Politecnico di Torino

Facoltà di Ingegneria

Laurea Magistrale in Ingegneria Gestionale

Analisi dei descrittori di progetti IT finalizzata all'ottimizzazione dei processi di stima a finire basati su algoritmi di Machine Learning

Candidato: Niccolò Revelli

Relatore: Alberto De Marco

Correlatore: Filippo Ottaviani

Anno accademico 2019-2020

RINGRAZIAMENTI

Terminato questo lavoro di tesi e questo percorso universitario, vorrei dedicare alcune parole a chi mi ha supportato nella sua redazione e a chi mi supporta nella vita di tutti i giorni.
Innanzitutto, ringrazio il mio relatore Alberto De Marco per l'opportunità che mi ha concesso e il mio correlatore Filippo Ottaviani, sempre pronto a darmi le giuste indicazioni in ogni fase della realizzazione dell'elaborato e ad aiutarmi nella ricerca del materiale utile a completarlo.
Un ringraziamento speciale va al mio tutor Davide Vitta che, in questi primi mesi di lavoro insieme, ha contribuito ad un netto miglioramento delle mie competenze tecniche e di management.
Ringrazio, inoltre, tutto il nostro team di lavoro, che mi ha subito fatto sentire uno di loro e mi ha stupito per il livello di competenza posseduta.
Ringrazio mio fratello Emanuele, perché nella nostra crescita insieme abbiamo imparato a sostenerci a vicenda e siamo diventati una squadra imbattibile.
Ringrazio i miei genitori Alberto e Patrizia, perché senza di loro non avrei mai potuto intraprendere questo percorso di studi e non avrei potuto vivere molte altre esperienze che hanno plasmato il mio carattere, rendendomi la persona che sono ora.
Ringrazio i miei nonni Pina, Nello, Lalla e Renzo per avermi protetto nei momenti difficili e avermi dato l'esempio a cui aspirare.
Ringrazio tutti gli amici di sempre Matteo, Enrico, Riccardo, Silvia, Giulia, Giulia, Luca, Pietro e tutti gli altri (avrei bisogno di ben più spazio per nominarvi tutti), che mi hanno accompagnato e spero continueranno ad accompagnarmi durante il mio percorso di vita dandomi forza e felicità anche solo con la loro presenza silenziosa. Ne abbiamo vissute tante insieme, ma sono convinto che il meglio debba ancora venire e che riusciremo a superare qualsiasi ostacolo la vita ci metterà davanti.
Infine, ultimo nell'ordine ma non ultimo per importanza, il più grande grazie di tutti vorrei rivolgerlo a me stesso.
Solo io so davvero quali e quante difficoltà ho affrontato durante il mio percorso universitario e, per questo motivo, sono profondamente orgoglioso dei risultati raggiunti.
Questo percorso mi ha fatto scoprire delle qualità che non sapevo di avere e mi ha reso una persona nuova, più forte e più consapevole.
Per questo motivo, mi preparo al futuro con grande curiosità riguardo alle sfide che mi attendono, conscio del fatto di essere pronto ad affrontare tutto ciò che la vita metterà sul mio cammino.
Oggi si chiude un percorso difficile e bellissimo, da domani si torna in pista per iniziarne uno nuovo.

Grazie di cuore a tutti!

Nick

INTRODUZIONE

L'obiettivo di questo lavoro è quello di porre le basi per effettuare stime predittive dei tempi e dei costi di attività progettuali legate all'Information Technology, mediante l'utilizzo di algoritmi di machine learning. Poichè i risultati raggiunti sono fortemente vincolati alla tecnologia utilizzata, è necessario impostare a monte il lavoro e delineare un framework definito che permetta una corretta interpretazione degli output forniti dall'algoritmo.

La presente tesi nasce e si sviluppa in parallelo a uno stage svolto presso una società di consulenza in ambito ingegneristico, che vuole implementare le sue metodologie di stima e monitoraggio dei tempi e dei costi delle attività progettuali in modo da avere un controllo più efficiente sullo stato di avanzamento dei lavori e sull'utilizzo delle risorse. Dato il forte interesse della società per i progetti IT, il caso di studio è stato sviluppato in questo ambito.

La prima parte della tesi si focalizzerà sulla descrizione dei principali processi del Project Management tradizionale e sullo studio delle metodologie di stima dei costi e dei tempi utilizzate nel Project Management in ambito IT, a partire da quelle di tipo lineare fino ad arrivare a quelle di tipo non lineare.

La seconda parte, invece, sarà focalizzata sull'identificazione dei principali descrittori dei work packages di un database di progetti IT e sull'analisi delle relazioni che intercorrono tra di essi, al fine di predisporre i dati all'utilizzo di metodologie di programmazione con intelligenza artificiale.

INDICE

1. IL PROJECT MANAGEMENT

Per dare una corretta definizione del Project Management bisogna innanzitutto partire dal concetto di **progetto**, ovvero un insieme di attività interdipendenti e complesse che iniziano con l'identificazione dell'opera che si vuole realizzare e finiscono con la gestione dei risultati raggiunti.

Basandosi sulle definizioni fornite dal *Project Management Institute*, dalla metodologia inglese Prince2 e dalle principali istituzioni nell'ambito del Project Management, è possibile identificare alcune caratteristiche comuni e imprescindibili nella definizione di un progetto:

- Un progetto è **temporaneo**, è caratterizzato da una data d'inizio e una di fine, e presenta sempre una durata stabilita in fase di pianificazione.
- Un progetto è **unico**, ha come obiettivo la realizzazione di qualcosa di nuovo che ancora non esiste e si distingue dagli altri grazie a delle caratteristiche ben definite.
- Un progetto è caratterizzato da una sua **complessità**, richiede l'integrazione di funzioni aziendali tecniche e gestionali e di un coordinamento ad alto livello per la sua realizzazione.
- Un progetto è sempre soggetto a **vincoli** legati alle risorse e si basa sul raggiungimento di un opportuno trade off tra tempi, costi e qualità.
- Un progetto ha sempre un **obiettivo** chiaro e definito in fase di pianificazione, che deve essere raggiunto rispettando i vincoli legati ai tempi e ai costi.

Oltre a questi aspetti comuni, è opportuno sottolineare che ogni progetto è diviso in varie **fasi** connesse e dipendenti tra loro, ciascuna delle quali presenta delle caratteristiche e degli obiettivi ben definiti. L'output di ogni fase influenza l'input di quella seguente e può provocare importanti conseguenze sul progetto dal punto

di vista del rischio, dell'incertezza, dei tempi e dei costi. Per questo motivo, risulta fondamentale la presenza di un'efficiente **organizzazione** dal punto di vista tecnico e manageriale finalizzata alla gestione di eventuali criticità.

Tali criticità nascono principalmente a causa della limitatezza delle risorse disponibili, che non permettono di raggiungere contemporaneamente i massimi risultati dal punto di vista di costi, tempi e qualità. La combinazione di questi tre elementi ed il loro impatto sui deliverables del progetto è detto **triplice vincolo.**

Infatti, ogni variazione in un lato del triangolo avrà un effetto sugli altri due e tutti insieme produrranno una differente configurazione qualitativa. Per esempio, dal momento che le risorse impegnate nel progetto sono considerate una voce di costo, se esse vengono modificate per gestire una quantità superiore di lavoro, i costi aumenteranno in modo proporzionale. Al contrario, l'impiego di una quantità inferiore di risorse porta sì ad una riduzione dei costi, ma comporta anche una dilatazione della schedulazione di progetto.

Per questa ragione, nella maggior parte dei progetti almeno uno dei due lati è fisso e si decide di agire sui rimanenti due per rispettare il mandato di progetto. Quindi è molto importante fare chiarezza su quali di questi elementi debba essere privilegiato per l'ottimizzazione del progetto.

A partire da queste considerazioni, si può affermare che il **Project Management** nasce per trovare il giusto trade off tra i fattori produttivi rispettando il vincolo delle risorse disponibili, con l'obiettivo di soddisfare tutti gli **stakeholder** di progetto. Il PMBOK (*Project Management Body of Knowledge*) identifica nove aree di conoscenza legate al Project Management: ambito, tempi, costi, qualità, rischi, integrazione, risorse umane, comunicazione e approvvigionamenti. La gestione di questi aspetti viene applicata alle cinque fasi che caratterizzano ciascun progetto: l'inizio del progetto, la pianificazione, l'esecuzione, il monitoraggio e controllo e, infine, la chiusura del progetto.

Una volta definito il significato del termine progetto, si può cercare di dare una definizione al concetto di Project Management che, secondo il **PMI**, è "*…l'applicazione di conoscenze, abilità, strumenti e tecniche alle attività di*

progetto per soddisfarne i requisiti. La gestione di progetti viene eseguita tramite l'uso di processi quali inizio ufficiale, pianificazione, esecuzione, controllo e chiusura".

A partire da questa definizione, si può quindi affermare che il Project Management consiste nell'unione e nell'uso di diverse caratteristiche e strumenti, sia tangibili sia intangibili, finalizzati alla realizzazione di processi ben definiti che portino ad una creazione di valore per i clienti e per gli stakeholder.

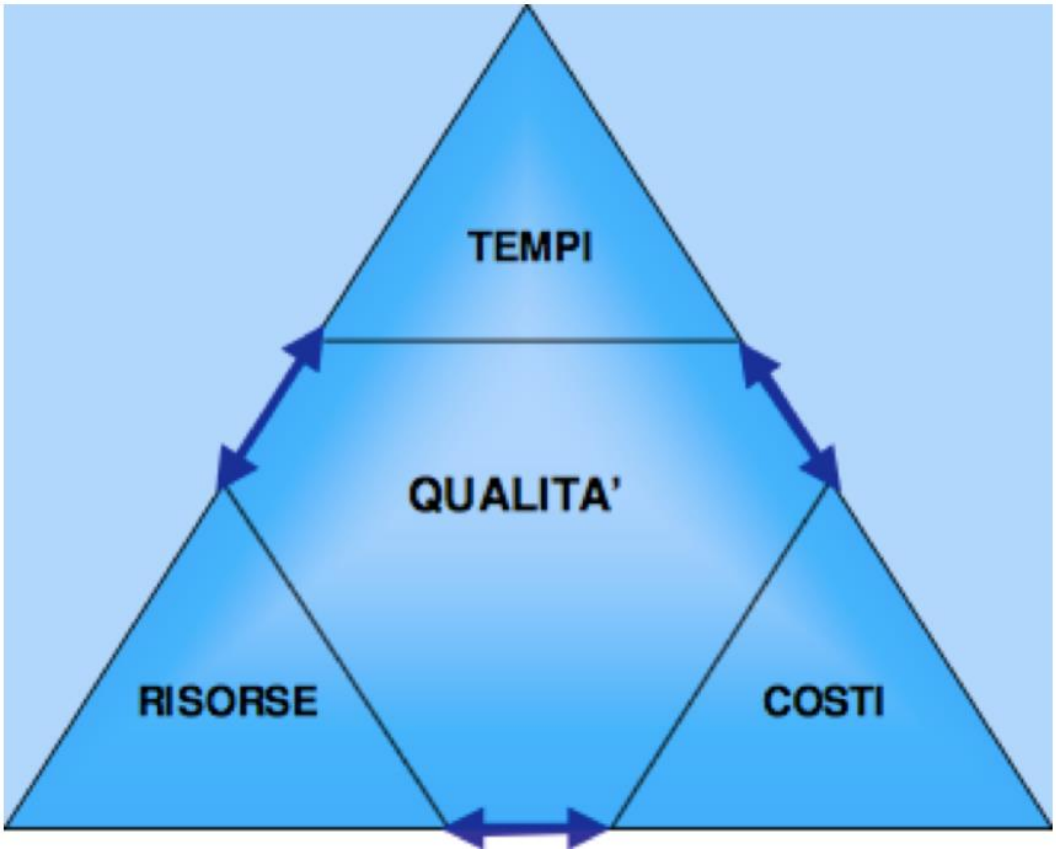

Figura 1. Il triangolo costi – tempi – qualità.

1.1. L'INFORMATION TECHNOLOGY

Il termine **Information Technology (IT)** sta ad indicare l'utilizzo di elaboratori e dispositivi di telecomunicazione per memorizzare, recuperare, trasmettere e manipolare dati legati a contesti di attività commerciali o di natura economica. Questo termine comprende non solo i computer e le reti di computer, ma anche altre tecnologie utilizzate per la diffusione di informazione come televisioni e telefoni. Ad oggi, quando si parla di IT, ci si riferisce prevalentemente a tecnologie legate ad hardware, software, elettronica e servizi informatici.

A partire dagli anni Ottanta, l'impatto dell'Information Technology sui meccanismi di business ha fatto sì che essa diventasse una risorsa strategica in grado di facilitare l'adattamento ai cambiamenti del mercato, di migliorare l'assetto organizzativo e funzionale dell'azienda e di creare un vantaggio competitivo. Per questo motivo, ogni azienda oggi ha la necessità di gestire questa risorsa nel modo più efficiente possibile al fine di adattarsi nel migliore dei modi al contesto economico, anche a costo di rivoluzionare i meccanismi alla base dei propri sistemi informativi di gestione.

Di conseguenza, le operazioni e gli investimenti legati al campo delle IT costituiscono una parte consistente del budget di un'azienda. Secondo uno studio effettuato da **Gartner** nel 2007, i costi legati alle IT hanno consumato il 4,4% dei ricavi delle aziende europee. Negli ultimi anni tali investimenti sono stati sempre più incentrati sui **Commercial Off-The-Shell** (COTS), ovvero delle soluzioni IT **plug and play** che non richiedono un grande cambiamento del codice sorgente per essere installati e i cui costi sono più facili da prevedere, essendo legati al pagamento di licenze mensili o annuali.

In questo contesto, si collocano le aziende fornitrici di servizi IT che non si focalizzano solo sull'aspetto tecnologico, ma anche sulla qualità dei servizi e sulla relazione con il cliente.

1.2. IT PROJECT MANAGEMENT

Per IT Project Management (**ITPM**) si intende il processo di gestione della pianificazione, dell'organizzazione e dell'aspetto economico legato ad un progetto di IT. Dal momento che questa tecnologia viene utilizzata da imprese anche molto diverse tra loro, lo scope di questo tipo di progetti può essere ampio e complesso. Per questo motivo, l'IT project manager non si può limitare ad applicare solo la sua conoscenza e ad utilizzare tools e tecniche utili al completamento del progetto, ma deve anche sapersi adattare ai rapidi upgrade tecnologici e ai cambiamenti che possono avvenire durante il ciclo di vita dello stesso e che richiedono la capacità di rivedere in corsa la pianificazione dei tempi.

L'ITPM è costituito dalle medesime cinque fasi del project management tradizionale, ciò che cambia è la gestione del ciclo di vita del progetto. Esistono diverse **metodologie** di gestione di progetti IT, di seguito verranno presentate quelle più utilizzate.

1.3. LA METODOLOGIA WATERFALL

La metodologia **Waterfall** fu introdotta dal Dr. Winston W. Royce nel 1970 nel paper "Managing the development of large software systems" e rispecchia il tipico approccio *stage and gate* dei processi legati alla produzione industriale. Questa metodologia prevede che l'attività progettuale sia suddivisa in una serie di **fasi** strettamente **sequenziali** e non sovrapponibili. Ciascuna fase è caratterizzata da attività differenti che producono **deliverable** da terminare prima dell'inizio della fase successiva. Inoltre, le persone che lavorano ad ogni attività possono essere diverse a seconda delle loro competenze.

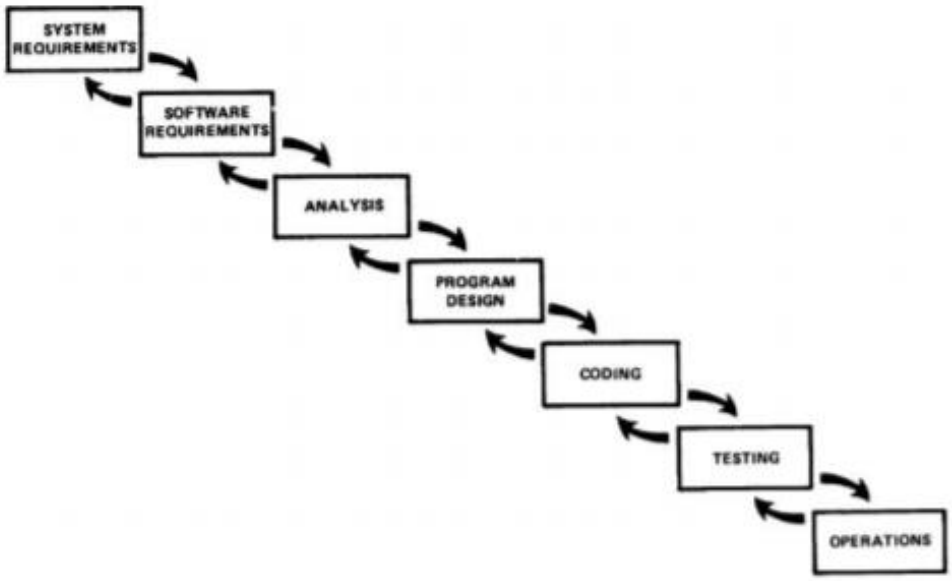

Figura 2. Fasi della metodologia Waterfall.

Un altro elemento importante di questo tipo di approccio è costituito dalle possibili **iterazioni** di ciascuno step della cascata. Infatti, a mano a mano che l'attività progettuale procede e la progettazione entra più nel dettaglio, si crea un'iterazione di uno step con quello precedente e quello successivo. Ciò significa che, nel caso in cui emergano problemi in una particolare fase, è possibile che sia necessario fare una revisione di una fase precedente o dell'intera attività progettuale. Questo aspetto rappresenta una **criticità** all'interno del processo, in quanto può capitare che il progetto debba essere soggetto a modifiche di tipo *disruptive* che possono portare ad un ricarico anche del 100% su costi e tempi. Per questo motivo, l'andamento del rischio per le attività che seguono questa metodologia segue una *curva a s* che aumenta con l'avanzare delle fasi di progetto.

