Ingegneria del Software

- ***<u>Lezione 1 - (7 / 10 / 2024)</u>***

➔ *<u>Primo Blocco</u>*

L'ingegneria del software è la disciplina che si occupa di sviluppare sistemi software di grandi dimensioni (in the large), garantendo la qualità e l'affidabilità del prodotto finale. A differenza della programmazione di piccoli progetti (in the small), l'ingegneria del software richiede un approccio sistematico e strutturato.

Quando parliamo di qualità del software parliamo di un prodotto che riesce a soddisfare le richieste e necessità del cliente.

L'ingegneria del Software (o Software Engineering) è la disciplina per la produzione del software secondo i principi dell'ingegneria, ovvero progettazione e validazione. Questo è essenziale per fare del software un prodotto industriale.
Se manca si incorre in scarsa qualità del prodotto e scarsa competitività (cost overrun, time overrun).

Per anni i costruttori di Hardware hanno visto la produzione di Software come un'attività banale, simile all'uso del calcolatore, che richiede principalmente abilità. Per anni l'abilità programmativa, la conoscenza delle ultime novità su linguaggi, interfacce etc., è stata considerata sufficiente a fare un ingegnere del Software. Infatti per anni l'Ingegneria del Software è stata considerata una branca della teoria della programmazione (o informatica teorica).

Prima degli anni '90 parecchi progetti software fallivano ancora. Nel Settembre del 1997 ci fu D.L.Parnas che scrisse un articolo dal titolo *"ingegneria del software un matrimonio non consumato"* su una rivista chiamata *comunication for ACM (CACM)*. Per matrimonio non consumato intendiamo che le cose da "far sposare" erano:

- ingegneri conoscano bene la programmazione
- informatici teorici conoscono bene la disciplina ingegneristica.

Il termine Ingegneria del Software è nato, circa alla fine degli anni '60, oltre 50 anni fa nella conferenza NATO, in Germania nel 1968. Qui si disse che l'Ingegneria del Software doveva essere considerato come una materia ingegneristica.

Risultati del '68: l'attività della programmazione non è né una scienza né una matematica, poiché il programmatore non aggiunge nessuna conoscenza a quella già esistente, ma costruisce un PRODOTTO. Gli ingegneri devono basare sulla teoria della programmazione i loro principi ingegneristici, ovvero progettazione e

validazione, dei prodotti software. I problemi e i rischi connessi alla produzione e all'uso del software (bassa qualità, time e cost overrun) sono tipici di informatici teorici che non conoscono e applicano i principi ingegneristici.

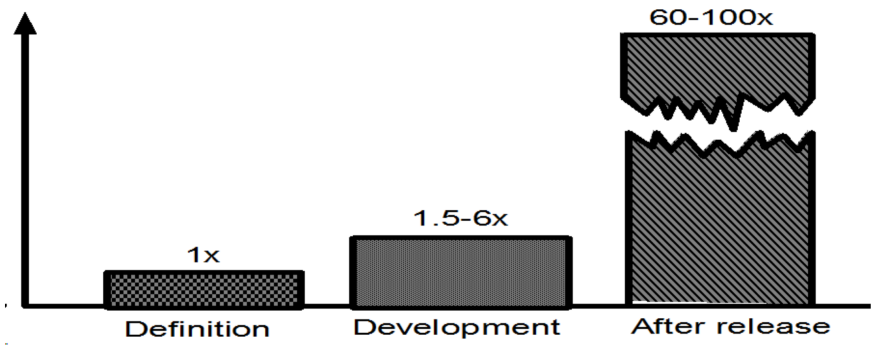
Aspetti tipici dell'ingegneria del Software:

- difficoltà ACCIDENTALI (superabili con il progesso) del prodotto software:
    - di attitudine
    - di manutenzione
    - di specifica e progetto
    - di teaming
- Aspetti ESSENZIALI (non superabili con il progresso) del prodotto Software:
    - complessità
    - conformità
    - cambiabilità
    - invisibilità

Il ciclo di vita di un software è l'intervallo temporale da quando nasce il software a quando viene dismesso (quindi fin quando muore). Si articola in 3 stadi e 6 fasi principali. La produzione si suddivide in due stadi: sviluppo e manutenzione. La fase finale è il terzo stadio, ovvero la dismissione.

