tipica forma a V. Esso illustra le relazioni tra ogni fase del ciclo di vita del software e la fase di testing ad essa associata. Il ramo discendete rappresenta la fase di definizione; il vertice della V rappresenta la fase di implementazione del codice, il ramo ascendente della V rappresenta la fase di testing e di integrazioni. Quindi se si trova un errore in una fase del ramo discendente, si esegue la fase del ramo ascendete a esso collegato.



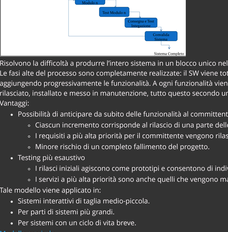
Modelli Evolutivi



Basati sull'idea di sviluppare un primo prototipo da sottoporre al committente e da raffinare successivamente.
E' un modello costituito da poche fasi che si ripetono:

1. realizzazione di un artefatto.
2. consegna al cliente.
3. ottenere dei feedback.
4. modificare il progetto in base a tali feedback.

Prototipazione Usa e getta(throw-away—>prototipo "cestinabile")



L'obiettivo è capire i requisiti del sistema e quindi sviluppare una definizione migliore dei requisiti. Si realizza un prototipo che sperimenta le parti del sistema che non sono ancora comprese con lo scopo di accertare la fattibilità del prodotto ed validare i requisiti.
In parole povere, è un mezzo attraverso il quale si interagisce con il cliente per accertarsi di aver bene compreso le sue richieste. Essendo un prototipo incompleto ed approssimativo, vengono gettati. Dopo tale fase si passa alla produzione della versione definitiva del SW mediante un modello che, in generale, è di tipo waterfall.
Esistono due tipi di prototipi usa&getta:

1. Mock-ups: produzione completa dell'interfaccia utente. Consente di definire con completezza e senza ambiguità i requisiti.
2. Breadboards: implementazioni di sottoinsiemi di funzionalità critiche del software, per verificare vincoli e fattibilità senza le interfacce utente. Produce feedbacks su come implementare la funzionalità(si cerca di conoscere prima di garantire)

Da questo modello è possibile usare un modello misto Cascata/Prototipi dove l'obiettivo è lavorare con i clienti ed evolvere i prototipi verso il sistema finale.
Vantaggi:

- Risparmia tempo e denaro
- Abilita il coinvolgimento anticipato del cliente

Svantaggi:

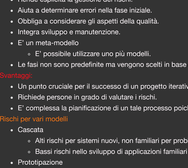
- Confusione dell'utente per prototipi e sistemi completati.
- Tempo di sviluppo eccessivo del prototipo.
- Non genera il codice riutilizzabile.

Prototipazione esplorativa
Il prototipo si può trasformare progressivamente in un prodotto quindi una serie di prototipi sviluppati con l'intenzione di migliorare. L'obiettivo del processo di sviluppo è lavorare in stretto contato con il cliente per indagarne i requisiti e giungere ad un "prodotto finale". Viene sviluppato la parte comprensibile del sistema, dopodiché viene consegnato al cliente. In base al feedback ricevuto, si continua a sviluppare prototipi fin a quando il cliente non è soddisfatto per il prototipo. A questo punto viene rilasciato come prodotto finale.
Svantaggi:

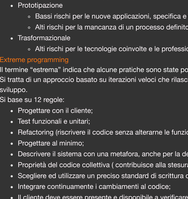
1. Il processo di sviluppo non è visibile(documentazione non disponibile). Il sistema sviluppato è poco strutturato(modifiche frequenti).
2. Sono richieste competenze specifiche nell'uso di linguaggi di prototipazione rapida.
3. Spesso il prodotto finito è scarsamente strutturato.

Per questi motivi, questi modelli sono spesso applicati solo in fase prototipiale, per giungere in tempi brevi ad un primo prodotto e validare le specifiche.
Tale modello viene applicato soltanto in:

- Sistemi piccoli o di dimensioni medie
- Sistemi che avranno breve durata
- Parti di sistemi più grandi (ad esempio, interfaccia utente)



Modello Incrementale



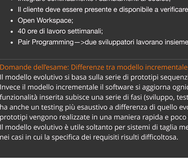
Risolvono la difficoltà a produrre l'intero sistema in un blocco unico nel caso di grandi progetti SW.
Le fasi alte del processo sono completamente realizzate: il SW viene totalmente definito nei requisiti, specificato e progettato. Quindi il software viene sviluppato aggiungendo progressivamente le funzionalità. A ogni funzionalità viene dedicato un ciclo di vita del software esclusi gli alti livelli: viene implementato, testato, rilasciato, installato e messo in manutenzione, tutto questo secondo un piano di priorità in tempi diversi.
Vantaggi:

- Possibilità di anticipare da subito delle funzionalità al committente.
    - Ciascun incremento corrisponde al rilascio di una parte delle funzionalità.
    - I requisiti a più alta priorità per il committente vengono rilasciati per prima.
    - Minore rischio di un completo fallimento del progetto.
- Testing più esaustivo
    - I rilasci iniziali agiscono come prototipi e consentono di individuare i requisiti per i successivi incrementi.
    - I servizi a più alta priorità sono anche quelli che vengono maggiormente testati.