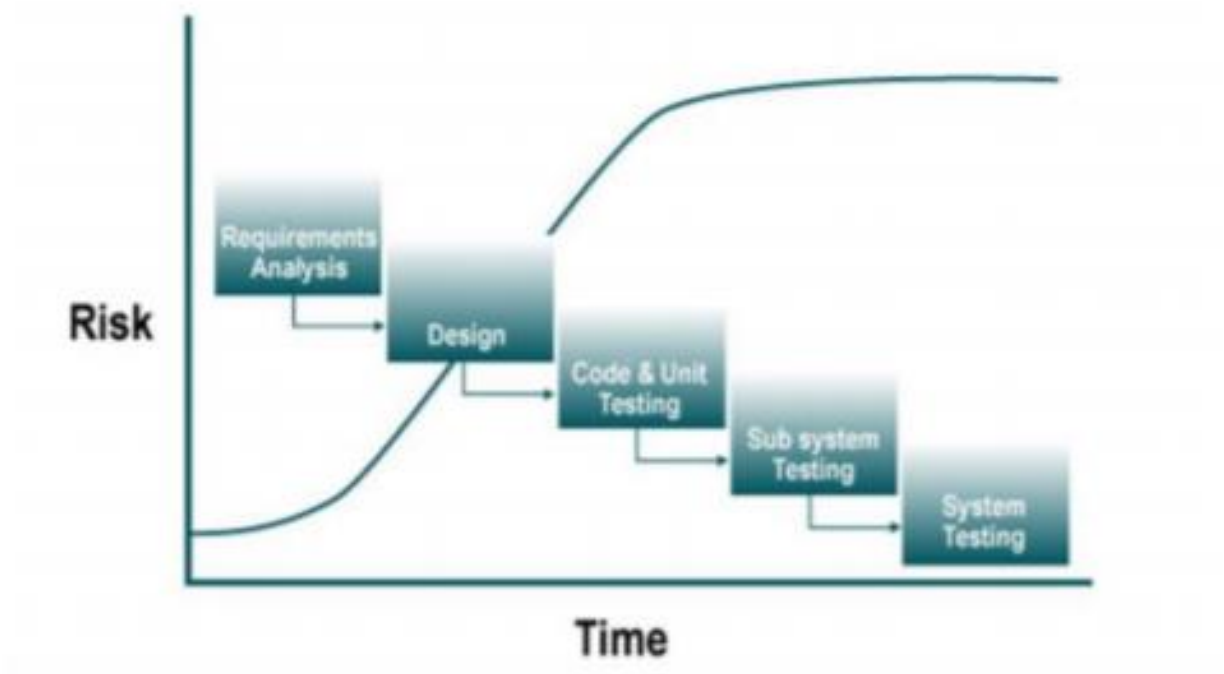

Grafico 1. Curva di andamento dei rischi nella metodologia Waterfall.

La metodologia Waterfall è diventata nel tempo un modello molto diffuso per i progetti di sviluppo software ed è largamente utilizzata da molte delle principali aziende del mondo, in particolare nei casi in cui si vogliono raggiungere elevati standard di qualità. È caratterizzata da un'alta attenzione per la pianificazione e dalla produzione di un'approfondita documentazione tecnica al termine di ogni fase. Spesso, al fine di prevenire l'incidenza di eventuali difetti di progettazione, si spende molto tempo prima di procedere allo step successivo. In questo modo, si evitano problemi di disallineamento tra chi ha terminato lo step di sua competenza e chi dovrà occuparsi della realizzazione di quello successivo.

La metodologia **Waterfall** garantisce quindi una chiara definizione e formalizzazione dei requisiti, la possibilità di intercettare potenziali difetti già nelle fasi preliminari di analisi e pianificazione e presenta dei vincoli temporali e di budget che permettono un agevole monitoraggio e controllo. Al termine di ogni fase, viene prodotta una documentazione molto dettagliata sui relativi deliverable e

ciò permette di gestire più facilmente il processo di sviluppo anche senza un personale con skills elevate.

Tuttavia, essa presenta alcune criticità legate alla sua **scarsa flessibilità**: infatti, la modifica in corso d'opera del progetto di fronte all'insorgenza di eventuali funzionalità da aggiungere o di nuovi requisiti si rivela spesso molto complicata. Inoltre, dal momento che il cliente prende visione dei deliverable solo al momento del completamento, la mancata identificazione degli errori lungo il processo può provocare gravi complicazioni. Queste caratteristiche fanno sì che la metodologia Waterfall sia più adatta per la gestione di progetti lunghi e con una bassa incertezza tecnologica, principalmente legati a mercati poco dinamici.

1.4. LA METODOLOGIA AGILE

Le metodologie **Agile** si concentrano maggiormente sulla realizzazione dei benefici piuttosto che sui deliverable da produrre e ciò ha rappresentato un cambiamento radicale negli approcci di gestione dei progetti in numerose aziende. Lo scopo delle imprese di sviluppo software è quello di produrre software di valore in poco tempo e minimizzando i costi. Per raggiungere questo obiettivo, è diventata fondamentale la capacità di gestire la crescente complessità legata allo sviluppo software e la continua evoluzione dei bisogni del cliente. Così, in contrasto con la metodologia Waterfall, caratterizzata per essere pesantemente regolata e controllata in ogni aspetto, nel 2001 un gruppo di studiosi pubblicò il *Manifesto per lo Sviluppo Agile di Software*, in cui vennero presentate le principali caratteristiche della metodologia Agile, di cui la **Scrum12** è la più diffusa.

La metodologia **Scrum** si basa su una suddivisione del progetto in processi **iterativi** e **incrementali**, detti **Sprint**, al termine dei quali vengono rilasciate piccoli parti funzionanti del software. Lo sviluppo del prodotto procede quindi per iterazioni, incrementi, espansioni e modifiche.

All’inizio del progetto, il team si riunisce per effettuare una prima pianificazione in cui vengono definiti i requisiti e viene redatto un documento ufficiale che dichiara qual è la **visione** del progetto. In questa fase, vengono identificati lo **Scrum Master** e gli stakeholders, viene creato lo **Scrum Team** e vengono scritte delle User Story, dette **Epic**, che vengono successivamente ordinate per andare a completare il **Prioritized Product Backlog** del progetto. Le User Story vengono approvate, stimate e vanno a comporre la lista delle attività che devono essere portate a termine durante ciascuno Sprint. In questa fase, inoltre, viene realizzato il piano dei rilasci e vengono stabilite le durate dei vari sprint.

Una volta completata la fase di pianificazione, il team inizia a lavorare alle attività utili all’implementazione dei deliverable dello Sprint. Man mano che il progetto va avanti, il team svolge degli stand up meeting giornalieri utili a fare il punto sullo stato di avanzamento dei lavori e a segnalare eventuali criticità.

Al termine di ogni Sprint, viene effettuato uno **Sprint Review Meeting** in cui viene data dimostrazione dei deliverable prodotti durante lo Sprint di fronte agli stakeholders al fine di ottenere l’accettazione del lavoro svolto. In caso di esito positivo, i deliverable vengono consegnati al cliente e si segnala la conclusione dello Sprint mediante un documento formale. Al termine di questo processo, gli stakeholders e il team si incontrano per valutare eventuali lezioni apprese durante lo Sprint.

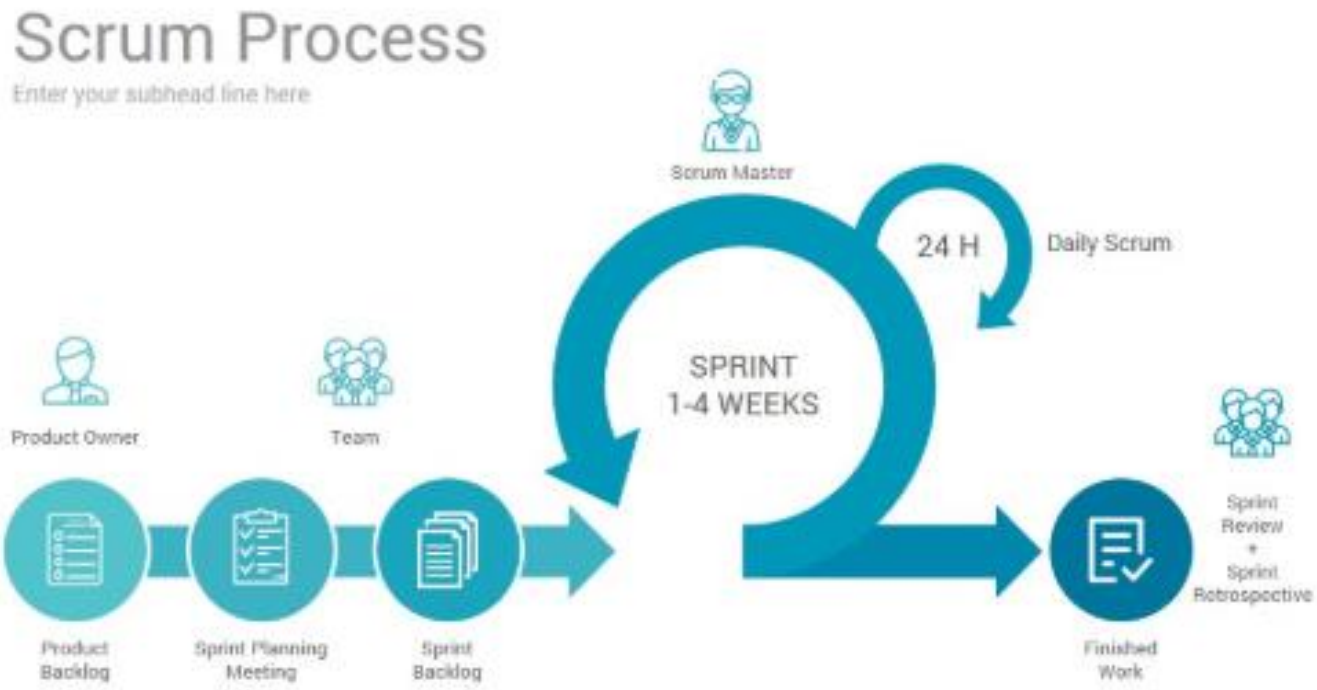

Figura 3. Il processo della metodologia Scrum.

Dall'analisi effettuata, risulta che il principale punto di forza delle metodologie Agile sia la capacità di rispondere in modo rapido ed efficace al cambiamento dei requisiti di progetto. Ciò fa sì che eventuali caratteristiche aggiuntive vengano integrate immediatamente, evitando che lo sforzo del team sia soggetto a sprechi. Inoltre, la metodologia Scrum prevede un continuo confronto con gli stakeholders che permette il rispetto dei criteri di qualità del software ed un approccio di lavoro che parte dalla realizzazione delle top features fino ad arrivare alle varie integrazioni.

Dal punto di vista del monitoraggio di progetto, questa metodologia permette al project manager di effettuare stime più precise dei tempi e delle risorse necessarie, garantendo un controllo più efficiente della schedulazione.

Tuttavia, essendo una metodologia basata sulle persone, la cooperazione del gruppo è fondamentale e una sua carenza può avere conseguenze gravi sulla buona riuscita del progetto. Inoltre, lo Scrum è più adatto a progetti di **piccole dimensioni** e di

complessità non troppo elevata, poiché per attività progettuali più grandi risulta più complicato valutare l'effort e il tempo necessari per la sua realizzazione.

1.5. LA METODOLOGIA PRINCE2

La metodologia **Prince2** (Projects IN Controlled Environments 2) è un marchio registrato dall'ufficio del Ministero del Tesoro del Regno Unito e si basa sulla gestione di progetti basata sui processi.

Questo modello presenta alcune caratteristiche peculiari che lo differenziano dalle altre metodologie e che ne hanno fatto uno dei tipi di IT project management più utilizzati in molti paesi.

Innanzitutto, un progetto gestito secondo Prince2 viene costantemente motivato dal punto di vista economico e, in base a questo tipo di valutazione, si decide se interrompere o portare avanti il lavoro. Ciascuna fase del progetto è di default un no-go e spetta al **Project Board**, ovvero al CdA del progetto, decidere se passare o meno alla fase seguente in base ad alcuni **key points** legati al processo di decision making.

Prima dell'inizio ufficiale del progetto, avviene un processo di **start up**, in cui viene verificata l'esistenza di prerequisiti come la presenza di una struttura adatta, di una forza lavoro competente e di tutta la documentazione necessaria. Inoltre, in questa fase sono identificati gli obiettivi da raggiungere e gli strumenti da utilizzare per riuscirci.

Un'altra caratteristica particolare di questa metodologia riguarda il processo di controllo e monitoraggio di progetto, che viene effettuato mediante il **Management by Exception**: a ciascun progetto vengono assegnati dei limiti di tolleranza legati a costi, tempi e qualità che devono essere rispettati in fase di realizzazione. Quando i driver del progetto si discostano da tali tolleranze, esso viene interrotto per valutarne l'impatto, individuare possibili contromisure da adottare e chiedere l'autorizzazione del Project Board per proseguire.

Dal punto di vista della programmazione, il Prince2 definisce diversi livelli di pianificazione, che vengono adattati alle dimensioni e alla complessità del progetto. Per questo motivo, si può affermare che la pianificazione sia **product-oriented** e che abbia come obiettivo il raggiungimento dei livelli qualitativi stabiliti nelle fasi tecniche del progetto stesso.

Inoltre, questa metodologia dà un'ampia importanza al processo di gestione dei rischi, che si basa sull'individuazione delle fasi critiche del ciclo di vita del progetto e su un approccio di analisi e raccolta dei dati utili a prevenire o a mitigare i rischi ad esse associati.

Per questo motivo, al triangolo tempi, costi e scope viene aggiunto un secondo triangolo legato a rischi, risorse e qualità, che va a formare una **stella**.

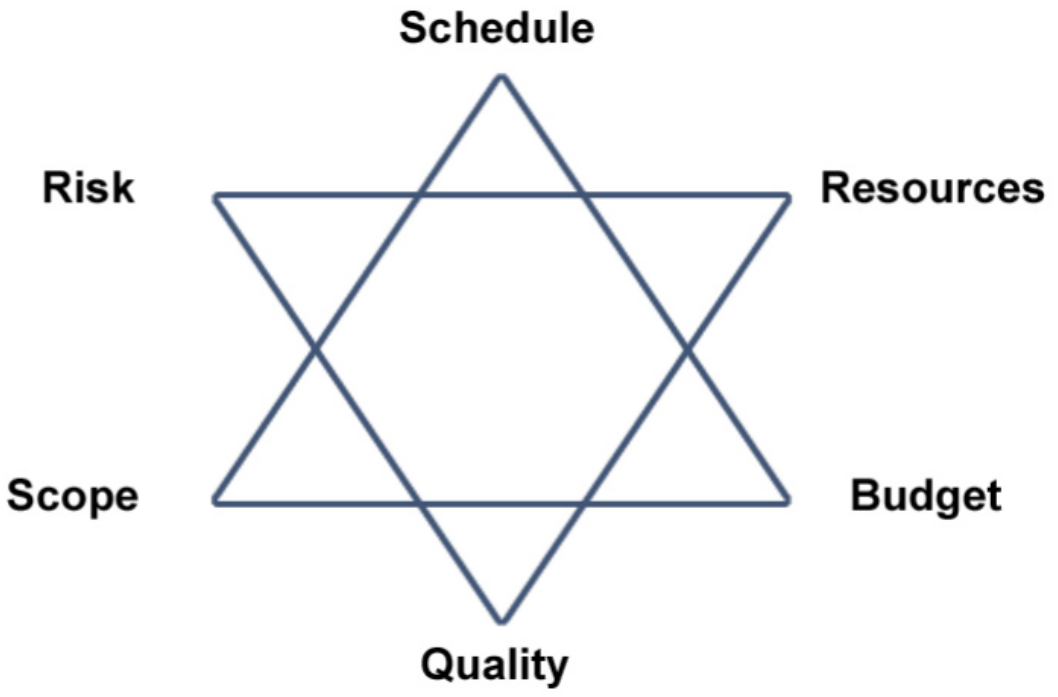

Figura 4. La stella della metodologia Prince2.

2. METODOLOGIE DI STIMA DEI COSTI E DEI TEMPI NELL'ITPM

Il costo legato al ciclo di vita di un sistema IT si divide in costi di progetto associati alla parte operativa e in costi di manutenzione che si presentano una volta che il programma è installato. Di seguito, una panoramica dei principali driver di costo legati ai progetti IT:

- C**osti di Project Management** associati alla gestione del progetto e alle attività di coordinamento e pianificazione dello stesso, a partire dall'identificazione delle specifiche tecniche fino ad arrivare al change management e al monitoraggio dello stato di avanzamento dei lavori.
- **Costi di implementazione tecnica** associati alla configurazione del sistema, all'integrazione di nuove funzionalità, alla migrazione dei dati, all'installazione in ambiente di test e all'utilizzo in parallelo del vecchio sistema e di quello nuovo.
- **Costi di operations e manutenzione** legati al pagamento di eventuali licenze, al supporto fornito agli user finali in caso di anomalie o malfunzionamenti del software e all'implementazione di upgrade del sistema che possono portare alla sostituzione di uno o più moduli vecchi con nuove versioni. Sono i costi maggiori legati ad un progetto IT.
- **Costi di procurement** legati all'analisi di fattibilità del progetto, all'acquisto dell'hardware, del software e delle strumentazioni necessarie alla programmazione.
- **Costi organizzativi** associati all'addestramento degli user all'utilizzo del nuovo programma, all'introduzione di cambiamenti all'interno dell'azienda e al pagamento del personale.

Al fine di poter effettuare una stima dei costi il più precisa possibile è fondamentale per le aziende definire in modo chiaro lo **scenario** del sistema, in particolare i **moduli** che lo compongono e come sono interconnessi, le **funzionalità** che ci si aspetta e quali caratteristiche possono essere soggette a un cambiamento.