- 1° Stadio) Sviluppo = 6 fasi
    1. Requisiti
    2. Specifiche (o analisi dei requisiti)
    3. Pianificazione
    4. Progetto (preliminare e dettagliato)
    5. Codifica
    6. Integrazione
- 2° Stadio) Manutenzione = copre circa il 60% dei costi del ciclo di vita
- 3° Stadio) Dismissione

L'effetto delle modifiche varia secondo la fase in cui vengono introdotte. In fasi avanzate, una modifica può comportare rivolgimenti che richiedono nuove risorse o correzioni importanti al progetto, cioè costi supplementari.



Comunemente noi per correggere vari errori in un programma lo “testiamo”.
In Ingegneria del Software il testing è una fase separata, non esplicitamente menzionato tra le 6 fasi, poiché è un’attività che ha luogo durante l’intero sviluppo.
Questa avviene in due modi:

- Verifica (alla fine di ogni fase) → [Verifica = la fase è stata ben svolta ?]
- Validazione (alla fine dello sviluppo) → [Validazione = il prodotto finale è buono ?]

La percentuale di difetti rilevati prima del rilascio del software (nota come Defect Removal Efficiency - DRE) indica quanto efficacemente un team di sviluppo riesce a individuare e correggere i difetti prima che il software venga distribuito agli utenti.

Esempio:

- Se un team di sviluppo trova 900 difetti nel software prima della data di rilascio, e gli utenti ne rilevano 100 durante un periodo standard (spesso di 90 giorni post-rilascio), allora il DRE è pari al 90%.
- Questo 90% si calcola come rapporto tra i difetti trovati prima del rilascio (900) e il totale dei difetti (900 trovati prima + 100 trovati dagli utenti), quindi:

$$DRE = \frac{900}{900 + 100} \times 100 = 90\%$$

- ***Lezione 2 - (10 / 10 / 2024)***

Costruire un Prodotto Software ha un costo. Questo costo lo si può applicare riguardo dimensione (size), repliche e ampiezza di mercato.

Per fare due prodotti di dimensione S/2 (due mezzi prodotti) costa meno che farne uno di dimensione S, poiché il costo della dimensione verrà calcolato come $C = a\,S^2$

$$\left(\frac{S}{2}\right)^2 + \left(\frac{S}{2}\right)^2 = \left(\frac{S}{2}\right)^2$$

Immaginiamo che realizzo un prodotto Software, che mi è costato *x*, produrre una replica di esso costa 0. Infatti per i Prodotti Software produrre una replica non costa niente, invece, per i Prodotti Hardware invece ci sta un costo di replica.

Un prodotto di dimensione (size) doppia richiede un prezzo quattro volte superiore o un mercato quattro volte più ampio.

Introduciamo alcuni termini:

- Con la dicitura *Prodotto Software* intendiamo non soltanto il prodotto finale di un codice, ma tutta la documentazione che viene prodotta dall'inizio del ciclo di vita del software (Documento di specifica + Codice).
- Un *Artefatto* può essere un documento requisiti, un documento di specifica o un documento di progetto.
- *Codice* è il prodotto software finale
- *Sistema Software* è un insieme organizzato di prodotti Software (ES: Word, PPT, ecc…), ma anche un insieme di software e di hardware

- *Cliente:* colui che ordina o commissiona e che paga per lo sviluppo e la manutenzione del prodotto Software
- *Sviluppatore:* è il soggetto al cui viene commissionato questo Prodotto Software (solitamente è un'azienda)
- *Utente:* colui che usa il prodotto Software

Abbiamo due tipologie di Software:

- *Software a contratto:* cliente e sviluppatore sono soggetti differenti – lo sviluppatore farà il prodotto software su richiesta del cliente
- *Software Interno:* è un sottotipo del software a contatto, qui il cliente e sviluppatore coincidono – ovvero cliente e sviluppatore fanno parte della stessa organizzazione/azienda

Un aspetto importante del prodotto Software è l'affidabilità

L'affidabilità la possiamo definire attraverso una definizione informale come la credibilità del prodotto software. Possiamo definirla pure dal punto di vista formale come la probabilità che il prodotto software lavora correttamente in un intervallo temporale.

