- *Sesto Blocco → è un ppt con un esercizio*

- ***Lezione 17***

- *Settimo Blocco*

Lo sviluppo software è un'attività complessa che richiede una specifica Gestione di Progetti Software (Software Project Management). Questa gestione implica:

- la pianificazione,
- il monitoraggio
- il controllo di persone, processi ed eventi durante lo sviluppo del prodotto

Il Software Project Management Plan (SPMP) è il documento che guida la gestione di un progetto software.

La gestione di un prodotto software si fonda su un concetto che viene riassunto con un termine "Le Quattro P". Rappresentano i 4 elementi fondamentali alla base del progetto Software.

- Persone, che rappresentano l'elemento più importante di un progetto software di successo (il SEI ha elaborato il People Management - Capability Maturity Model)
- Prodotto, che identifica le caratteristiche del software che deve essere sviluppato (obiettivi, dati, funzioni, comportamenti principali, alternative, vincoli)
- Processo, che definisce il quadro di riferimento entro cui si stabilisce il piano complessivo di sviluppo del prodotto software
- Progetto, che definisce l'insieme delle attività da svolgere, identificando persone, compiti, tempi e costi

L'ingegneria ci aiuta fornendoci vari approcci che possono essere utilizzati per gestire team o per organizzare le persone che sviluppano il software e così via.

- Approccio democratico

  È stato introdotto da Weinberg, nel 1971. È basato sul concetto di *egoless programming*, secondo cui ogni sviluppatore valuta la scoperta di fault (guasto) nel proprio codice come un attacco alla propria persona e non come un evento usuale. Questo approccio punta ad organizzare team che lavorino con un obiettivo comune, senza un singolo leader. Il codice appartiene al team come entità e non al singolo sviluppatore.

- Vantaggi:

  - atteggiamento positivo verso la ricerca di fault

  - molto produttivo in caso di problemi difficili da risolvere (es. ambienti di ricerca)

- Svantaggi:

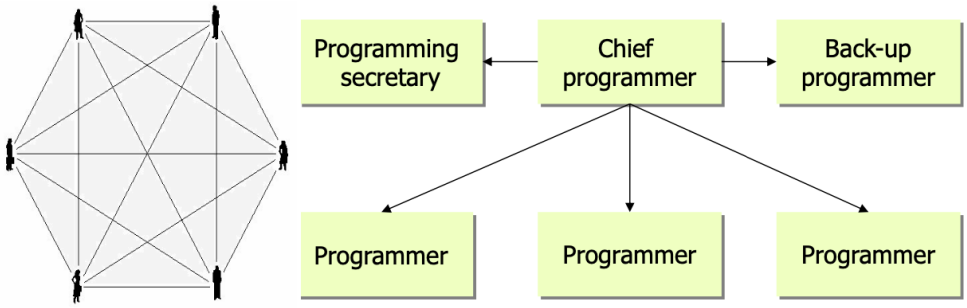
  - l'approccio non può essere imposto dall'esterno ma deve nascere spontaneamente

  - gli sviluppatori più anziani non desiderano essere valutati dai più giovani

- Approccio con Capo – Programmatore, detto anche Gerarchico

  Questo si basa su concetti di:

  - Specializzazione → ogni partecipante svolge i compiti per i quali è stato formato
  - Gerarchia → ogni sviluppatore comunica esclusivamente con il Capo - Programmatore, che dirige le attività ed è responsabile dei risultati



All'interno di ogni team c'è un Chief Programmaer (Capo Programmatore), al quale tutti i programmatori rendono conto.

Esiste la figura da Back-up Programmer, che è una persona (programmatore) che testa i programmi in caso di conflitti tra Programmatori e Capo Programmatore.

- Vantaggio → Un vantaggio immediato è che diminuisce il numero di canali di comunicazione
- Svantaggio → è che richiede personale molto esperto per ricoprire gli incarichi di:

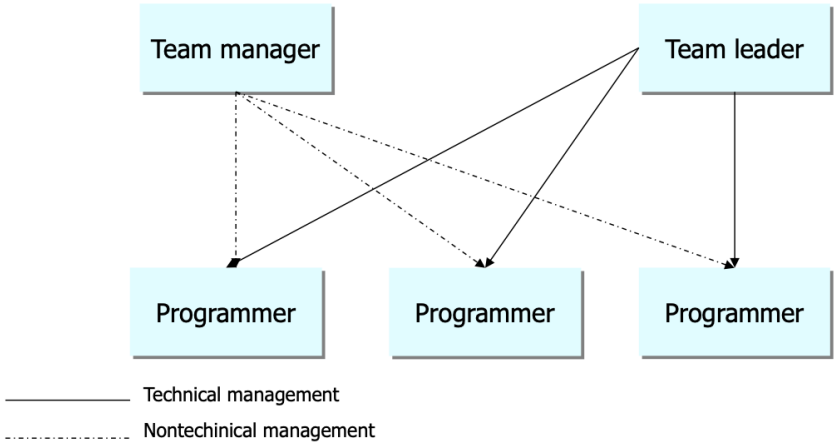
  - capo-programmatore (è sia un manager che un tecnico, sviluppa il progetto architetturale e le parti di codice critiche)

- programmatore di back-up (sostituisce il capo-programmatore ed è responsabile delle attività di testing)

- segretario di programmazione (responsabile della documentazione e dell'archivio di produzione)

C’è stata una sorta di Evoluzione degli Approcci, con il quale si vuole distinguere il ruolo gestionale dal ruolo tecnico. Il capo-programmatore, essendo sia manager che tecnico, risulta essere il valutatore di se stesso. Va dunque sostituito con due individui:

- Team Leader (responsabile aspetti tecnici)
- Team Manager (responsabile aspetti gestionali)



Esiste una statistica elaborata su progetti di sviluppo software negli USA dal 1984 al 2016, per dire quanti di questi progetti sono stati terminati in tempo, ritardati o cancellati.

	Application Types	On-time	Late	Canceled
1	Scientific	68%	20%	12%
2	Smart phones	67%	19%	14%
3	Open source	63%	36%	7%
4	U.S. outsource	60%	30%	10%
5	Cloud	59%	29%	12%
6	Web applications	55%	30%	15%
7	Games and entertainment	54%	36%	10%
8	Offshore outsource	48%	37%	15%
9	Embedded software	47%	33%	20%
10	Systems and middleware	45%	45%	10%
11	Information technology (IT)	45%	40%	15%
12	Commercial	44%	41%	15%
13	Military and defense	40%	45%	15%
14	Legacy renovation	30%	55%	15%
15	Civilian government	27%	63%	10%
	Total Applications	**50.13%**	**37.27%**	**13%**

La riuscita di un Progetto Software dipende dalla pianificazione di Progetti Software

- Obiettivo è definire un quadro di riferimento per controllare, determinare l'avanzamento ed osservare lo sviluppo di un progetto software
- Motivazione è essere in grado di sviluppare prodotti software nei tempi e costi stabiliti, con le desiderate caratteristiche di qualità
- Componenti fondamentali di una corretta pianificazione software sono:
    - Scoping (raggio d'azione): comprendere il problema ed il lavoro che deve essere svolto
    - Stime: prevedere tempi, costi e effort (quantità di lavoro)
    - Rischi: definire le modalità per l'analisi e la gestione dei rischi
    - Schedule: allocare le risorse disponibili e stabilire i punti di controllo nell'arco temporale del progetto
    - Strategia di controllo: stabilire un quadro di riferimento per il controllo di qualità e per il controllo dei cambiamenti

Una volta compreso cosa dovrà fornire il software. Quanto costerà sviluppare un prodotto?

Lo faremo con la “Stima”. Le attività di stima di tempi, costi ed effort nei progetti software sono effettuate con gli obiettivi di:

- ridurre al minimo il grado di incertezza
- limitare i rischi comportati da una stima

Risulta quindi necessario usare tecniche per incrementare l'affidabilità e l’accuratezza di una stima. Le tecniche di stima possono basarsi su:

- stime su progetti simili già completati (expert judgement by analogy)
- "tecniche di scomposizione" (approccio bottom-up), utilizzano una strategia "divide et impera" e sono basate su:
    - stime dimensionali, ad es. LOC (Lines Of Code) o FP (Function Point)
    - suddivisione dei task e/o delle funzioni con relativa stima di allocazione dell'effort

- modelli algoritmici empirici, si basano su dati storici e su relazioni del tipo:

$$d = f\ (\ v_i\ )$$

  dove d è il valore da stimare (es. effort, costo, durata) e v_i sono le variabili indipendenti (es. LOC o FP stimati)