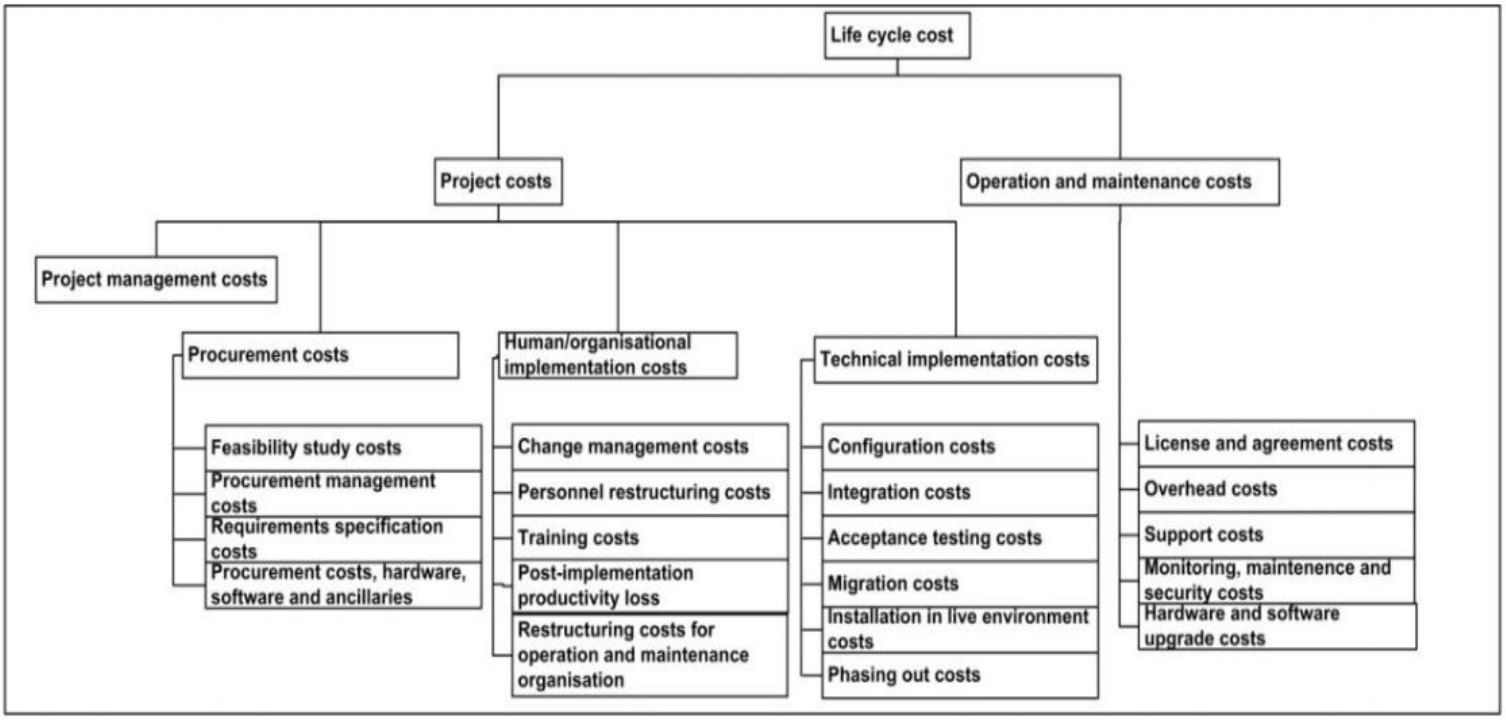

Figura 5. Tassonomia dei costi di un progetto IT.

Essere i primi a lanciare un prodotto IT sul mercato permette ai fornitori di software di ottenere maggiori ricavi e profitti, oltre che a ricevere un feedback anticipato da parte del mercato sulla direzione da prendere per gli sviluppi futuri. Allo stesso tempo, la rapida evoluzione tecnologica ha diminuito sensibilmente la durata del ciclo di vita dei prodotti IT e ha introdotto l'esigenza di accorciare i periodi di ricerca e sviluppo. Questa pressione legata al **time-to-market** e l'incertezza del settore fanno sì che il processo di stima di tempi, costi ed effort sia diventata una delle più grandi sfide che i project manager moderni devono affrontare. Infatti, stime troppo basse possono portare ad un **overrun** dei costi, mentre stime troppo alte possono causare la **perdita** di opportunità di guadagno.

Considerando che ogni anno vengono spesi più di **300 miliardi** di dollari per circa **250.000** progetti, quindi una media di **1.2 miliardi** di dollari per progetto, le implicazioni finanziarie non sono trascurabili. Secondo uno studio effettuato da Boehm, gli **errori** di stima oscillano in un range tra il **25% e il 400%** quando ci si trova all'inizio del progetto e ciò porta ad una varianza finanziaria che va dai **300.000 ai 4.8 milioni** di dollari.

La difficoltà nell'effettuare una stima accurata dei costi è spesso legata alla **mancanza di informazioni** dettagliate sulle caratteristiche del progetto nelle sue

fasi iniziali. Inoltre, si aggiungono problemi legati alla carenza di dati legati a progetti precedenti, ai **rapidi cambiamenti** del settore IT, alla difficoltà di definire requisiti precisi e alla **mancanza di esperienza** in progetti di un certo tipo. Tutto ciò fa sì che ci possano essere degli errori nel calcolo delle percentuali di completamento, delle deadline e delle metriche legate al progetto. Alla luce di queste considerazioni, l'ITPM abbandona il classico approccio di stima **bottom-up** in cui, a partire dai requisiti del progetto, viene prodotto un **diagramma di Gantt** che mostra tutti i **task** da completare e permette una stima del budget e dei tempi di completamento in base alle risorse disponibili.

L'approccio adottato diventa quindi di tipo **top-down**: si parte dalle componenti di alto livello della **WBS**, che vengono valutate per analogia rispetto ai progetti precedenti o in base a stime parametriche. Una volta terminata questa prima fase di stima, i valori ottenuti vengono spalmati sui **work packages** di livello inferiore fino ad arrivare alle singole attività che li compongono. Questa metodologia permette quindi di effettuare stime rapide senza la necessità di avere informazioni dettagliate, anche se si limita ad una visione di massima del progetto. Per questo motivo, le stime risultanti sono iterativamente verificate e raffinate man mano che il progetto procede. Ciò fa sì che il team di progetto sia in grado di reagire rapidamente alle difficoltà e ai cambiamenti del mercato.

Entrando maggiormente nel dettaglio, il PMBOK suddivide le metodologie di stima effettuata in fase di pianificazione in tre categorie:

- Stime basate sul giudizio di un pannello di esperti o su analogie con vecchi progetti (**metodo Delphi, planning poker**).
- Stime basate su modelli parametrici e matematici (**regressione, PERT, COCOMO**).
- Stime basate sull'utilizzo di tecniche di machine learning e intelligenza artificiale (**reti neurali**).

Ad oggi, nessuna di queste metodologie è totalmente efficiente dal punto di vista della **precisione** e dell'**applicabilità**, dal momento che è ancora molto presente l'intervento umano per interpretare i risultati e spesso i database utilizzati non sono

abbastanza completi. Tuttavia, si può affermare che il processo di stima dia risultati più affidabili man mano che il progetto va avanti.

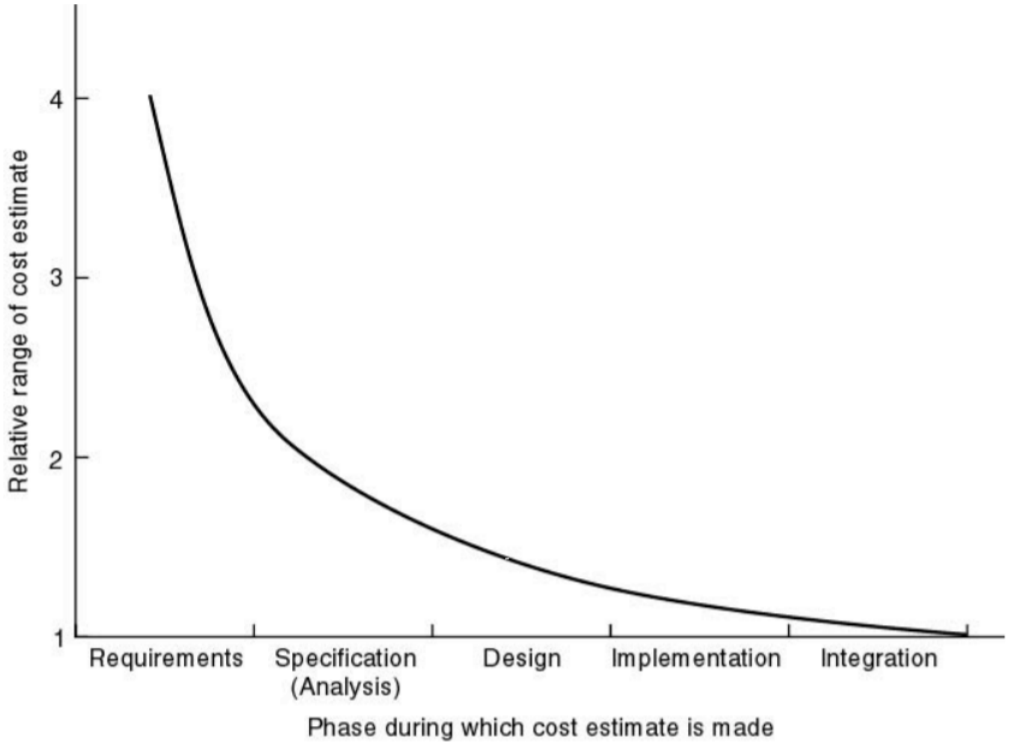

Grafico 2. Grafico su accuratezza stima in base a fase del progetto.

2.1. TECNICHE DI STIMA PER ANALOGIA

La stima per analogia si basa sulla raccolta di una serie di variabili in base alle quali si vuole fare la stima del progetto e sul confronto con progetti **simili** che sono già stati completati. Si può applicare sia a livello di sistema complessivo che di singoli work packages della WBS.

Questa metodologia serve ad effettuare delle valutazioni di massima nelle fasi precedenti all'inizio del progetto ed è costituita dai seguenti passaggi:

- Analisi dell'ambito del progetto e delle sue caratteristiche.
- Selezione di un insieme di progetti simili. Maggiore è la quantità di progetti dello stesso tipo, più la stima sarà accurata.
- Identificare eventuali differenze tra il progetto in partenza e quelli passati.
- Realizzazione della stima.

Sebbene sia rapida e si basi su dati reali, la stima per analogia presenta alcune criticità: innanzitutto, è necessario identificare con precisione le caratteristiche che descrivono al meglio il progetto, la cui disponibilità varia in base al momento in cui deve essere eseguita la stima. Inoltre, la determinazione dell'analogia tra progetti può essere un meccanismo **soggettivo** ed è necessario capire in anticipo quante analogie si stanno cercando al fine di evitare di restringere o allargare eccessivamente il portfolio preso in esame. Infine, è necessario capire come utilizzare i dati dei progetti analoghi per derivare la stima, cercando di utilizzare valori medi o intervalli di confidenza.

2.2. IL METODO DELPHI

Questo tipo di stima si basa sul coinvolgimento di **esperti esterni** alla squadra di progetto o all'organizzazione aziendale che abbiano delle competenze legate all'ambito in cui si vuole sviluppare il progetto.

Il **metodo Delphi** è utile per ottenere previsioni a lungo termine durante la fase di pianificazione di progetti in cui il team non possiede tutte le skills necessarie ad eseguire una stima o una valutazione efficaci. Esso è costituito da diverse fasi che vengono direttamente supervisionate dal project manager:

- Si procede con vari cicli di approfondimento utili ad isolare le questioni più rilevanti e approfondirle il più possibile.
- È cura del project manager garantire l'**anonimato** di ogni esperto in modo da preservarne l'indipendenza di giudizio e limitare l'influenza degli esperti più autorevoli sugli altri.
- Ogni esperto riceve dei **questionari** utili a raccogliere le informazioni in modo strutturato e in forma anonima.
- Il project manager e il team si occupano della distribuzione e della raccolta dei questionari, dell'analisi delle risposte e dell'identificazione delle aree di convergenza e di divergenza.
- Il processo viene ripetuto per approfondire le aree di divergenza per capirne le cause e le possibili modalità di superamento.
- Al termine della raccolta dei dati, viene creata una sintesi dei pareri degli esperti da condividere con gli stakeholders in fase di stima.

Questa metodologia permette di raccogliere utili esperienze di persone competenti in materia che possono essere applicate ai progetti presenti e futuri e garantisce una visione anonima e oggettiva della situazione. Tuttavia, essa richiede tempo per la preparazione dei questionari legati ai vari cicli, per la raccolta e per l'analisi delle informazioni. Inoltre, se il project manager non si occupa in modo efficiente della protezione dell'anonimato o se manipola le opinioni degli esperti per fare sì che rispecchino le sue convinzioni, può portare a risultati non obiettivi.

2.3. IL METODO PERT

Il metodo **PERT**, o stima a tre punti, si basa su uno schema **AOA** (Activity On Arrow) in cui le attività sono descritte da frecce che collegano i nodi che ne rappresentano la durata.

Questa tecnica è particolarmente indicata per progetti non facilmente prevedibili e in cui bisogna ragionare in **termini probabilistici**. Innanzitutto, il progetto viene suddiviso in milestones che rappresentano il punto di passaggio da una fase a quella successiva. Per effettuare la stima vengono presi in considerazione tre diversi valori:

- Un **valore ottimistico** (OT) ricavato da un'analisi top down o mediante uno studio di fattibilità, che rappresenta il best case scenario legato al completamento della milestone in esame.
- Un **valore pessimistico** (PE) derivato da un'analisi del team e che comprende elementi oggettivi combinati ad elementi di **contingency**. Rappresenta il worst case scenario legato al completamento della milestone in esame.
- Un **valore probabile** (PR) indipendente dagli altri due e ottenuto al termine di un'indagine fatta coinvolgendo esperti o terze parti.

Il risultato della stima si ottiene applicando la seguente formula matematica basata sui suddetti valori:

$$\mathbf{(OT + PE + 4PR)/6}$$

Questo approccio si rivela efficace quando sono disponibili pochi dati per la pianificazione, anche se il risultato è caratterizzato da condizioni di incertezza. Inoltre, permette di identificare il **percorso critico** e di fare ipotesi sulla data di completamento del progetto, favorendo la riduzione del rischio e l'ottimizzazione delle risorse.

Tuttavia, il metodo PERT può provocare un maggiore lavoro di ripianificazione soprattutto in situazioni dove ci sono rapidi e frequenti cambiamenti in corso d'opera. Per questo motivo, risulta generalmente poco efficace in ambito di IT.

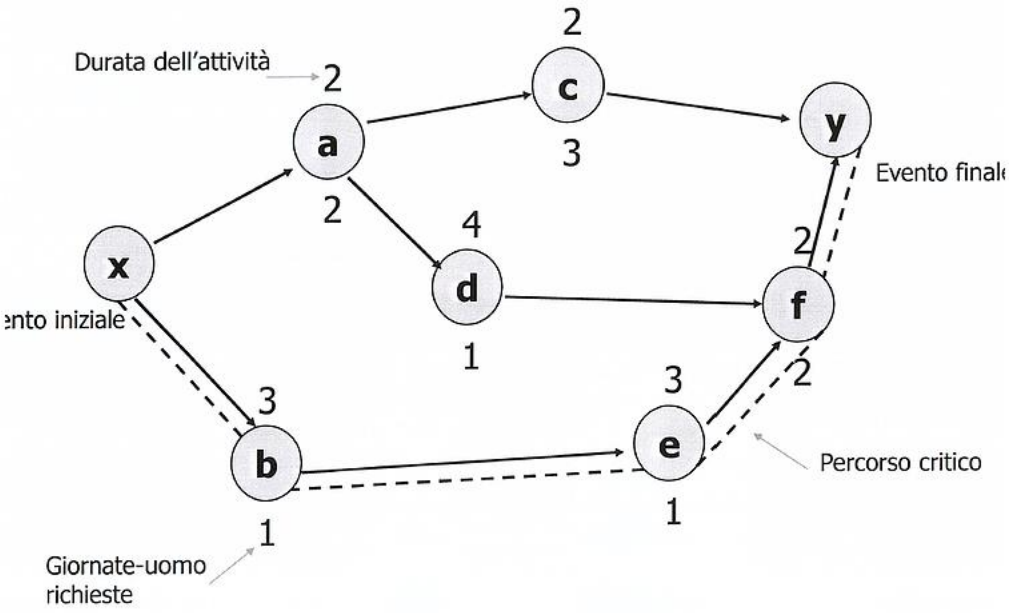

Figura 6. Esempio di diagramma di PERT.

2.4. IL METODO COCOMO

Il nome **COCOMO** è l'abbreviazione di **COnstructive Cost Model**, un modello matematico creato da Barry Boehm per effettuare la stima di alcuni parametri fondamentali per lo sviluppo di un software. Questo modello si basa su una **regressione** che considera vari parametri storici di progetto, che vengono pesati tramite una griglia di valutazione.

Questa metodologia si basa sul calcolo dell'equazione dell'effort, utile a stimare il numero di mesi-uomo necessari al progetto: **#mp*costo lavoro = costo stimato**. A partire da questo risultato, vengono poi calcolate le altre grandezze da stimare (requisiti, manutenzione, ecc..).

Esistono 3 diverse tipologie di modello legate a questo metodo, che si differenziano tra loro per l'accuratezza con cui vengono stimati i vari valori. Essi si basano sull'equazione **Effort = a*S^b*EAF** dove l'Effort rappresenta lo sforzo in mesi-persona, EAF è il coefficiente di assestamento, S è la dimensione del software da sviluppare contata in migliaia di linee di codice (KLOC), a e b sono coefficienti che dipendono dal tipo di progetto. Di seguito, sono presentate le caratteristiche dei 3 modelli:

- **Basic**: è il meno preciso, generalmente viene applicato all'inizio del ciclo di vita del progetto ed è adatto a progetti di tipo **Organic**, ovvero di piccole dimensioni e con problematiche ben note al team di sviluppo.
- **Intermediate**: è quello più usato, generalmente viene applicato dopo che sono stati specificati i requisiti, basa il suo calcolo su un insieme di **cost driver** legati agli attributi di progetto (skills del personale, complessità prodotto, tempo di sviluppo, limiti dell'hardware) a cui vengono assegnati dei valori soggettivi e sulla grandezza del programma in KLOC. I fattori moltiplicativi legati a tali attributi indicano il loro scostamento da un valore considerato normale e vengono calcolati in base ai progetti passati. Questo modello si adatta bene a progetti di tipo **Semi-detached** legati ad applicazioni più complesse e per le quali l'esperienza del team è limitata.
- **Advanced**: generalmente viene applicato al termine della fase di design, aggiunge una serie di **cost driver** pesati per ciascuna fase del progetto (requisiti e progettazione di alto livello, progetto dettagliato, codice e test, integrazioni e test) e li moltiplica per il precedente valore di EAF. Si adatta a progetti di tipo **Embedded** legati ad applicazioni di grandi dimensioni con forti vincoli di costi e di tempi, in cui l'esperienza del team è limitata.

