Ingegneria del Software

L’essenza dell’ingegneria del software deve rispondere a queste sei domande:

1. Quale è il problema?
2. Quali caratteristiche dell’entità vengono utilizzate per risolvere il problema?
3. In che modo l’oggetto (e quindi la soluzione) sarà realizzata?
4. In che modo sarà costruito?
5. In che modo si condurrà la ricerca degli errori compiuti nella progettazione e costruzione dell’oggetto?
6. Quale supporto di sviluppo si darà all’oggetto a lungo termine, quando i suoi utenti richiederanno correzioni, adattamenti e migliorie?

Possiamo dire che queste domande sono il fulcro di tutto il lavoro dell’ingegnere. Così il processo di sviluppo software (risposta a tutte le domande precedenti) si divide in 3 blocchi di operazioni:

1. **La fase di definizione** - Si occupa del “cosa” (rispondendo alla domanda 1). Lo sviluppatore cerca di determinare quali siano le informazioni da elaborare, quali siano le funzioni e prestazioni attesa, quali debbano essere il comportamento del sistema, quali interfacce si debbano definire, quali siano i vincoli progettuali e i criteri di validazione in base ai quali valutare la riuscita del risultato. Si devono determinare i requisiti fondamentali del sistema e del software.

Le attività principali sono:

1. La strutturazione del sistema e delle informazioni (capitolo 10)
2. La pianificazione del progetto (capitolo 3, 5, 6 e 7)
3. L’analisi dei requisiti (capitolo 11, 12, 21)
4. **La fase di sviluppo** – Si occupa del come (rispondendo alle domande 2, 3, 4 e 5). In questa fase si definiscono la struttura dati, come implementare le funzioni entro una data architettura software, come implementare i dettagli procedurali, come strutturare le interfacce, come tradurre il progetto complessivo in un linguaggio di programmazione e come svolgere i collaudi.

Le attività tecniche specifiche sono:

1. Progettazione del software (capitolo 13, 16, 22)
2. La generazione del codice
3. Collaudo del software (capitolo 17, 18, 23)
4. **La fase di supporto** – Si occupa delle modifiche (rispondendo alla domanda 6) legate alla correzione degli errori, agli adattamenti necessari per l’evoluzione dell’ambiente del software e modifiche tese a soddisfare le nuove esigenze della clientela si applicano le fasi precedenti al contesto software.

Si individualo quattro tipi di modifiche:

1. Correzioni – La manutenzione correttiva modifica il software per eliminare i difetti.
2. Adattamenti – La modifica adattativa modifica il software per adattarlo al nuovo ambiente (cambi di CPU – Sistema Operativo – regole aziendali – caratteristiche del prodotto – ecc.)
3. Migliorie – La modifica perfettiva estende il software oltre i requisiti funzionali originali.
4. Prevenzione – La manutenzione preventiva o software reengineering, sono modifiche orientate alla semplificazione di correzioni, adattamenti e migliorie.

Da quanto sopra si deduce:

PROGETTAZIONE

ANALISI E MANUTENZIONE

CODICE

Possiamo adottare un ulteriore raffinamento per semplificare ulteriormente le cose

PROBLEMA

ANALISI (1) Comprendere il problema

ALGORITMO (2) Comprendere le soluzioni – Idea risolutrice

FORMALIZZAZIONE

SPECIFICAZIONI (3) Formalizzazione dell’algoritmo

PROGRAMMAZIONE

PROGRAMMA (4) Algoritmo tradotto in un linguaggio di alto livello

TRADUZIONE

ESEGUIBILE (5) Traduzione in linguaggio macchina

ESECUZIONE

RISULTATI (6) Valutazione dei risultati

MANUTENZIONE

ANALISI

-

-

- Modelli

-

PROGETTAZIONE

-

-

-

-

-

CODICE

-

-

-

-

MANUTENZIONE

-

-

-

-

MODELLI

MODELLO SEQUENZIALE LINEARE

Chiamato anche ciclo di vita classico o modello a cascata; suggerisce un approccio sistematico e sequenziale allo sviluppo.

Prevede:

1. **La strutturazione e modellazione del sistema e dei dati** – Il lavoro comincia dalla determinazione dei requisiti di tutti gli elementi del sistema e della attribuzione al software di una parte di tali requisiti quando il software è a contatto con elementi diversi come hardware, persone e database. In seguito la raccolta dei requisiti relativi a livello strategico aziendale e di area per la strutturazione dei dati.
2. **Analisi dei requisiti del software** – La raccolta dei requisiti si intensifica e si concentra nel software al fine di comprendere la natura dei programmi d costruire, l’analista deve comprendere il dominio delle informazioni per il software (Capitolo 11), casi come la funzionalità, il comportamento, le prestazioni e le interfacce richieste. I requisiti, sia del sistema sia del software, devono essere documentati e riveduti insieme al cliente.
3. **Progettazione** – Processo in quattro fasi distinte, la strutturazione dei dati, l’architettura software, le interfacce e i dettagli procedurali(algoritmi). Con queste fasi si può valutare la qualità prima della stesure del codice. Anche questa fase è documentata ed entra a far parte della configurazione del software.
4. **Generazione del Codice** – Se i passi precedenti sono avvenuti correttamente il processo di codifica nel linguaggio scelto può svolgersi in modo automatico.
5. **Collaudo** – Questa fase si concentra sugli aspetti logici interni del software, al fine di garantire che tutte le istruzioni siano provate, e sulle funzionalità esterne, al fine di scoprire eventuali errori e di accertarsi che, a fronte di dati di input specifici vengano prodotti risultati previsti.
6. **Manutenzione** – Le modifiche si rendono necessarie o perché si sono scoperti errori o per adeguare il software a mutate condizioni esterne (nuovo sistema operativo o nuova periferica) o perché il cliente richiede migliorie funzionali o prestazionali. Nella manutenzione si applicano tutte le fasi precedenti, in questo caso a un programma già esistente.

Il modello sequenziale lineare è il più vecchio, il più semplice da applicare all’ingegneria del software per cui il più diffuso.

PRO

– Nonostante sia troppo “vecchio” rimane un approccio ragionevole quando sono ben noti i requisiti

CONTRO

* Causa la poca iterazione (fra i moduli), ogni modifica è causa di confusione a mano a mano che il progetto avanza.
* Spesso è difficile per il cliente enunciare esplicitamente tutti i requisiti. Questo modello lo esige e non è in grado di governare in modo adeguato l’incertezza che si accompagna naturalmente al principio di un progetto.
* Il cliente vedrà una versione del software sono verso la fine dell’arco temporale di un progetto; un errore importante può avere conseguenze disastrose se non viene rilevato prima della fine del processo.
* In team complessi questo modello può portare a delle pause importanti nell’attesa che un altro team completi il suo ciclo di lavoro.