Il Funzionamento Corretto di un prodotto è legato al concetto di Guasto (*failure*), ovvero il comportamento anomalo del prodotto Software dovuto alla presenza di un difetto.

Un difetto (detto *Defect* o *Bug*) è un'anomalia presente nel prodotto software.

Questi difetti nel prodotto Software ci finiscono per Errore, ovvero un'azione errata di chi per ignoranza o distrazione introduce un difetto nel prodotto Software.

In un prodotto Software quando ho un Difetto non per forza deve verificarsi un Guasto.
Un prodotto software con molti difetti è poco affidabile. Infatti l'affidabilità del prodotto migliora via via che si riduce il numero di difetti.

Tra l'affidabilità osservata e il numero di difetti latenti c'è una relazione non-semplice.

L'eliminare difetti da parti del prodotto usate raramente, non comporta grandi effetti sull'affidabilità osservata.

Esempio: su word uso determinate funzioni di più (finestra Edit, il Correttore, la Funzione Stampa), poi esistono altre funzioni che usiamo raramente (modificare il layout inserendo colonne).

Esiste una regola 10-90 la quale evidenzia che, in programmi di grandi dimensioni, la maggior parte del tempo di esecuzione (il 90%) è spesa eseguendo una piccola porzione di istruzioni (il 10%). Questo 10% delle istruzioni, che richiede la maggior parte del tempo di calcolo, è chiamato core o nucleo del programma. Quindi, il core del programma rappresenta quella parte cruciale del codice che è più "pesante" in termini di esecuzione e quindi merita un'ottimizzazione particolare per migliorare le prestazioni complessive del software.

Il miglioramento dell'affidabilità di un software dipende dall'eliminazione dei difetti e, soprattutto, dalla loro posizione: è più significativo se i difetti si trovano nel nucleo del programma.

Tutto ciò definisce il profilo Operativo, detto anche *operational profile*.

I difetti in un prodotto Software sono unici e soggettivi, poiché non tutti gli utenti usano il Software allo stesso modo. Infatti l'affidabilità di un prodotto Software dipende dall'utente.

Confronto tra Affidabilità Software e Hardware:

- I guasti in un prodotto Software sono dovuti alla presenza di difetti nei programmi. Negli anni i software non si "consumano". Questi difetti sono latenti / nascosti.
- I guasti in un prodotto Hardware sono quasi sempre dovuti al consumo o al deterioramento dei componenti, ciò comporta che qualche componente non si comporta più come dovrebbe o si rompe.

Una volta corretti questi difetti, dopo la riparazione:

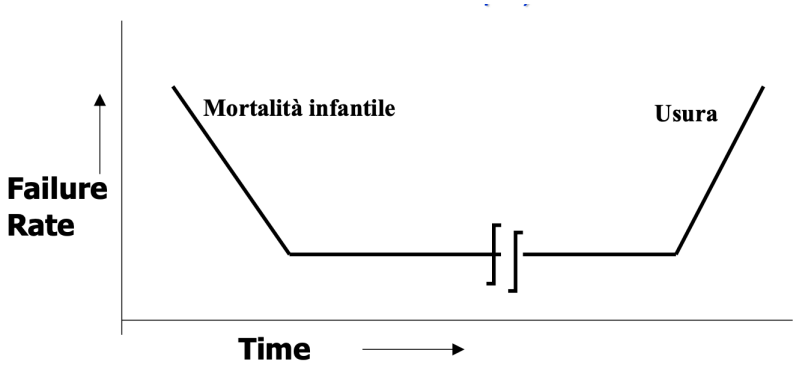
- l'affidabilità dell'Hardware torna come era prima
- l'affidabilità del Software può aumentare o diminuire

L'obiettivo dell'affidabilità per l'hardware è mantenere stabile la frequenza di guasto, mentre per il software è ridurre la frequenza di guasto nel tempo, aumentando così l'affidabilità.

Esempio:

- vogliamo che un computer portatile non inizi a rompersi di più con l'uso, ma mantenga la sua affidabilità costante.
- vogliamo che il programma diventi sempre più stabile e con meno errori man mano che viene aggiornato.