Coloro che hanno introdotto il metodo COCOMO si sono basati sulla loro esperienza e su alcuni dati storici che hanno raccolto. Di conseguenza, per poter utilizzare questa metodologia, è necessario che l'azienda la tari in base alla sua realtà. Inoltre, gli attributi che vengono presi in considerazione sono spesso di ardua valutazione e difficilmente ottenibili dalle aziende e ciò rappresenta un notevole limite al suo utilizzo.

Per concludere, il modello COCOMO risulta efficace nel predire lo sforzo di sviluppo considerando fattori relativi al progetto, al personale e all'hardware. Tuttavia, è opportuno utilizzare contemporaneamente altre tecniche più accurate nella stima dei costi e dei tempi al fine di ottenere un risultato più affidabile possibile.

3. MONITORAGGIO E CONTROLLO: LE STIME PREDITTIVE

Le metodologie di stima presentate finora sono utilizzate in fase di pianificazione e programmazione del progetto. Tuttavia, a prescindere dall'attenzione con cui sono portate avanti le suddette fasi, spesso la fase di sviluppo non rimane in linea con quanto predetto a causa di molteplici variabili che possono entrare in gioco in corso d'opera e che inizialmente erano ignote.

Nella fase di **monitoraggio e controllo** di un progetto, si tiene conto dei fattori di incertezza e si cerca di eliminarli o di ridurne gli effetti, implementando dei processi utili ad avere un controllo costante sullo stato di avanzamento dei lavori e ad effettuare eventuali azioni correttive in corso d'opera finalizzati al rispetto dei requisiti legati ai tempi, ai costi e alla qualità del progetto. Per questo motivo, l'adozione di metodologie efficienti per effettuare delle **stime a finire** del progetto risulta fondamentale per il project manager e per il team.

Di conseguenza, è importante che il team stabilisca degli **indicatori** di performance, dei processi di raccolta dati e delle **metriche** condivise in modo da poter identificare eventuali **scostamenti** critici rispetto alla pianificazione che possono impattare i costi o i tempi di completamento. Dall'analisi di questi scostamenti si possono individuare le conseguenze sulla stima a finire del progetto e ricercare possibili azioni correttive basate su una filosofia **Performance-Driven** o **Target-Driven**.

A livello operativo, lo strumento utilizzato per fare il confronto tra la pianificazione e l'effettivo andamento del progetto è rappresentato dalle **curve a S**: sull'ascissa è riportato il tempo e sull'ordinata è riportata la variabile che si sta monitorando. Per esempio, se si vuole andare ad analizzare l'andamento del costo di progetto, ad ogni istante di tempo sono associati un budget a disposizione e il costo effettivo sostenuto e il delta tra questi due valori rappresenta lo scostamento, che può essere positivo o negativo. Le variabili di progetto presentano un andamento a S poichè, nel periodo centrale, si ha una crescita maggiore rispetto alla fase iniziale in cui i

processi non hanno ancora raggiunto la massima efficacia e alla fase finale in cui la maggior parte del lavoro è conclusa.

Dal punto di vista formale, la curva a S è descritta dall'equazione matematica $\boldsymbol{X = \frac{K}{(1+ce^{-ht})}}$, dove K rappresenta la capacità di carico, c una costante funzione dei valori iniziali, h il tasso di crescita e t il tempo. Tuttavia, il budget preventivato e il costo effettivo non sono sufficienti a stabilire se il costo del progetto sia superiore o inferiore a quanto pianificato e lo stesso discorso si può fare per i tempi. Per questo motivo, ai due valori già citati se ne aggiunge un terzo detto Earned Value.

3.1. IL METODO DELL'EARNED VALUE

Il metodo dell'**Earned Value** permette l'analisi dello stato di avanzamento del progetto, l'identificazione degli scostamenti rispetto alla pianificazione ed il calcolo delle stime a finire legate ai costi e ai tempi finali. Questo metodo si basa sui seguenti concetti:

- Il **Planned Value** (PV, BCWS), ovvero il budget a disposizione programmato per ogni periodo, è dato dal prodotto tra la quantità di lavoro programmata per il periodo e il costo delle risorse coinvolte.

$$\mathbf{BCWS = Budget\ Cost\ x\ Work\ Schedule}$$

- L'**Actual Cost** (AC, ACWP), ovvero il costo effettivamente sostenuto per ogni periodo, è dato dal prodotto tra la quantità di lavoro effettivamente compiuta e il costo reale che si è sostenuto nel periodo in esame.

$$\mathbf{ACWP = Actual\ Cost\ x\ Work\ Performed}$$

Lo scostamento tra BCWS ed ACWP (**scostamento contabile = BCWS – ACWP**) misura la differenza tra ciò che si sarebbe dovuto spendere e quanto si è realmente speso, senza però fornire una reale indicazione di quanto si è realmente prodotto in termini di stato di avanzamento del progetto, risultato che si raggiunge grazie all'utilizzo dell'Earned Value.

- L'**Earned Value** (EV, BCWP) rappresenta il lavoro effettivamente eseguito utilizzando il budget pianificato inizialmente ed è dato dal prodotto tra il budget pianificato e il lavoro effettuato nel periodo in esame.

$$\mathbf{BCWP} = \mathbf{Budget\ Cost\ x\ Work\ Performed}$$

  Questo valore permette di individuare eventuali anticipi o ritardi nei tempi e di vedere se si stanno sostenendo costi maggiori o minori di quelli pianificati.

Per essere calcolato, l'Earned Value richiede la percentuale di avanzamento effettiva del lavoro, che può essere identificata in base alle caratteristiche del progetto. Per quanto riguarda i progetti di tipo **IT** presi in esame in sede di tirocinio, le due metodologie utilizzate sono le seguenti:

- Per quanto riguarda il monitoraggio della fase di sviluppo del software è usata la tecnica dell'**ON/OFF**, in cui lo stato viene cambiato da OFF a ON solo quando l'attività viene completata. A ciascun work package vengono assegnati degli **story points** a partire dai quali vengono calcolati i giorni necessari alla loro realizzazione. A partire dai valori ottenuti, si ricavano la percentuale di avanzamento della fase di sviluppo e quella dell'intero progetto.

$$\%\ \mathbf{avanzamento\ Sviluppo\ SW} = \mathbf{SP(t)/SP(Tot)}$$

$$\%\ \mathbf{avanzamento\ Progetto} = \%\ \mathbf{avanzamento\ Sviluppo\ SW} * \mathbf{P(Sviluppo\ SW)}$$

  dove P(Sviluppo SW) rappresenta il peso dell'attività di sviluppo SW rispetto al complesso delle attività di progetto.

- La seconda metodologia utilizzata è basata sull'identificazione delle **milestone** di progetto, a cui viene assegnato un peso legato allo sforzo necessario per raggiungerla. A partire da questi valori, si ricava la percentuale di avanzamento del progetto.

$$\% \textbf{ avanzamento progetto } = \sum_{\mathbf{i=1}}^{\mathbf{n}} \mathbf{Mi} * \mathbf{Pi}$$

dove Mi rappresenta la milestone legata all'attività i-esima ed è una variabile booleana ($Mi = 1$ se fase è completata, altrimenti $Mi = 0$), mentre Pi rappresenta il peso della fase in esame rispetto all'intera attività.

Una volta determinata la percentuale di completamento del progetto, sarà possibile calcolare l'Earned Value legato all'attività.

A partire dalle variabili appena definite, si possono calcolare i seguenti indici di performance legati all'analisi degli scostamenti dal punto di vista dei costi e dei tempi:

- **La Cost Variance (CV = BCWP – ACWP).** Se è minore di zero significa che il costo del progetto è maggiore di quello pianificato, se è maggiore di zero significa che si sta risparmiando sul budget.
- Il **Cost Performance Index (CPI = BCWP/ACWP).** Se è maggiore di uno significa che si sta spendendo meno di quanto pianificato, se è minore di uno significa che i costi sono maggiori del budget.
- La **Schedule Variance (SV = BCWP – BCWS).** Se è minore di zero significa che si è in ritardo rispetto alla pianificazione, se è maggiore di zero significa che si è in anticipo.
- Lo **Schedule Performance Index (SPI = BCWP/BCWS).** Se è maggiore di uno significa che il tempo di completamento del progetto sarà minore rispetto a quello pianificato, se è minore di uno significa che il progetto finirà in ritardo.

Basandosi contemporaneamente sui parametri progettuali e sugli indici di performance, è possibile calcolare le stime a finire relative ai tempi ed i costi di progetto.

3.1.1 CEAC: COST ESTIMATION AT COMPLETION

La **Cost Estimate at completion** (EAC o CEAC) viene utilizzata per realizzare stime a finire dei costi totali di un progetto. Esistono due diversi approcci, original e revise, le cui formule per effettuare il calcolo sono le seguenti:

- **Original estimate approach**:

$$\mathbf{EAC} = \mathbf{ACWP} + (\mathbf{BAC} - \mathbf{BCWP}) = \mathbf{BAC} - (\mathbf{BCWP} - \mathbf{ACWP})$$
$$= \mathbf{BAC} - \mathbf{CV}$$

  Il **BAC** (Budget at Completion) rappresenta il budget totale di progetto. Tale approccio si basa sull'assunzione che, a prescindere da come sia andato fino al momento dell'analisi, il progetto avrà il costo preventivato secondo il Budget Value. Di conseguenza, lo scostamento totale di costo si ricava semplicemente sottraendo la CV al BAC. La criticità di questo modello è che esso assume che, qualunque sia stata la produttività fino al momento dell'analisi, il trend applicato fino alla fine del progetto sarà quello dettato dal budget preventivo.

- **Revise estimate appproach**:

$$\mathbf{EAC} = \mathbf{ACWP} + (\mathbf{BAC} - \mathbf{BCWP}) / \mathbf{CI} = \mathbf{BAC} / \mathbf{CI}$$

  Questo approccio si basa sull'assunzione che la produttività delle risorse dedicate al progetto al momento dell'analisi rimarrà costante fino alla fine. Partendo, quindi, dal **BAC** come riferimento di costo totale, un **CI** minore di uno (costi maggiori del previsto) porterà ad un **EAC** maggiore del BAC. Viceversa, un CI maggiore di uno farà sì che l'EAC sia minore del BAC.

- **Approccio pessimistico**:

$$\mathbf{EAC} = \mathbf{BAC} / (\mathbf{SI} * \mathbf{CI})$$

Quest'ultimo approccio tiene conto sia dell'effetto di un eventuale ritardo/anticipo dal punto di vista dei tempi sia di quello legato ad un eventuale guadagno/perdita dal punto di vista dei costi. Quando SI e CI sono contemporaneamente maggiori o minori di uno, il loro effetto sull'EAC si amplifica mentre, se presentano una tendenza opposta, i loro effetti si smorzano a vicenda.

La CEAC basata sugli indici dell'EV è considerata una tecnica affidabile per formulare stime predittive dei costi. Negli anni, questa metodologia è stata perfezionata unendo il CPI standard con elementi di statistica che permettono l'identificazione di intervalli di confidenza all'interno dei quali deve ricadere la previsione (**Lipke, Zwikael, K. e Anbari**, 2009). Un ulteriore passo nell'evoluzione della CEAC è stato fatto integrando la simulazione di Montecarlo nel processo di stima dei costi (**Naeini e Heravi**, 2011): i dati riguardanti l'Actual Cost venivano inseriti nel processo al fine di simulare lo sviluppo futuro delle curve a S ed ottenere la stima. Un altro contributo importante alla rifinitura della metodologia CEAC è stato dato da **De Marco e Narbaev** (2014, 2013), che hanno introdotto la prospettiva secondo cui le performance del progetto dal punto di vista dei tempi hanno una forte influenza sulla stima finale dei costi. In particolare, Narbaev e De Marco hanno utilizzato diverse distribuzioni per rappresentare i costi (Gompertz, Weibull, modello di Bass) per trovare una combinazione tra i dati di AC e di PV ed ottenere delle curve che rappresentassero i costi cumulati del progetto in particolare nelle sue fasi iniziali.

3.1.2 TEAC: TIME ESTIMATION AT COMPLETION

La **Time Estimate at completion** (AC o TEAC) viene utilizzata per realizzare stime a finire dei tempi e presenta a sua volta due diversi approcci, le cui formule sono le seguenti:

- **Orignal estimate approach**:

$$\mathbf{AC} = \mathbf{T} + (\mathbf{BAC} - \mathbf{BCWP}) * (\mathbf{BC} - \mathbf{T}) / (\mathbf{BAC} - \mathbf{BCWS})$$

Il **BC** rappresenta il tempo preventivato a budget per il completamento. La data di completamento è ricavata aggiungendo alla data in cui viene fatta l'analisi ($\mathbf{T}$, **time now**) la quota di budget ancora a disposizione, moltiplicata per la produttività effettiva riscontrata fino a quel momento.

- **Revise estimate approach**:

$$\mathbf{AC} = \mathbf{T} + (\mathbf{BAC} - \mathbf{BCWP}) * (\mathbf{BC} - \mathbf{T}) / ((\mathbf{BAC} - \mathbf{BCWS}) * \mathbf{SI})$$

In questo approccio, si tiene conto anche del SI: se è minore di uno, significa che il progetto si trova in una situazione di inefficienza e, di conseguenza, i tempi previsti di completamento si allungano.

3.2 IL METODO DELL'EARNED SCHEDULE

Il metodo dell'**Earned Schedule** (ES) è stato introdotto da Walter Lipke nel 2003 e, a partire dal 2005, è stato adottato anche dal PMI. Esso rappresenta una variante rispetto al classico metodo dell'Earned Value, in quanto migliora la predittività delle stime dei tempi senza aver bisogno della raccolta di ulteriori dati rispetto a quelli utilizzati nel classico EV. L'ES, infatti, permette di trasformare le metriche dell'EV in metriche temporali al fine di valutare meglio le prestazioni legate alla pianificazione del progetto e la durata necessaria al suo completamento. La formula dell'ES è la seguente:

$$\mathbf{ES} = \mathbf{c} + \frac{\mathbf{EV} - \mathbf{PV(c)}}{\mathbf{PV(c+1)} - \mathbf{PV(c)}}$$

Il parametro c rappresenta il numero di intervalli temporali in cui risulta un EV maggiore o uguale al PV. Inoltre, il SV classico viene sostituito dal nuovo indice **SV(t) = ES – AT**, dove AT è l'Actual Time, mentre il SPI classico viene sostituito con il nuovo indice **SPI(t) = ES/AT**.

Ad oggi, questa metodologia è considerata migliore dell'EV nell'effettuare la previsione della durata del progetto finale. Infatti, mentre l'EV fornisce una stima di una probabile data di fine del progetto, l'ES permette una visione della probabilità di completamento in precisi momenti nel tempo senza un eccessivo sforzo aggiuntivo. Per questo motivo, spesso le due metodologie vengono integrate con limiti di confidenza statistica al fine di ricavare risultati probabili riguardo al costo finale e alla durata del progetto.

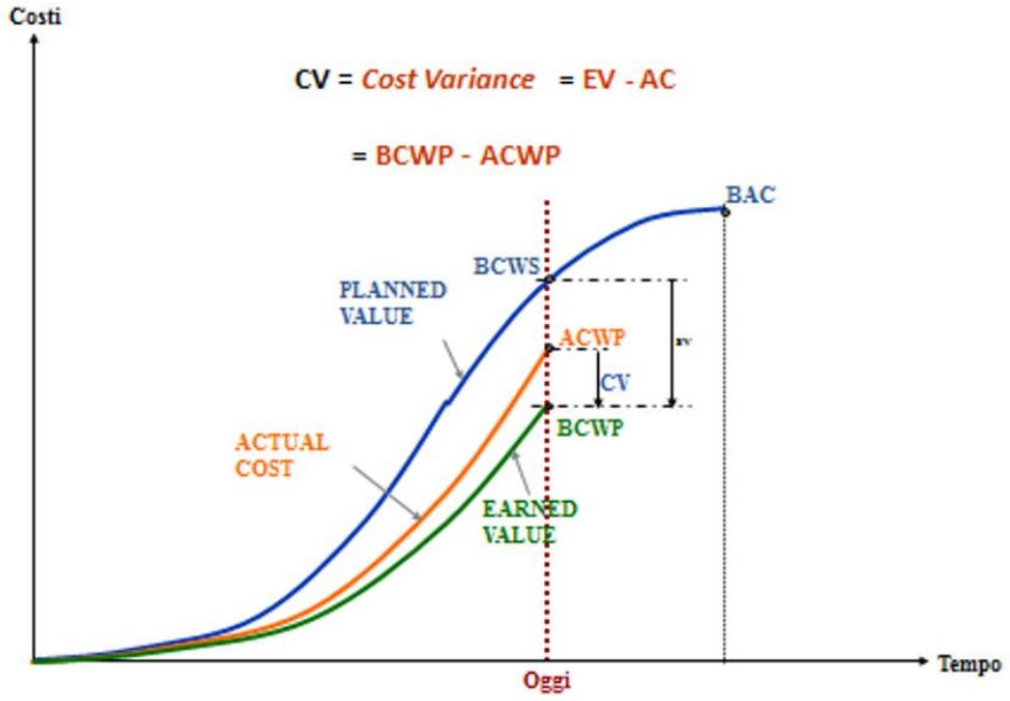

Grafico 3. Curva a S dell'EAC.

3.3 STIME A FINIRE NON LINEARI

Il team di progetto può applicare il CEAC e il TEAC in qualsiasi fase di sviluppo del progetto, utilizzando gli indici di performance dei tempi e dei costi forniti dal metodo EV. Tuttavia, l'accuratezza e l'affidabilità dell'utilizzo di tali indici è limitata, dal momento che la maggior parte dei progetti presenta delle **curve a S non lineari** mentre l'intero EV si basa su assunzioni lineari.

Infatti, le metodologie dell'EV e dell'ES si fondano su una costruzione geometrica basata sul punto di intersezione tra la curva dell'**Earned Value** e quella del **Planned Value** in corrispondenza del tempo corrente. Andando ad analizzare la formula della durata del progetto, si può quindi affermare che essa sia **lineare** e si basi sulle **curve cumulate** dell'Earned Value e del Planned Value. Tuttavia, dal momento che le curve dei costi di progetti reali hanno un andamento **non lineare**, alcuni studiosi hanno lanciato l'idea di combinare la generale formula matematica dell'ES con dei modelli non lineari di crescita dei costi (**Narbaev e De Marco**, 2014, **Warburton**, 2014). Per ovviare a questo problema, nel 2016 **Warburton e Cioffi** hanno applicato la formula dell'ES a dei progetti con costi non lineari e hanno dimostrato come derivare la corretta espressione per stimare la durata finale di progetto.