Tale modello viene applicato in:

- Sistemi interattivi di taglia medio-piccola.
- Per parti di sistemi più grandi.
- Per sistemi con un ciclo di vita breve.

Modello a spirale



L'obiettivo principale è quello di analizzare il rischio. Le fasi del modello a spirale includono determinazione dei obiettivi, analisi del rischio, pianificazione e sviluppo&verifica. Il progetto passa continuamente attraverso queste fasi interazioni chiamati spirali.
All'inizio di un progetto di sviluppo software , i rischi sono tipicamente molto elevati poiché il rischio è collegato alla quantità e qualità delle informazioni disponibili. La chiarezza sui requisiti, le scelte sulla tecnologie e la strutturazione del sistema sono ipotesi non ancora consolidate; in alcuni casi, sono state scelte tecnologie innovative, per la quali manca una sufficiente esperienza nel gruppo di progetto. Di conseguenza meno informazioni si ha più alto sono i rischi;
Ogni iterazione ha lo scopo di ridurre i rischi di progetto.
Inizialmente, vengono creati dei prototipi di interazione per affrontare i rischi legati all'incertezza sui requisiti e prototipi architetturali per affrontare i rischi legati alla scelta delle tecnologie ed i dubbi sulla strutturazione del sistema. Successivamente, quando i rischi principali sono stati messi sotto controllo, ogni iterazione ha lo scopo di costruire nuove porzioni del sistema, via via integrate con le precedenti, e di verificarle con il committente e le altri parte interessate.
Esso è adatto per progetti grandi e complessi poiché fornisce un maggiore controllo verso tutte le fasi di sviluppi.
Vantaggi:

- Rende esplicita la gestione dei rischi.
- Aiuta a determinare errori nella fase iniziale.
- Obbliga a considerare gli aspetti della qualità.
- Integra sviluppo e manutenzione.
- E' un meta-modello
    - E' possibile utilizzare uno più modelli.
- Le fasi non sono predefinite ma vengono scelti in base ai tipo di prodotti.

Svantaggi:

- Un punto cruciale per il successo di un progetto iterativo è la collaborazione sistematica tra committenti e gruppo di progetto.
- Richiede persone in grado di valutare i rischi.
- E' complessa la pianificazione di un tale processo poiché richiede un controllo sistematico degli avanzamenti.

Rischi per vari modelli

- Cascata
    - Alti rischi per sistemi nuovi, non familiari per problemi di specifica e progetto.
    - Bassi rischi nello sviluppo di applicazioni familiari con tecnologie note.
- Prototipazione
    - Bassi rischi per le nuove applicazioni, specifica e sviluppo vanno di pari passo.
    - Alti rischi per la mancanza di un processo definito e visibile
- Trasformazionale
    - Alti rischi per le tecnologie coinvolte e le professionalità richieste

Extreme programming
Il termine "estrema" indica che alcune pratiche sono state portate all'estremo: così per esempio invece che scrivere Unit Test di tanto in tanto, si scrivono sempre.
Si tratta di un approccio basato su iterazioni veloci che rilasciano piccoli incrementi delle funzionalità. Si ha una partecipazione più attiva del committente al team di sviluppo.
Si base su 12 regole:

- Progettare con il cliente;
- Test funzionali e unitari;
- Refactoring (riscrivere il codice senza alterarne le funzionalità esterne);
- Progettare al minimo;
- Descrivere il sistema con una metafora, anche per la descrizione formale;
- Proprietà del codice collettiva (contribuisce alla stesura chiunque sia coinvolto nel progetto);
- Scegliere ed utilizzare un preciso standard di scrittura del codice;
- Integrare continuamente i cambiamenti al codice;
- Il cliente deve essere presente e disponibile a verificare;
- Open Workspace;
- 40 ore di lavoro settimanali;
- Pair Programming —>due sviluppatori lavorano insieme alla scrittura del codice:uno assume il ruolo i guidatore e l'altro assume il ruolo di osservatore.

Domande dell'esame: Differenze tra modello incrementale e modello evolutivo?
Il modello evolutivo si basa sulla serie di prototipi sequenziali fatti in maniera rapida, così da poterli migliorare ad ogni feedback ricevuto dal cliente.
Invece il modello incrementale il software si aggiorna ogniqualvolta gli viene aggiunto una funzionalità nuova seguendo il criterio del ciclo di vita ovvero ad ogni funzionalità inserita subisce una serie di fasi (sviluppo, test, consegna, convalida) con lo scopo di raffinarlo. Essendo che le fasi alte del processo sono completati, si ha anche un testing più esaustivo a differenza di quello evolutivo perché le parti alte del processo non sono realizzati in un maniera tanto ottimali essendo che i prototipi vengono realizzate in una maniera rapida e poco strutturata.
Il modello evolutivo è utile soltanto per sistemi di taglia medio-bassa, per sistemi con un ciclo di vita breve perché lo fa inMentre il modello incrementale è indicato nei casi in cui la specifica dei requisiti risulti difficoltosa.