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.

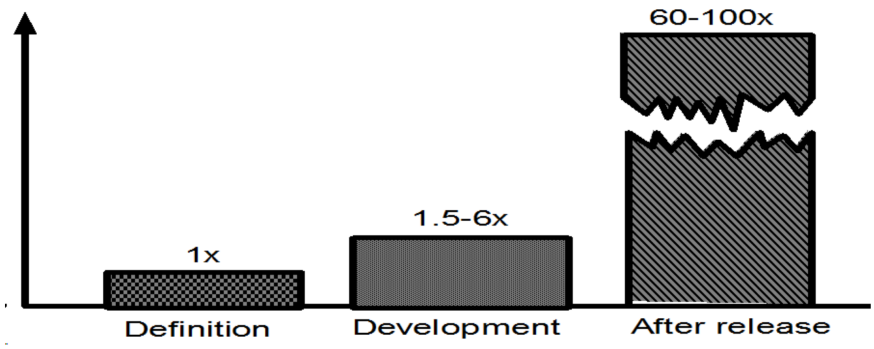
Aspetti tipici dell'ingegneria del Software:

- 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:

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