Il modello del **TEAC** si basa sull'approccio standard dell'EV: se non si ottengono risultati diversi da quelli pianificati, l'Earned Value al termine del progetto non è né minore né maggiore del valore preventivato inizialmente e tutto il lavoro pianificato è stato completato. Inoltre, da un punto di vista empirico, se la struttura del progetto non cambia è improbabile che ci siano altre variazioni durante la sua realizzazione e ciò fa sì che le curve dell'EV e del PV presentino la stessa forma. Questa ipotesi è stata avvalorata da molteplici studi secondo i quali le curve cumulate dell'EV e del PV seguono entrambe la forma della curva di Putnam-Norden-Rayleigh (**PNR**) (**Warburton**, 1983, **Basili e Beane**, 1981, **Lee**, 2002, **Gallagher e Lee**, 1996) ed è diventata una delle idee alla base dell'EV. Infatti, le formule standard del TEAC e del CEAC si basano sull'assunzione che le curve dell'EV e del PV siano entrambe lineari, seppure con valori di pendenza differenti.

Quando, invece, si vanno ad analizzare curve dei costi non lineari, è fondamentale specificare quali siano i parametri costanti e quali siano quelli soggetti a variazioni. Se si decide di rappresentare le due curve con una curva di **Gompertz**, la formula che si ottiene è la seguente:

$$\mathbf{G(t) = \alpha * e - e^{*(\beta - \gamma t)}}$$

Il parametro **α** rappresenta l'asintoto per G(t) che tende a infinito, **β** è l'intercetta sull'asse y ed è generalmente un valore basso, **γ** è il tasso di crescita e il prodotto **αe** rappresenta il costo finale pianificato. Dal momento che il valore totale guadagnato è pari al valore totale pianificato, l'asintoto della curva dell'EV sarà uguale a quello della curva del PV (**αe = αp**). Se questa condizione non è rispettata si va incontro a delle stime dei tempi errate, in particolare nelle fasi iniziali del progetto in cui si ha a disposizione una quantità limitata di dati. Inoltre, una variazione di αe, corrispondente ad un'analoga variazione di αp, indica un'evoluzione nell'andamento della curva dell'EV.

La crescita dei dati relativi al PV è rappresentata dal prodotto **γp** mentre quella dei dati relativi all'EV dal prodotto **γe**. Per un progetto che prosegue più rapidamente o più lentamente, il secondo valore è diverso dal primo, ma tipicamente i due valori sono uguali (**γp = γe**). Dal momento che il parametro α influenza la curva solo in direzione verticale, l'aspetto davvero interessante delle curve cumulate dei costi è rappresentato dal parametro γ.

Nella seguente analisi effettuata da Narbaev e De Marco, si indicheranno i dati relativi ai costi attuali all'istante ti come **Cp(ti)** e quelli relativi all'Earned Value come **Ce(ti)**. Inoltre, si definiscono l'istante di fine pianificato per il progetto come T1 e si assume che, in caso di ritardo, il progetto terminerà all'istante T1'>T1 mentre, in caso di anticipo, terminerà all'istante T1'<T1. Il costo totale pianificato verrà indicato come **Cp(T1)**.

Si parte dall'assunzione che i dati relativi al PV sono disponibili prima dell'inizio del progetto, mentre quelli relativi all'EV sono disponibili solo in tempo reale e vengono utilizzati all'inizio, durante e al termine del progetto, in modo da poter analizzare la precisione della stima finale della durata del progetto.

3.3.1 LA METODOLOGIA AC-PV FIT

Narbaev e De Marco hanno sviluppato un metodo per calcolare il **CEAC** integrando l'approccio standard relativo all'**ES** con dei modelli di crescita che utilizzano un'analisi regressiva non lineare per ottenere la corrispondenza tra i costi pianificati e quelli effettivi. Questa corrispondenza è denominata **AC-PV fit**.

La stima della durata del progetto permette il calcolo di un fattore di completamento (**CF**), che si può definire come il rapporto tra la stima a finire del progetto e la durata pianificata. Di conseguenza, il CF non è altro che l'inverso del SPI: se è maggiore di uno il progetto è in ritardo, se è minore di uno il progetto è in anticipo. A partire dal CF, si può effettuare una previsione sul costo finale mediante la seguente formula:

$$\mathbf{CEAC = Ca(t) + (GMM[CF] - GGM[t]) * BAC}$$

dove il parametro Ca(t) rappresenta il costo effettivo all'istante t, il BAC rappresenta il budget pianificato, GMM e GGM sono parametri legati al **Gompertz growth model** e al **generalized method of moments,** che vengono utilizzati per fare stime non lineari. Il CF calcolato all'istante t rappresenta la funzione di crescita non lineare nella regressione.

Al fine di parametrizzare un particolare GGM, Narbaev e De Marco hanno fatto combaciare i dati disponibili riguardanti i costi effettivi fino al presente con i dati riguardanti i costi pianificati fino alla fine del progetto. L'obiettivo di questa metodologia AC-PV fit è quella di ottenere un punto GGM[CF] il più vicino possibile all'effettivo CEAC.

Questa tecnica si è dimostrata efficace su una vasta gamma di progetti, ma non fornisce stime affidabili in situazioni in cui un progetto viene ritardato in modo significativo o è soggetto ad un notevole superamento dei costi pianificati. Infatti, in una situazione simile la differenza tra l'ultimo Ca(t) disponibile e il Cp(t) rischia di essere troppo ampia, rendendo la stima errata.

Quando il progetto è soggetto ad un significativo eccesso di costo si ottiene una curva dei costi effettivi Ga(t) mediante l'utilizzo di un GGM con parametri **αa, βa** e **γa** dove il parametro αa, che rappresenta l'asintoto della curva dei costi effettivi, deve essere diverso da αp. Di conseguenza, viene ricavato un nuovo asintoto αa mediante il GGM e, successivamente, il software calcola i nuovi valori dei parametri **αa, βa** e **γa** basandosi sui dati relativi al costo effettivo del progetto all'istante corrente.

Quando, invece, il progetto è soggetto ad un significativo ritardo, la curva dell'Earned Value sarà Ge(t), ovvero un GGM con parametri **αe, βe γe.** Il vincolo legato al fatto che le due curve hanno la stessa forma fa sì che **αp = αe** e **βp = βe,** mentre **γe** rappresenta la diversa crescita della curva dell'Earned Value dovuta al ritardo. Basandosi sull'assunzione che nelle fasi conclusive del progetto la curva di crescita tende ad appiattirsi e che il valore Cp(CF) può essere considerato come una stima a finire dei costi approssimata, αa può essere calcolato con la seguente formula:

$$\alpha^*a = Cp(CF) / Cp(T1)$$

dove Cp(T1) rappresenta il costo pianificato in base alla schedulazione originale e Cp(CF) è il valore della curva del PV, Gp(t), nel punto CF.

Basandosi sulle equazioni presentate in precedenza, è possibile calcolare il TEAC utilizzando il modello di crescita di Gompertz utilizzando uno dei tre seguenti approcci:

- **Stima standard**: si basa sull'approccio standard di Warburton e Cioffi e sull'assunzione che il TEAC rimane costante durante tutto il progetto.

$$T1' = T1(t)/ES(t)$$

- **Stima puntuale**: ad ogni istante t, la curva del PV è definita come Gp(αp, βp, γp, t). Inoltre, si assume che la curva dell'EV segua una curva GGM con gli stessi valori di βp e αp, ma con un valore diverso di γe. Usando il ritardo δ(t), si ottiene la curva Ge[αp, βp, γp, ti − δ(ti)] = Ce(ti) da cui si calcola l'ES.

$$ES(ti) = (1/\gamma) * (\beta - \ln[-\ln\{Ce(ti) / \alpha\}])$$

Ad ogni istante ti, si utilizza il corrispondente dato Ce(ti) per calcolare il singolo valore di γe:

$$\gamma e = (1 / ti) * (\beta p - \ln [-\ln\{Ce(ti) / \alpha p\}])$$

Dal momento che il valore γe dipende da quello dell'Earned Value nel singolo punto in esame, la stima della durata si definisce puntuale ed è data dalla seguente formula:

$$T1' = (1 / \gamma e) * (\beta p - \ln[-\ln\{1 / \alpha p\}])$$

Delle tre metodologie proposte, la stima puntuale risulta essere la migliore per effettuare stime dei tempi nelle fasi iniziali del progetto rispetto alle classiche metodologie basate sugli indici dell'EV.

- **Stima cumulativa:** si utilizzano tutti i valori di Ce(ti) calcolati fino all'istante corrente e i vincoli rimangono gli stessi. Si ottiene una curva Ce(t)= G (αp, βp, γe, t) e il TEAC viene calcolato con la stessa formula della stima puntuale.

Per verificare l'accuratezza di questo metodo di stima, Narbaev e De Marco hanno utilizzato due indicatori chiave: il **Mean Absolute Percentage Error** (MAPE) e il **Percentage Error** (PE).

$$PE = [(T1' - T1) / T1] * 100 \qquad MAPE = \sum_{t=i}^{t=i+n} \frac{|PEt|}{n}$$

Il MAPE è utilizzato per comparare le performance delle metodologie di stima usate all'inizio, nel mezzo e alla fine del progetto ed è costituito da una serie di osservazioni effettuate in unità temporali consecutive utilizzando n punti dati. Dall'applicazione di questa metodologia di stima ad un portfolio di progetti risulta che essa si riveli più efficace man mano che il progetto si avvia verso la sua conclusione, anche se i risultati possono essere influenzati dall'imprevedibilità dei dati di alcuni progetti, dalla debolezza del modello utilizzato, dalla relazione tra i dati a disposizione e il profilo utilizzato per rappresentarli e dall'accuratezza del processo di fit delle due curve. Tuttavia, pur non esistendo una metodologia di stima a finire che sia migliore delle altre per qualunque tipo di progetto, Narbaev e De

Marco hanno dimostrato che l'accuratezza del TEAC aumenta se si tiene conto dell'andamento non lineare del profilo dei costi e, di conseguenza, anche il CEAC basato su formule legate alla schedulazione fornisce risultati più affidabili.

Le metodologie non lineari proposte non forniscono sempre risultati migliori rispetto a quelle lineari: infatti, l'approccio lineare fornisce stime accettabili per un'ampia gamma di progetti nonostante abbiano una curva dei costi non lineare. Questo perché la classica formula dell'ES risulta essere la stessa anche per molte curve dei costi non lineari.

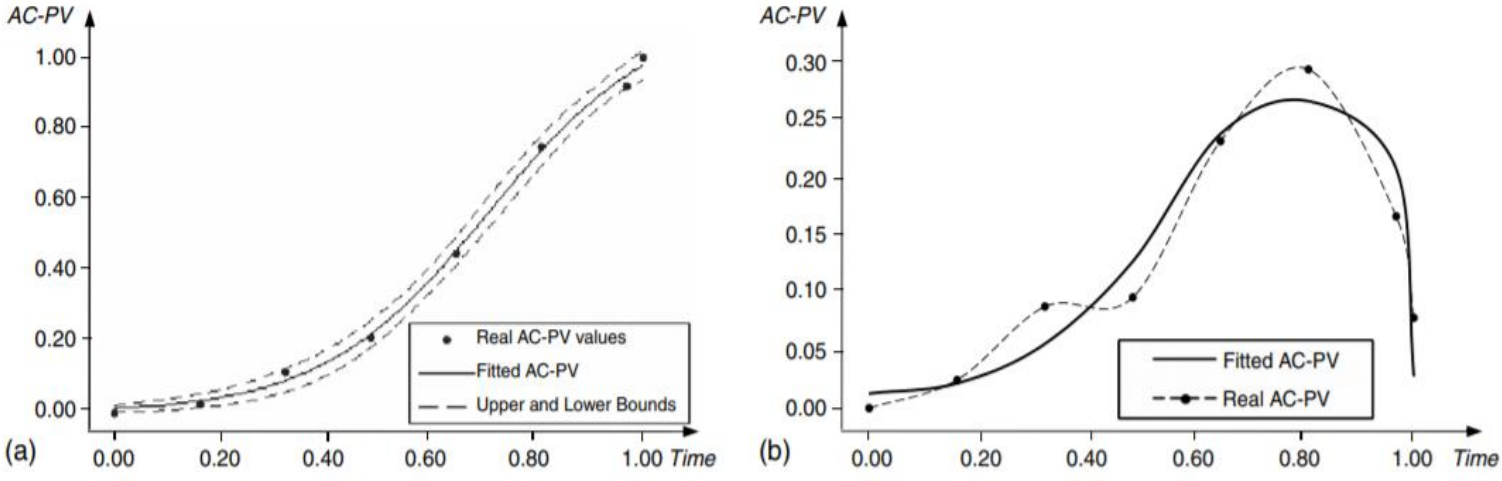

Grafico 4. Grafico AC-PV fit.

4. STIME PREDITTIVE MEDIANTE TECNICHE DI MACHINE LEARNING

Come sottolineato in precedenza, un progetto **software** è più difficile da monitorare e controllare dal punto di vista dei tempi e dei costi rispetto ad un progetto che ha come output un prodotto fisico come, ad esempio, un edificio. Per questo motivo, la pianificazione di un progetto software è una delle attività più critiche dell'intero processo ed eventuali carenze rischiano di generare inefficienza nella sua gestione. Per evitare criticità durante l'avanzamento del progetto, diventa quindi fondamentale disporre di un **dataset** storico con le informazioni riguardanti l'effort, la durata e i costi dei progetti passati al fine di sviluppare dei modelli predittivi efficaci. Da un punto di vista ideale, l'utilizzo di metodologie di stima basate su algoritmi di **machine learning** permetterebbe di controllare e di diminuire in modo significativo i costi associati ai progetti software. Tuttavia, le maggiori criticità legate a questo tipo di progetti sono la scarsità di dati empirici e la difficoltà nel costruire un **framework** ben definito con relazioni ben chiare tra i **descrittori** chiave. Inoltre, le metodologie di stima dei costi più utilizzate si basano su un approccio top-down (project-based invece che product-based) che rende più difficile la raccolta di dati rispetto ad un approccio bottom-up in cui le metriche si basano direttamente sul prodotto. Per questi motivi, ad oggi, le metodologie di machine learning sono scarsamente utilizzate a livello professionale e non risultano ancora abbastanza affidabili, soprattutto nelle fasi iniziali del ciclo di vita del software.

Al fine di sviluppare una metodologia di stima predittiva, è necessario utilizzare una funzione predittiva molto precisa che, a partire dai dati riguardanti i progetti passati, permetta di stimare l'effort e la durata dei progetti futuri. Questa capacità dell'algoritmo di imparare dai dati osservati rappresenta un grande vantaggio nella formulazione di una stima legata a progetti caratterizzati da un gran numero di relazioni complesse. Di seguito, saranno presentate le metodologie di machine learning più utilizzate.

4.1. FEED-FORWARD NEURAL NETWORKS

Una **rete neurale** (NN) è un modello di computazione e di processo di segnali che è ispirato al meccanismo utilizzato dalle reti neurali biologiche. L'input è costituito da un vettore di valori numerici a partire dai quali la rete impara in maniera costante, andando ad aggiustare i suoi parametri di bias e le connessioni (pesi) tra i suoi neuroni. Il processo di apprendimento è costituito da un periodo di **addestramento**, in cui la rete processa gli input e aggiusta i suoi parametri al fine di perfezionare le sue performance, e da un periodo di **applicazione**, in cui la rete applica i valori che ha imparato per eseguire il task assegnato.

Durante il processo di addestramento, la rete fornisce ripetutamente un output e il valore di discostamento tra esso e l'output desiderato. L'obiettivo è quello di minimizzare questa differenza e, per raggiungerlo, la rete ricalibra in modo continuativo i suoi pesi dopo la ricezione di ciascun vettore di input finchè non ottiene il valore desiderato. A questo punto, la fase di addestramento termina ed inizia una fase di **test** che viene eseguita utilizzando dei vettori, detti test set. Se la fase di test viene superata con successo, la rete è pronta per essere utilizzata.

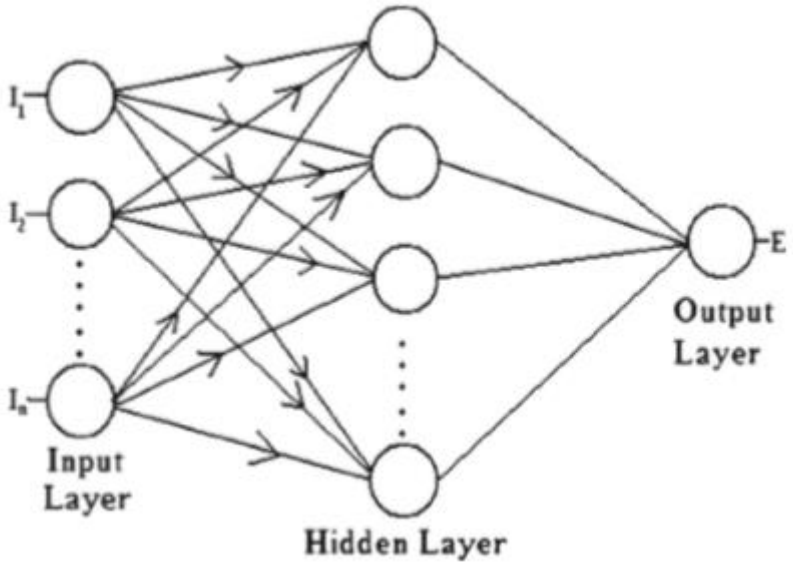

Figura 7. Esempio semplificato di una rete neurale.

Il **Feed-forward neural network** è la forma più semplice di rete neurale ed è rappresentato nella figura 4.1. È costituito da tre tipi di layers e le uniche connessioni esistenti sono tra un layer e quello successivo. L'**Input Layer** è costituito da un numero di neuroni pari al numero dei parametri di input, il **Hidden Layer** prende gli input dall'Input Layer, a cui è totalmente o parzialmente connesso, e li fornisce all'output Layer, con cui è totalmente connesso. L'**Output Layer** ha uno o più neuroni di output in base all'output del modello.

Ciascun neurone ha un valore di attivazione **a** (generalmente è un numero compreso tra 0 e 1) e, se è connesso ad un altro neurone, un peso **w** (generalmente un numero di tipo double). Per ottenere un neurone del layer successivo si utilizza la seguente formula:

$$\sigma(w_1 a_1 + \ldots + w_n a_n + b) = \text{new neuron}$$

dove b è il valore di bias, il cui scopo è quello di approssimare il momento in cui il valore del nuovo neurone inizia ad avere un significato, e σ è la funzione di attivazione (generalmente una funzione sigmoide) che serve a scalare l'output in modo tale che sia di nuovo compreso tra 0 e 1.