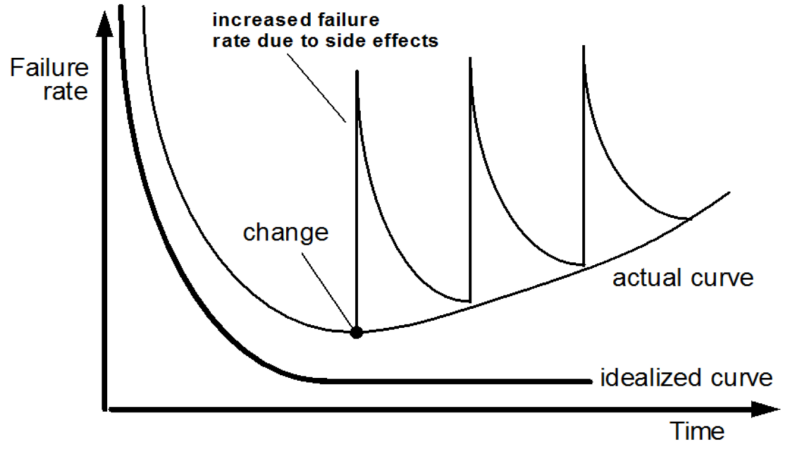
- Andamento frequenza di guasto Hardware



Possiamo definire questo andamento come “andamento vasca da bagno”. Questo andamento ci dice che:

È noto che i prodotti una volta usciti dalla fabbrica tendono ad avere una mortalità infantile. Bisognerebbe tenere la frequenza di guasto costante. Ma dopo un lungo periodo arriveremo all’usura del prodotto.

- Andamento frequenza di guasto Software



- ***<u>Lezione 3 – (14/10/2024)</u>***

Una metrica importante legata all'affidabilità del software è la disponibilità, che rappresenta la percentuale di tempo durante il quale il software è risultato usabile nel corso della sua vita. Questa percentuale è influenzata dal numero di guasti che si verificano e dal tempo necessario per risolverli.

L'affidabilità e la disponibilità sono metriche fondamentali /importanti per quei sistemi in cui l'interruzione del servizio può portare a una perdita di efficienza e sicurezza, con conseguenze economiche e sociali significative. Come ad esempio:

- Sistemi di trasporto
- Sistemi di gestione del traffico aereo
- Sistemi di controllo del volo
- Sistemi di produzione e distribuzione di energia
- Sistemi di comunicazione

Nel corso degli anni, lo sviluppo software ha attraversato diverse fasi:

- Fase di abilità: dove predominano gli aspetti di lavoro individuale e creativo
- Fase artigianale: è quella che ha portato a non usare un singolo programmatore ma a sviluppare piccoli gruppi/team specializzati
- Fase industriale: Qui, lo sviluppo e la manutenzione del software sono pianificati e coordinati, con un crescente supporto da parte di strumenti automatici.

Lo standard IEEE Std. 610.12 (1990) ha fornito una definizione più completa dell'ingegneria del software, comprendente:

- L'applicazione di un approccio sistematico, disciplinato e misurabile nello sviluppo, esercizio e manutenzione del software, utilizzando principi ingegneristici.
- Lo studio di questi approcci delineati nel punto precedente.

Quindi possiamo dire che il software può essere considerato come l'insieme di programmi, documenti e dati multimediali. Realizzati dall'ingegnere del software applicando un processo che conduca a risultati migliori. Questo viene fatto attraverso un approccio ingegneristico. Abbiamo già detto precedentemente che il software non si consuma, è complesso, invisibile, si conforma, si cambia.

I metodi e le tecniche di ingegneria del software mirano a garantire la qualità del prodotto, gestire la crescente domanda rispettando il budget, aggiornare applicazioni legacy, evitare ritardi nelle consegne e implementare con successo nuove tecnologie.

Miti (da sfatare) del Software:

- In caso di ritardo, basta aumentare il numero di programmatori;
- Una descrizione generica è sufficiente a scrivere i programmi. Eventuali modifiche si possono facilmente effettuare in seguito;
- Una volta messo in opera il programma, il lavoro è finito;
- Non c'è modo di valutare la qualità fino a quando non si ha a disposizione il prodotto finale;
- L'ingegneria del software è costosa e rallenta la produzione;