4.2 FEED-FORWARD NEURAL BACKPROPAGATION NETWORKS

Il **Feed-forward backpropagation neural network** è uno dei tipi di reti neurali più utilizzate nella stima dei costi di un progetto software. Essa serve a calcolare in modo efficiente i gradienti, i quali vengono poi utilizzati per effettuare il training della rete. Si parte dall'Output Layer e si segue una propagazione backwards, andando ad aggiornare i pesi e i bias di ciascun layer in modo da ottenere l'output desiderato. Per effettuare il calcolo del gradiente si utilizzano le seguenti tre equazioni, basate su delle derivate parziali, che servono ad aggiornare i valori di bias, dei pesi e delle attivazioni:

$$\frac{\partial C}{\partial w^{(L)}} = \frac{\partial C}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial z^{(L)}}{\partial w^{(L)}}$$

$$\frac{\partial C}{\partial b} = \frac{\partial C}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial z^{(L)}}{\partial b}$$

$$\frac{\partial C}{\partial a^{(L-1)}} = \frac{\partial C}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial z^{(L)}}{\partial a^{(L-1)}}$$

dove L = Last Layer, L–1 = Second Last Layer, l = layer, C = Funzione di Costo, a = attivazione, w = peso, b = bias, z = w*a+b.

Ciascuna derivata parziale legata ai pesi e ai bias viene salvata all'interno del **vettore dei gradienti** e raggruppata insieme ad altre in uno o più mini-batch. Successivamente, per ciascuna osservazione all'interno del mini-batch, viene eseguita la media degli output per ogni peso e ogni bias. Infine, l'output del vettore è dato dalla media tra gli output di ciascun mini-batch e i pesi e i bias vengono aggiornati di conseguenza secondo le seguenti formule:

$$w^{(l)} = w^{(l)} - \text{learning rate} * \frac{\partial C}{\partial w^{(L)}}$$

$$b^{(l)} = b - \text{learning rate} * \frac{\partial C}{\partial b^{(L)}}$$

4.3 RECURRENT NEURAL NETWORKS

Le **reti neurali ricorrenti** (RNN) sono caratterizzate da neuroni che, oltre ad essere connessi ai neuroni successivi nella rete, presentano anche delle connessioni con quelli precedenti e con se stessi. Questa ricorrenza agisce da **memoria** a breve termine e permette alla rete di imparare non solo attraverso il training ma anche attraverso i ricordi legati agli input precedenti. La RNN può prendere uno o più vettori di input e produce uno o più vettori di output, i quali sono influenzati sia dai

pesi legati agli input sia da vettori "nascosti" che rappresentano il contesto legato ai precedenti input e output e vengono aggiornati in modo ricorrente. Di conseguenza, lo stesso input può produrre output diversi a seconda degli input applicati nei cicli precedenti. Inoltre, l'utilizzo degli stati nascosti permette di identificare le relazioni tra i neuroni vicini anche in una situazione in cui si hanno una serie di input diversi.

Un altro vantaggio delle reti neurali ricorsive è che esse sono **bidirezionali**: infatti, ciascun neurone può avere connessioni con i neuroni successivi e con quelli precedenti. Questo perché il loro funzionamento non si basa soltanto sulla previsione del futuro a partire dal passato, ma anche sul perfezionamento delle informazioni legate al passato a partire dallo studio del futuro.

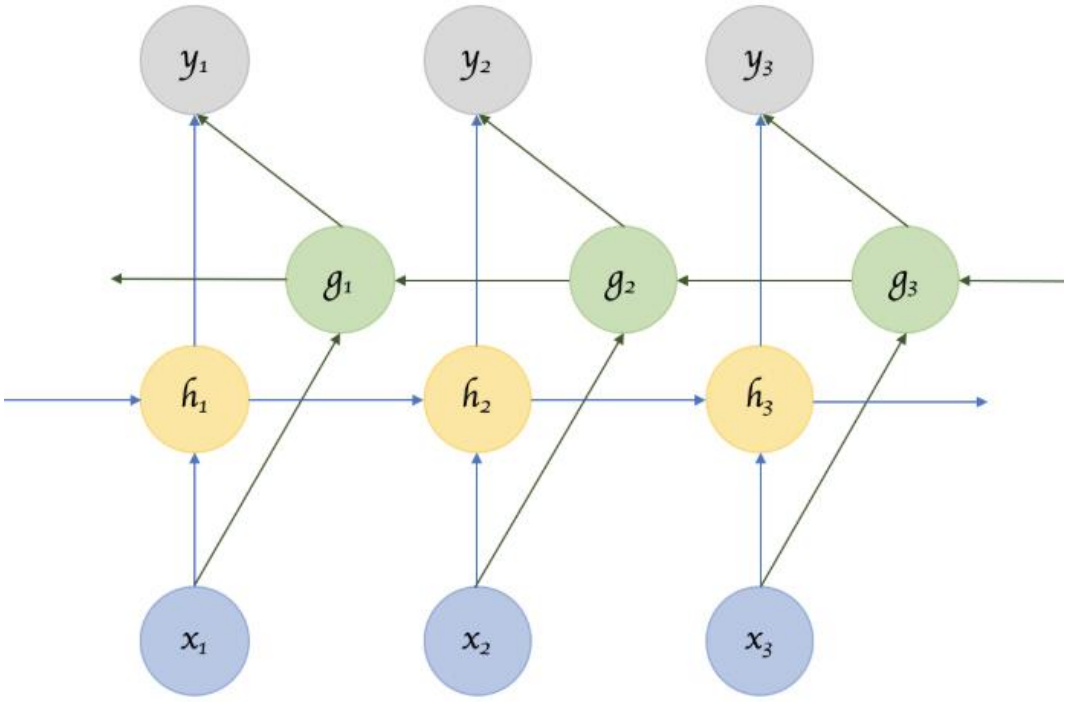

Figura 8. Una RNN bidirezionale.

4.4 FUZZY NEURAL NETWORKS

Un fuzzy set è un modello matematico basato sulla logica degli insiemi e costituito da dati qualitativi o quantitativi **vaghi**, generati tramite il linguaggio naturale. Il vantaggio dell'utilizzo di questo tipo di modello è dato dal fatto che esso permette di fare previsioni su fattori caratterizzati da una **classificazione incerta**, mentre l'approccio probabilistico permette di fare previsioni basate su dati certi e ben definiti. Un sottoinsieme fuzzy **A** è descritto da una funzione di membership $\boldsymbol{\mu_A}$ che mappa un più ampio insieme **U** all'interno dell'intervallo [0,1]. Può essere descritto con il formalismo seguente:

$\boldsymbol{\mu_A(x) = 0}$ se x non appartiene ad A

$\boldsymbol{\mu_A(x) = 1}$ se x appartiene ad A

$\boldsymbol{0 \leq \mu_A(x) \leq 1}$ se è possibile che x appartenga ad A

Più il valore di μ_A è vicino a 1 e più è alta la possibilità che x appartenga ad A.

Una **rete neurale fuzzy** è costituita da un algoritmo di machine learning che prende i suoi parametri da un insieme fuzzy e può essere utilizzata per risolvere un problema matematico per cui non esiste un preciso modello. Un sistema neuro-fuzzy è costituito da una particolare rete neurale feed-forward formata da tre diversi layer: il primo layer corrisponde alle variabili di input, il secondo rappresenta le regole dell'insieme fuzzy e il terzo corrisponde alle variabili di output. In qualsiasi momento del processo di learning, esso può essere rappresentato come un insieme di regole fuzzy, che devono essere inizializzate a monte del processo e possono essere considerate come dei prototipi dei dati di addestramento. Esistono due diversi tipi di reti neurali fuzzy (**Nauck et al**., 1997):

- **Cooperative fuzzy neural network**: la rete neurale e il sistema fuzzy lavorano in modo indipendente l'uno dall'altro. Le funzioni di membership dell'insieme fuzzy e i dati dell'addestramento vengono utilizzati dalla rete neurale per determinare le regole fuzzy e i pesi ad esse assegnate in base alla loro influenza.

- **Hybrid fuzzy neural network**: il sistema fuzzy è interpretato come una speciale rete neurale e l'architettura di questa tipologia di rete fa in modo che la rete neurale e il sistema fuzzy non debbano comunicare tra loro. I fuzzy set costituiscono i pesi mentre gli input, gli output e le regole sono modellate come neuroni. Le funzioni di membership vengono utilizzate come controller per formalizzare le regole fuzzy in termini di linguaggio.

Le reti neurali fuzzy vengono utilizzate per realizzare stime predittive dei costi di progetti software, dal momento che sono molto utili nel modellare e controllare sistemi non lineari. Tuttavia, questa metodologia presenta dei limiti: innanzitutto, essa non è universale in quanto si basa sulla self-organization delle funzioni di membership e spesso non è facile associare un significato fisico ai parametri chiave che, di conseguenza, risultano difficili da inizializzare. Inoltre, spesso la rete presenta una struttura complicata, i parametri da calibrare ad ogni step risultano essere troppi e ciò causa un notevole rallentamento del processo.

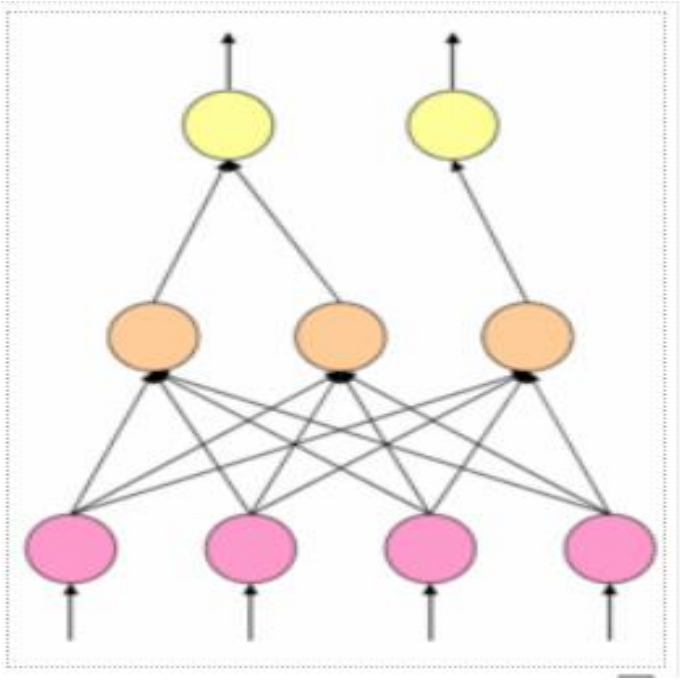

Figura 9. Una rete neurale fuzzy a 3 layer.

4.5 MODELLI BLACK-BOX VS. MODELLI WHITE-BOX

La principale criticità legata all'utilizzo di algoritmi di **machine learning** è legata all'abilità di dare una spiegazione agli stakeholders riguardo ai risultati della previsione. Per questo motivo, quando si deve scegliere il modello di machine learning da utilizzare, si deve ragionare sul trade-off tra la **precisione** e l'**interpretabilità** dei risultati. Esistono due tipologie di modello:

- **Modello black-box**: i meccanismi più profondi di funzionamento dell'algoritmo sono difficili da capire e non forniscono una visione dell'importanza delle caratteristiche legate alla previsione e di come esse interagiscano tra loro. Tuttavia, questo tipo di modello fornisce risultati molto precisi. In questa categoria rientrano, per esempio, le reti neurali.
- **Modello white-box**: i meccanismi alla base dell'algoritmo sono più facili da capire e da interpretare, ma le capacità di previsione e di modellare dataset particolarmente complessi è limitata. In questa categoria, rientrano la regressione lineare e gli alberi decisionali.

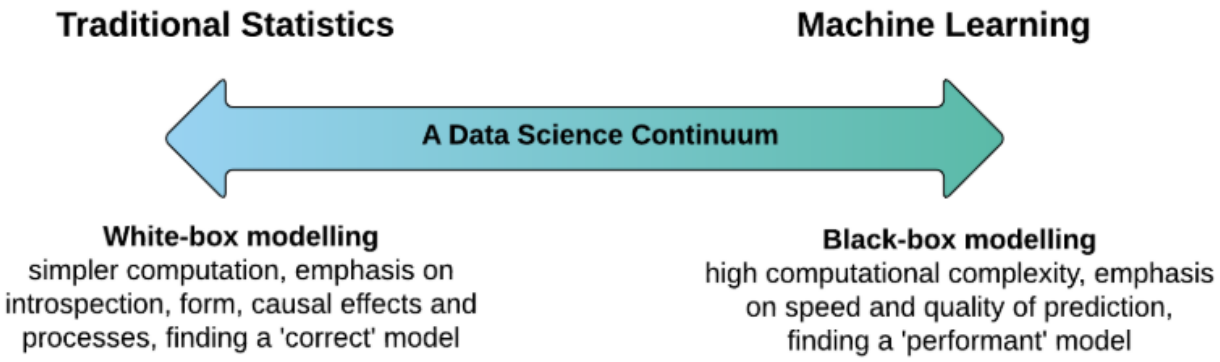

Figura 10. White-box VS. Black-box.

Il grado di spiegabilità di un modello è legato a due proprietà inerenti alla **funzione di risposta f(x)**, che definisce la relazione tra gli input (caratteristiche x) e gli output (target f(x)). In base al tipo di modello, questa funziona presenta le seguenti caratteristiche:

- **Linearità**: in una funzione di risposta lineare, l'associazione tra le caratteristiche e il target si comporta in modo lineare. Ciò significa che, se una caratteristica ha una variazione lineare, ci si aspetta che anche la funzione target vari in modo lineare in maniera proporzionale.
- **Monotonicità**: in una funzione di risposta monotona, la relazione tra una caratteristica e la funzione target è sempre in una precisa direzione (crescente o decrescente). Inoltre, la relazione si estende solo sul dominio della caratteristica presa in esame ed è indipendente dalle altre caratteristiche.

I modelli di regressione lineare presentano funzioni di risposta lineari e monotone, mentre le reti neurali presentano funzioni di risposta non lineari e non monotone. Come mostrato dai grafici sottostanti, i modelli white-box sono preferibili quando sono richiesti risultati semplici e di facile interpretazione, ma i vincoli legati alla linearità della funzione di risposta possono portare alla perdita di alcuni dati. Di conseguenza, se si vogliono ottenere delle osservazioni dei dati e dei risultati più accurati, è preferibile utilizzare modelli black-box, che però sono lineari e monotoni soltanto a livello locale. Per questo motivo, per andare a studiare il comportamento di un modello, si deve andare ad investigare a livello locale.

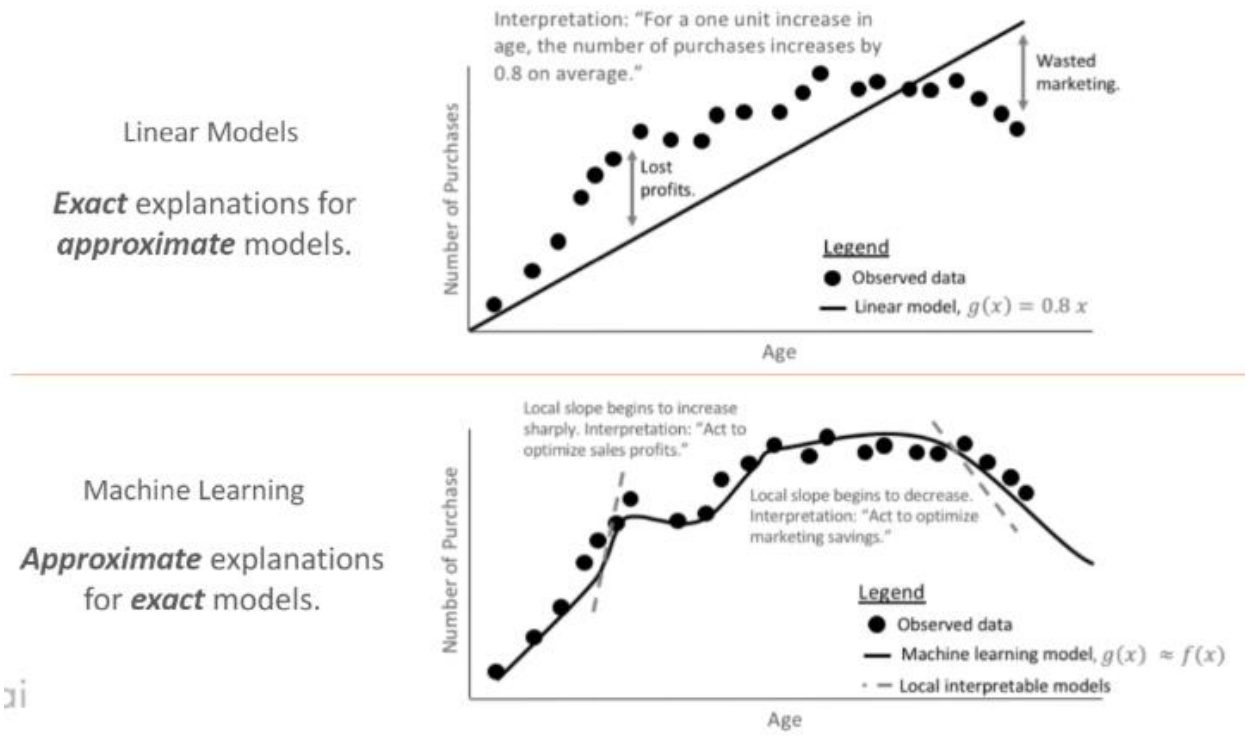

Grafico 5. Funzioni di risposta di un modello white-box e di un modello black-box a confronto.

Nonostante esistano alcune tecniche utili all'interpretazione dei modelli di machine learning come, ad esempio, l'utilizzo di **modelli surrogati** più semplici (alberi decisionali, modelli lineari) per fare un riassunto dei modelli più complessi, al fine di ottenere un'interpretazione delle previsioni il più accurata possibile è fondamentale andare ad identificare un framework ben definito e a chiarire le relazioni tra i principali descrittori del database preso in esame. Il lavoro di questa tesi si focalizzerà proprio su questo aspetto.

5. CASO DI STUDIO: DATABASE DI PROGETTI SOFTWARE DI UNA SOCIETÀ DI CONSULENZA IT

Il caso di studio è stato analizzato durante un percorso di tirocinio curriculare presso una **società di consulenza in ambito IT**, che si occupa di realizzare attività di supporto ai processi di business aziendali, di ricerca e sviluppo e dell'implementazione di nuove soluzioni ed ambienti di test nel campo dei **Financial Services** legati ad uno dei più importanti gruppi bancari italiani. Di seguito, verranno presentati l'iter di presa in carico delle attività progettuali e gli step che portano alla sua realizzazione.

Figura 11. Iter di realizzazione delle attività progettuali.

5.1 PIANIFICAZIONE E STIMA TEMPI/COSTI

Ciascuna attività progettuale viene presa in carico in seguito ad un contatto diretto con il cliente, in questo caso la direzione sistemi informativi della banca, che fornisce in input le specifiche tecniche del progetto mediante un'**analisi funzionale** (AFU) che deve contenere un'analisi dettagliata delle caratteristiche richieste e delle prestazioni desiderate, che devono essere espresse sia in termini descrittivi sia in termini quantitativi al fine di permettere una corretta identificazione degli **indicatori chiave** utili alla misurazione dei risultati ottenuti. A questo punto, i consulenti si occupano della redazione di una stima dei costi e dell'effort legati all'attività utilizzando la metodologia di **stima per analogia** con progetti precedenti. In seguito ad un processo di validazione da parte del project manager, la stima viene inviata al cliente e, in caso di una sua accettazione, si stabilisce una pianificazione concordata e si dà il go ufficiale all'attività.

Ciascuna attività progettuale è caratterizzata dalle seguenti fasi preliminari:

- **Analisi e pianificazione**: l'obiettivo di questa fase è quello di analizzare i requisiti del progetto ed identificare i **work packages** utili a suddividere l'attività e ad effettuarne una stima il più precisa possibile. I consulenti si occupano della redazione di tutti i **casi d'uso** del sistema, al fine di valutare la dimensione del codice prodotto e di definire requisiti precisi e non ambigui che mostrino tutti i possibili modi in cui un sistema può essere utilizzato e che permettano di valutarne la complessità. I casi d'uso sono descritti nell'AFU mediante dei diagrammi appositi definiti dagli standard UML, gli **Use Case Diagram**, che mostrano le interazioni che il sistema deve avere con sistemi esterni e con gli utenti. Per questo motivo, all'interno del caso d'uso vengono specificati gli attori coinvolti, la ragione per cui l'attore si appella al sistema, eventuali condizioni per poter accedere al caso d'uso, le possibili situazioni di errore e le conseguenti risposte del sistema la sequenza che attori e sistema devono seguire per completare positivamente l'utilizzo della funzionalità. L'insieme di tutti i casi d'uso rappresenta l'intera funzionalità da sviluppare e fornisce un'idea della

dimensione totale del progetto, che viene ricavata mediante la tecnica dei **function points** (FP). Questa metodologia permette la misurazione della dimensione dei requisiti funzionali in base alla loro tipologia, al numero di informazioni in input e in output ed alla quantità di memoria necessaria. Questo tipo di valutazione viene effettuata a prescindere dal linguaggio di programmazione utilizzato per svilupparla ed è quindi uniformemente utilizzabile per la maggior parte dei progetti software. Inoltre, il dimensionamento mediante function points prescinde dall'effort richiesto per lo sviluppo della funzionalità. Dal punto di vista del management, in questa fase vengono identificate le **milestone** di progetto e viene effettuata un'accurata pianificazione delle diverse fasi dell'attività progettuale. Inoltre, parallelamente alla pianificazione temporale, vengono identificate le risorse da assegnare al progetto in seguito ad un'accurata analisi della loro allocazione sulle varie attività.

- **Stima dei tempi e dei costi dell'attività**: una volta definiti i work packages e le relative durate, si ottiene una lista di attività a partire dalla quale si possono definire gli obiettivi temporali relativi alle diverse fasi ed attività di progetto, l'ordine in cui queste vengono svolte ed eventuali relazioni di dipendenza operativa tra i diversi pacchetti di lavoro da eseguire. A ciascun work package viene assegnato un numero di **story points** che rappresenta l'effort necessario alla sua realizzazione e le difficoltà ad essa legate. Come già accennato in precedenza, l'attribuzione degli story points viene fatta basandosi sull'ammontare dei function points derivanti dall'AFU e sull'esperienza legata a progetti simili realizzati in passato. In questo contesto, assume quindi grande importanza l'archiviazione delle attività passate in un database il più accurato possibile, che permetta al project manager e ai consulenti di accedere con facilità alle cosiddette **lessons learned**. L'utilizzo dei documenti relativi ad attività passate permette di risalire alla valutazione dell'**effort** reale che l'attività aveva richiesto e le motivazioni riguardanti eventuali scostamenti tra l'effort pianificato e quello effettivo. Inoltre, fornisce la possibilità di quantificare gli story points

da aggiungere in fase di stima come riserva di **contingency** da utilizzare in caso di forti criticità. Una volta ottenuto l'ammontare definitivo di story points necessari a svolgere l'attività progettuale e una volta che la stima è stata validata dal project manager, i dati relativi all'effort vengono convertiti in una stima dei tempi e dei costi dell'attività progettuale.
Ciò avviene mediante un **fattore di conversione** che, basandosi sulla produttività media registrata nei progetti passati, associa 1SP a 1,5 giorni di lavoro per risorsa. Questo fattore viene rivisto periodicamente in base alle performance e all'esperienza acquisita dalle risorse e presenta generalmente un andamento decrescente, tranne nei momenti in cui vengono inserite nuove risorse che necessitano di un periodo di inserimento e hanno una produttività iniziale minore. Il processo di stima termina con la definizione dei costi di progetto, che vengono ricavati moltiplicando una tariffa fissa giornaliera per il numero totale dei giorni.

Prospetto Costi Progetto

NOME PROGETTO

Funzioni	SP	GG/Uomo a Tariffa Media	Importo	AREA
Requisiti				
Recepimento requisiti e coordinamento	0,5	X	X	REQUISITI
Analisi				
Documentazione Tecnica	0	X	X	REQUISITI
Sviluppo software				
NDCE				
Adeguamento servizio generico				
Adeguamento flusso di orchestrazione del servizio	1	X	X	SVILUPPO
Nuovo connettore REST per recupero token	4	X	X	SVILUPPO
Implementazione logiche di gestione errori	2,5	X	X	SVILUPPO
Test Unitari				
Implementazione ed esecuzione test dei connettori	1,5	X	X	TEST
Adeguamento ed esecuzione test dei flussi	0	X	X	TEST
Adeguamento ed esecuzione test EJB	0	X	X	TEST
Supporto Independent Test(*)	0	X	X	TEST
Rilascio				
Gestione dei processi di change	0,5	X	X	CHANGE
Garanzia 1 anno	**1**	**0**	X	
Totale	**11**	**0**	X	
Note: Le tariffe esposte sono in gg / figura professionale	**IVA**	**22%**	X	
	Totale IVA inclusa		**X**	

Figura 12. Template di stima dei costi e dei tempi.

5.2 ESECUZIONE DEL PROGETTO

L'esecuzione di ciascuna attività progettuale è suddivisa nelle seguenti fasi:

- Sviluppo
- Sanity Test (Application Test)
- System Test
- UAT (User Acceptance Testing)
- Produzione

Parallelamente ad esse, gli sviluppatori del software si occupano della produzione della documentazione tecnica del progetto (**ATE**), che verrà fornita al cliente al termine dello stesso.

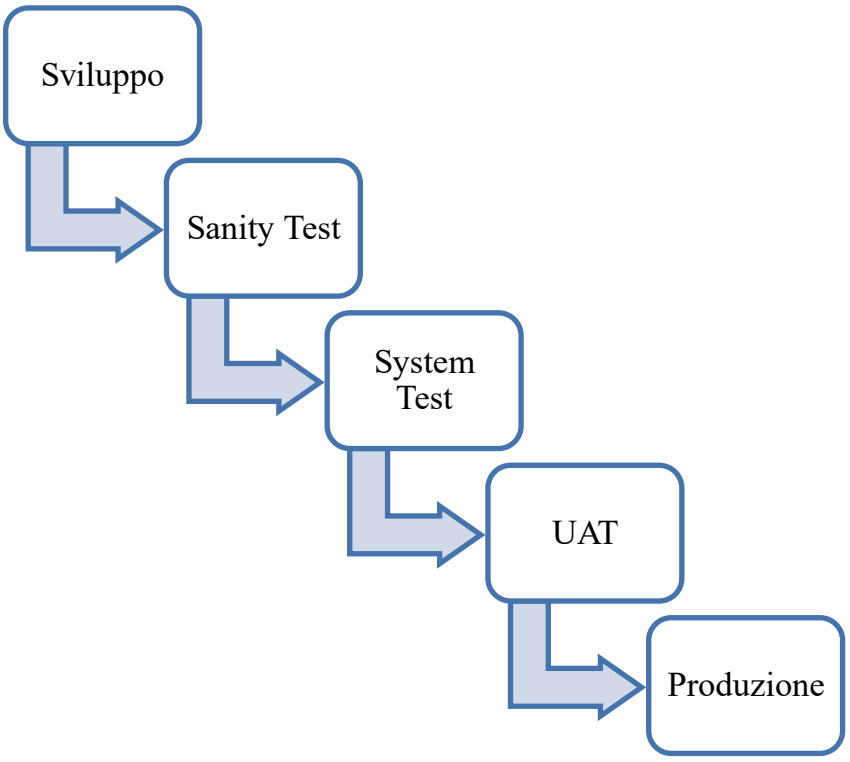

Figura 13. Milestones della fase di esecuzione del progetto.

5.2.1 SVILUPPO

Lo **sviluppo** del software consiste nella scrittura del codice necessario ad eseguire le funzionalità ed i servizi specificati nei requisiti di progetto. Esso viene eseguito all'interno di un **ambiente locale** che si appoggia a dei server fisici e a un cloud in cui sono installati gli applicativi necessari alla stesura del codice. La maggiore criticità legata a questa fase è data dal fatto che il codice prodotto deve tenere conto anche delle funzionalità e dei sistemi esterni con cui deve interagire e, di conseguenza, anche una modifica semplice effettuata in locale potrebbe rivelarsi più complicata e richiedere un effort maggiore per permetterne l'integrazione con gli altri sistemi preesistenti.

Prima che inizi la fase di sviluppo, il cliente crea un nuovo ramo progettuale, detto **branch**, e lo comunica formalmente al fornitore, il quale a sua volta crea un branch equivalente all'interno dell'ambiente locale di sviluppo. Per accedere a questo ambiente, gli sviluppatori utilizzano un'**utenza personale certificata** dotata di una firma digitale che permette alla banca di vedere chi ha scritto il codice.

La programmazione del software avviene mediante lo sviluppo di una serie di **moduli** che andranno a comporre il codice finale e che sono testabili dallo sviluppatore mediante **Unit Test**, un controllo che permette di verificare la correttezza delle istruzioni utilizzate.

5.2.2 SANITY TEST

Il **Sanity Test**, o Application Test, rappresenta il primo step di validazione interna del software e viene effettuato all'interno di un ambiente denominato **SVIL**. L'obiettivo di questa fase è quello di evidenziare eventuali grandi errori di programmazione o la presenza di gravi incompatibilità strutturali con gli standard di sviluppo aziendali. Il valore aggiunto fornito da questa tipologia di test consiste

nella **velocità** di esecuzione, in quanto viene effettuato unicamente sulle funzioni fondamentali. In caso di esito positivo dei test, si potrà quindi procedere ad effettuare verifiche più approfondite sulle funzionalità più specifiche.

L'esecuzione della fase di Sanity Test è eseguita mediante applicativi dedicati che eseguono la compilazione del codice ed il test delle principali funzioni sviluppate nel programma. In caso di esito negativo, lo sviluppatore esegue il **debug** manuale del codice, andando ad analizzare le singole istruzioni del codice e a correggere i **bug** emersi.

Terminata questa fase, il codice generato viene incapsulato all'interno di una **UDC** (Unità di Cambiamento) codificata creata appositamente dal cliente, all'interno della quale vengono racchiuse tutte le evolutive software riguardanti un unico progetto. Ottenuta la validazione mediante Sanity Test ed inoltrata la richiesta di creazione UDC, si procede alla fase successiva del ciclo di vita del progetto.

5.2.3 SYSTEM TEST

Terminato lo sviluppo locale e il Sanity Test, ha inizio la fase di **System Test,** durante la quale viene eseguita una verifica puntuale della soddisfazione dei requisiti definiti nelle specifiche.

Il trasferimento del codice validato dall'ambiente SVIL a quello di **TEST** avviene attraverso l'UDC assegnata e mediante lo scambio dei **changeset**, ovvero delle tabelle riportanti i nomi dei file di codice da rilasciare in System test, i relativi owner per ambo le parti ed altre informazioni legate al rilascio in produzione e ad eventuali relazioni con altri codici da tenere presenti durante la fase di test. Una volta ricevute queste informazioni, gli addetti all'esecuzione dei test in System prelevano le parti di codice da testare e le rilasciano in ambiente di TEST.

A questo punto, ha inizio una fase di **riciclo** in cui vengono eseguiti degli **Integration Test** utili a identificare eventuali anomalie legate all'aggiunta del

nuovo codice ad un'architettura già consolidata. Se non vengono riscontrate criticità, si procede al test massivo di tutte le componenti del codice, in cui viene verificata l'effettiva integrazione tra il sistema oggetto dei test e tutte le periferiche esterne e che tutte le funzionalità del software specificate nei requisiti siano soddisfatte. Ciò avviene fornendo un input al sistema e verificando che esso fornisca un output coerente con la funzionalità richiamata (**black-box testing**). In questa fase vengono anche testati gli aspetti legati all'utilizzatore finale, quali l'usabilità, le interfaccia grafiche ed eventuali tempi di recovery.

A monte della fase di System Test viene stabilito un **System Test Plan** che includa scopo ed obiettivi, il contesto, le aree critiche, il risultato atteso ed eventuali risultati intermedi, i metodi di esecuzione del test, la pianificazione dei test da un punto di vista temporale e sequenziale, i criteri di accettazione/rifiuto, i criteri di sospensione e ripresa dei test, l'ambiente in cui verranno effettuati i test ed, infine, i ruoli e le relative responsabilità.

Il cuore della fase di System Test è, invece, costituita dall'esecuzione automatizzata o manuale di un certo numero di **Test Cases** legati ad aspetti funzionali relativi alla **User Interface**. Oltre ad alcuni casi di base che vengono eseguiti per tutti i tipi di codice, vengono aggiunti anche casi di test specifici sulle singole funzionalità del codice oggetto in cui si verifica se sono ottenuti i risultati attesi.

Una volta terminata l'esecuzione dei casi di test, viene emessa la documentazione necessaria alla segnalazione dei problemi riscontrati e, se necessario, vengono eseguiti ulteriori test di regressione sulla funzionalità coinvolte. Questa procedura viene ripetuta iterativamente finchè non si ottengono i risultati desiderati. Tuttavia, è importante sottolineare come la fase di System Test sia utile alla verifica della correttezza formale del prodotto e del rispetto dei requisiti, ma non tiene conto dell'esperienza provata dall'utilizzatore finale, che viene testata in una fase successiva di validazione da parte dell'utente.

5.2.4 UAT

La fase di **UAT** (User Acceptance Test*)* consiste in una serie di prove di utilizzo delle nuove evolutive in via di rilascio da parte di un campione selezionato di utenti, detti **Beta Testers**. Questi utenti si occupano di provare tutte le funzionalità sviluppate dal lato dell'interfaccia grafica e sono totalmente indifferenti alla logica e al linguaggio di programmazione utilizzati in fase di sviluppo. Di conseguenza, essi si concentrano unicamente sul risultato finale e sull'interfaccia grafico del sistema.

Questa fase viene eseguita in caso di esito positivo al termine di tutti gli Unit Test, gli Integration Test ed i System Test e precede il rilascio in produzione del software.

L'importanza degli UAT deriva dal fatto che gli sviluppatori, per quanto un requisito possa essere ampiamente dettagliato, devono realizzare una funzionalità inizialmente inesistente e, quindi, soggetta ad interpretazione. Inoltre, alcuni aspetti dei requisiti espressi nel contratto sono realmente comprensibili solo dagli utenti finali e ciò fa sì che gli sviluppatori possano avere dei problemi nella loro implementazione.

In aggiunta a questi aspetti, è importante sottolineare che il costo di apportare una modifica al software in ambiente UAT è decisamente inferiore rispetto a quello di una modifica effettuata in ambiente di Produzione, dove c'è il rischio di coinvolgimento di altre entità dell'architettura oltre ad un danno d'immagine per il fornitore.

Poiché lo UAT è una fase molto delicata del processo di validazione del software, è importante stabilire in modo preciso il tempo necessario allo svolgimento dei test in modo da rispettare la pianificazione temporale del progetto e da avere dei buffer temporali utili alla risoluzione di eventuali defect. Inoltre, è necessario formare in modo adeguato gli utenti che dovranno svolgere i test ed emettere una documentazione precisa sulle finalità e sui particolari relativi ai vari Test Cases.

5.2.5 PRODUZIONE

Prima di entrare in fase di **Produzione**, il software viene rilasciato in un ambiente di **Pre-Production** gestito dal cliente, all'interno del quale vengono inserite tutte le UDC che non devono subire più modifiche e che hanno superato con esito positivo i test. Questo ambiente presenta delle caratteristiche strutturali e funzionali uguali a quelle dell'ambiente di produzione e permette di rilasciare tutte le UDC in produzione nello stesso momento.

In questo modo, al momento del passaggio in produzione, non è più necessario rilasciare le singole UDC, ma è sufficiente sincronizzare l'ambiente di Production con quello di Pre-Production, andando a modificare i riferimenti ai database.

Con il passaggio del software in Produzione si abilitano tutte le nuove funzionalità e, contemporaneamente, si crea una copia dell'ambiente che diventerà l'ambiente di Pre-Production per il rilascio successivo.

Questo insieme di operazioni viene eseguito tramite una multipiattaforma che facilita l'integrazione tra mainframe con dati ed applicativi differenti e si basa sull'utilizzo di un linguaggio comune.

Al termine di questa fase, il software viene messo a disposizione degli utenti ed inizia una fase di un anno detta **follow up**, in cui il fornitore si impegna ad effettuare interventi di manutenzione in caso di malfunzionamenti del codice prodotto.

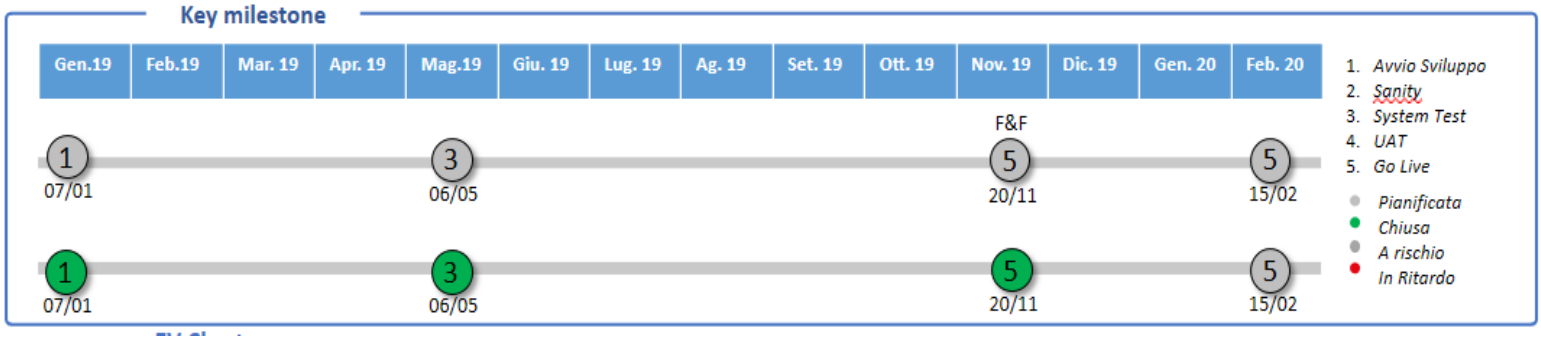

Figura 14. Esempio di Gantt di progetto.

5.3 ANALISI DEL DATABASE E DEI DESCRITTORI DI PROGETTO

Come accennato in precedenza, la metodologia di stima utilizzata dall'azienda sede del tirocinio è quella per **analogia**. Sebbene questo metodo sia di semplice applicazione ed eviti di coinvolgere nel processo di stima il parere di esperti o l'utilizzo di modelli parametrici e algoritmici complicati, esso presenta alcuni limiti da tenere in considerazione.

Innanzitutto, prima che il sistema sia utilizzabile, è necessario un lungo processo di raccolta di dati storici che può essere soggetto ad imprecisioni che possono causare degli errori durante le analisi che precedono la stima. Inoltre, può succedere che due progetti che potrebbero sembrare simili sulla carta si rivelino in realtà notevolmente diversi e che si possano presentare dei nuovi casi che non presentano analogie con progetti passati. Di conseguenza, l'**incertezza** e la **soggettività** legate alle similarità e alle differenze tra progetti possono portare a situazioni in cui due analisti diversi producano una stima con valori completamente discordanti.

Tuttavia, il passaggio ad una metodologia di stima basata su algoritmi di machine learning necessita di uno studio a monte, che permetta l'identificazione dei principali descrittori di progetto e la costruzione di un **framework** ben definito utile a saper interpretare correttamente i risultati che l'algoritmo fornisce in output.

Il primo passo di questo processo è costituito da un accurata raccolta dei dati, i quali devono essere analizzati e filtrati in modo da ottenere un database il più efficiente possibile. Una volta terminata l'analisi delle relazioni tra i principali descrittori di progetto, sarà più facile interpretare i risultati di una stima dei tempi e dei costi effettuata tramite un algoritmo di tipo black-box e capire se esso fornisce un output che sia più affidabile rispetto ad una metodologia classica come, ad esempio, la regressione lineare. Per valutare questa affidabilità, è necessario effettuare una stima utilizzando entrambe le metodologie e, in un secondo momento, andare ad effettuare una **sensitivity analysis** che fornisca alcune indicazioni sull'errore standard associato a ciascuna di esse. Solo al termine di questa operazione, dopo

aver confrontato i risultati ottenuti, sarà possibile scegliere la metodologia di stima più adatta. Dal momento che l'obiettivo di questo lavoro di tesi non è quello di proporre una metodologia particolare di machine learning da utilizzare, di seguito verranno presentati due esempi di utilizzo del modello di regressione lineare per la stima della durata e dell'effort di progetto. Al termine di questa parte esemplificativa, verrà infine illustrata la metodologia di sensitivity analyisis da seguire.

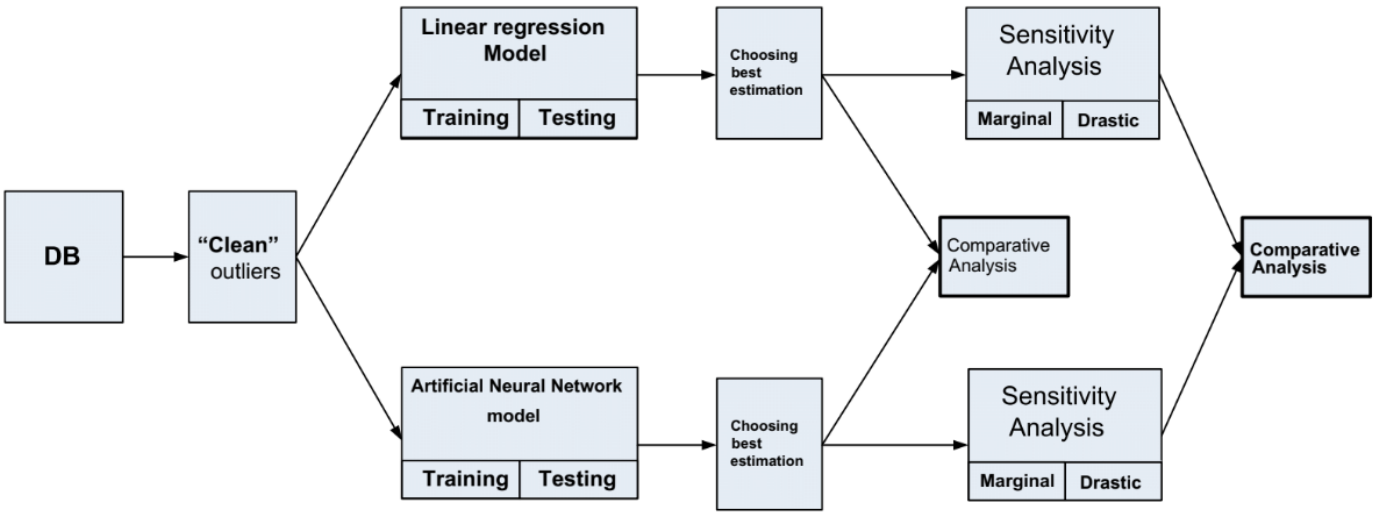

Figura 15. Modello della procedura da seguire per delineare il framework di stima.

5.3.1 DATA PREPARATION

Il processo di preparazione dei dati inizia con l'identificazione dei **descrittori** di progetto, ovvero quegli **indicatori chiave** che permettono di valutare le performance e di effettuare delle previsioni sull'andamento futuro dei tempi e dei costi legati al progetto.

Dall'analisi di un database costituito da 300 attività progettuali legate al campo dell'IT Finance Services, sono stati individuati i seguenti descrittori:

- **La produttività delle risorse assegnate al progetto**: è una misura dell'efficienza produttiva calcolata come il rapporto tra ciò che viene prodotto e ciò che si doveva produrre. Misurare la produttività di un ingegnere o di un processo ingegneristico non è un processo intuitivo poiché, quando si ha in input l'**esperienza**, la **conoscenza** e il **tempo investito** da uno sviluppatore e si ha in output un computer o un software, la relazione tra input e output non è così immediata da identificare. Inoltre, le difficoltà aumentano quando si riusano parti di codice legate a progetti passati, pratica molto diffusa nel mondo dello sviluppo software. Infatti, l'affidabilità della stima dipende da quanto il progetto in corso sia simile a quelli passati e sull'abilità del manager di individuare tutte le loro sfaccettature. Nel caso di studio preso in esame, per calcolare la produttività di ciascun gruppo di lavoro, è stato assegnato a ciascuna risorsa un fattore di performance deciso dal project manager in modo soggettivo sulla base dell'esperienza legata ai progetti passati. In seconda battuta, a ciascuna risorsa è stato assegnato un FTE di costo calcolato come il rapporto tra il suo costo giornaliero e la tariffa media giornaliera utilizzata in fase di stima del progetto. Infine, il valore di FTE di performance è stato normalizzato utilizzando il valore del FTE di costo, in modo da assegnare a ciascuna risorsa un fattore di performance che sia correlato al suo costo. Di seguito, un calcolo di esempio.

  Situazione ideale: 1 Risorsa = 1 FTE

Risorsa X:

Fattore di performance soggettivo (assegnato dal PM)

FTE costo = costo giornaliero risorsa/tariffa media giornaliera del progetto

Productivity = Fattore di performance sogg/FTE costo

Se la produttività della risorsa è maggiore del suo FTE di costo, significa che essa ha delle performance maggiori rispetto a quello che è il suo costo. Viceversa, se la produttività è minore del suo FTE di costo, significa che la risorsa rende meno di quanto dovrebbe e il suo costo è troppo alto rispetto ai risultati raggiunti.

La produttività totale di ciascun gruppo è data dalla somma della produttività delle singole risorse che lo compongono.

- **Le dimensioni del codice prodotto**: Basandosi sui requisiti, è possibile determinare la dimensione del prodotto in SLOC (**Source Lines of Code**) che, a sua volta, determina l'effort del progetto e la sua durata. In questo calcolo sono incluse tutte le istruzioni create dal personale di progetto che vengono trasformate in codice macchina da processori, compilatori e assemblatori. Sono esclusi i commenti e le parti non modificate del software.

- **La complessità del prodotto (dimensione funzionale)**: questo indicatore è di fondamentale importanza, in quanto va ad influenzare i tempi, i costi e la qualità del progetto. Generalmente, un'alta complessità funzionale è associata ad una crescente durata e ad un superiore costo. Essa si può, quindi, definire come l'insieme delle varie **parti correlate** legate alla **differenziazione** e all'**interdipendenza** del progetto (**Baccarini**). Secondo uno studio eseguito da Meyer e Utteback nel 1995, la complessità del prodotto non è associata alla performance tecnologica, al costo, ai tempi e ai risultati generali del progetto. Al contrario, l'aumento del numero di tecnologie coinvolte nella sua realizzazione è positivamente associato ai tempi legati al **ciclo di sviluppo**. Al fine di quantificare la complessità del

prodotto, si può utilizzare la tecnica dei **function points** o si può semplicemente andare ad identificare tutti i tipi di **file** coinvolti nella sua realizzazione (input e output files, read-from files, written-to files, file interni utili alla creazione del prodotto). Nel caso di studio in esame, si è deciso di assegnare un valore di complessità compreso tra 1 e 10 in base alla quantità di function points relativi all'attività.

- **La durata del progetto:** è calcolata come l'**elapsed time** totale in giorni e comprende tutte le attività, pianificate e non pianificate, che contribuiscono alla realizzazione del progetto.

- **L'effort di progetto:** la creazione di un modello preciso e affidabile per stimare l'effort legato a progetti altamente tecnologici è un obiettivo complicato da raggiungere. Il **PMBOK** definisce l'effort di progetto come "il numero di unità lavorative richieste per completare un'attività o un altro elemento del progetto. Generalmente, è espresso in ore/uomo, giorni/uomo o settimane/uomo e non va confuso con la durata del progetto." I dati legati all'effort di progetto vengono ricavati dalla documentazione e dal database. In base a questi dati, è anche possibile ottenere un'indicazione di quanto l'effort effettivo sia stato proporzionale a quanto era stato messo in preventivo in fase di stima.

- **Numero di defects legati al progetto:** la qualità di un prodotto software è più complessa da valutare rispetto a quella di un prodotto più concreto. Infatti, generalmente non sono previsti questionari o altri metodi per ottenere un feedback dai clienti. Perciò, l'unico indicatore utile a valutare la bontà dell'output di progetto è il numero di segnalazioni (**defects**) che il fornitore riceve da parte dei clienti o degli utenti finali.

Una volta identificati i principali descrittori di progetto, il primo passo per processare il database è quello di analizzare a campione i dati al fine di trovare

eventuali **outliers**, ovvero valori che sono anomali rispetto alle altre osservazioni effettuate. Infatti, la presenza di outliers all'interno di un database tende a far crescere il valore dell'**errore standard** legato alla stima. Ci sono diversi metodi per trovare gli outliers, come l'utilizzo di tool grafiche o di formule matematiche. In questo caso, è stato usato il metodo di **Hidiroglou e Berthelot,** che si basa sul ricorso a soglie di accettazione derivate dai **quartili** della distribuzione empirica. Identificati con $q_{0,25}$, $q_{0,50}$, $q_{0,75}$ i tre quartili della distribuzione empirica della variabile di interesse relativa ad un campione di n osservazioni, si possono definire gli scarti interquartili inferiore e superiore, dati dalle seguenti formule:

$$\boldsymbol{d_{inf} = q_{0,50} - q_{0,25}}$$

$$\boldsymbol{d_{sup} = q_{0,75} - q_{0,50}}$$

L'**intervallo di accettazione** di una generica osservazione sarà dato da:

$$\boldsymbol{A = (q_{0,50} - d_{inf} * c_{inf}; q_{0,50} + d_{sup} * c_{sup})}$$

Dove c_{inf} e c_{sup} sono parametri arbitrari che, se posti entrambi uguali a 1, fanno sì che l'intervallo di accettazione si riduca allo scarto interquartile.

Nel caso di studio preso in esame, sono state considerate le 25 attività progettuali partite dalla data di inizio del tirocinio (ottobre 2019), che possono essere considerate esemplificative dei dati presenti nell'intero database.

Projects	Assigned Resources	Productivity	SLOC	Functional Complexity	Duration	Effort	Number of Defects
Project 1	Gruppo 7	1,995299145	748	7	80	95	9
Project 2	Gruppo 6	3,036324786	293	3	40	15	8
Project 3	Gruppo 6	3,036324786	584	4	25	10	6
Project 4	Gruppo 5	2,12	365	6	45	47	4
Project 5	Gruppo 5	2,12	398	3	56	33	7
Project 6	Gruppo 1	3,123076923	555	4	34	15	20
Project 7	Gruppo 6	3,036324786	1003	5	79	57	9
Project 8	Gruppo 8	2,082051282	441	7	105	98	14
Project 9	Gruppo 1	3,123076923	289	5	23	18	30
Project 10	Gruppo 1	3,123076923	937	4	55	35	12
Project 11	Gruppo 6	3,036324786	162	4	35	21	9
Project 12	Gruppo 5	2,12	349	2	18	8	22
Project 13	Gruppo 4	1,77	1051	8	78	86	0
Project 14	Gruppo 6	3,036324786	1059	9	98	135	17
Project 15	Gruppo 9	3,21	963	7	78	40	7
Project 16	Gruppo 4	1,77	491	5	80	92	22
Project 17	Gruppo 3	3,643589744	311	4	35	13	23
Project 18	Gruppo 2	2,07	755	6	66	70	7
Project 19	Gruppo 3	3,643589744	432	3	13	3	22
Project 20	Gruppo 2	2,07	354	3	29	28	30
Project 21	Gruppo 6	3,036324786	407	4	35	46	35
Project 22	Gruppo 8	2,082051282	1417	10	110	140	15
Project 23	Gruppo 5	2,12	442	4	42	33	44
Project 24	Gruppo 4	1,77	458	5	75	35	21
Project 25	Gruppo 1	3,123076923	378	1	18	6	38

Tabella 1. Database delle attività progettuali.

Di seguito, un calcolo di esempio legato all'identificazione degli outlier:

Descrittore preso in esame: **Duration**

N = 25; $\bar{X} = 54{,}08\ gg$; $c_{inf} = c_{sup} = 1$; $q_{0,25} = 13{,}5$; $q_{0,50} = 27$; $q_{0,75} = 40{,}5$; $d_{inf} = 13{,}5$; $d_{sup} = 13{,}5$ -> **A = (13,5 ; 40,5)**

In base al risultato ottenuto, possono essere considerati outlier tutti i progetti che hanno avuto una duration al di fuori dell'intervallo A. Una volta terminata questa operazione, al fine di rafforzare il processo di stima di un nuovo progetto, il secondo step è andare a studiare le **relazioni** tra i descrittori in modo da essere in grado di interpretare nel modo più accurato possibile l'output della stima. Questa parte del processo è di fondamentale importanza quando si vuole utilizzare una metodologia basata su un algoritmo che non è ben conosciuto dal project manager o quando si vuole utilizzare un algoritmo di machine learning black-box, di cui non sono note e osservabili le funzioni interne.

Inoltre, prima di procedere con il passo successivo, sono state analizzate le distribuzioni statistiche di ciascun descrittore preso in esame. Per farlo, è stata utilizzata la funzionalità **Input Analyzer** del tool **Arena Simulation** che, mediante la sua funzione di fit, permette di individuare la distribuzione statistica che fornisce il **Mean Square Error** minore e che, di conseguenza, modella nel modo migliore il descrittore in esame.

- Dall'analisi effettuata, per la **produttività** sono stati ottenuti i seguenti risultati:

```
       Distribution Summary

Distribution:   Beta
Expression:     1.58 + 2.25 * BETA(1.32, 1.55)
Square Error:   0.122124

Chi Square Test
  Number of intervals   = 3
  Degrees of freedom    = 0
  Test Statistic        = 16.6
  Corresponding p-value < 0.005

Kolmogorov-Smirnov Test
  Test Statistic        = 0.248
  Corresponding p-value = 0.081

       Data Summary

Number of Data Points   = 25
Min Data Value          = 1.77
Max Data Value          = 3.64
Sample Mean             = 2.61
Sample Std Dev          = 0.62
```

Function	Sq Error
Beta	0.122
Triangular	0.125
Uniform	0.128
Weibull	0.146
Normal	0.147
Erlang	0.148
Gamma	0.148
Lognormal	0.155
Exponential	0.187

Dal momento che presenta un valore del MSE inferiore rispetto alle altre, si evince che la migliore distribuzione che descrive la produttività all'interno del database è quella di tipo **Beta**. Come si può notare, il valore del p-value relativo al test del Chi quadro è molto più basso della soglia di significatività, che è pari a 0.05. Ciò significa che il test effettuato è statisticamente significativo.

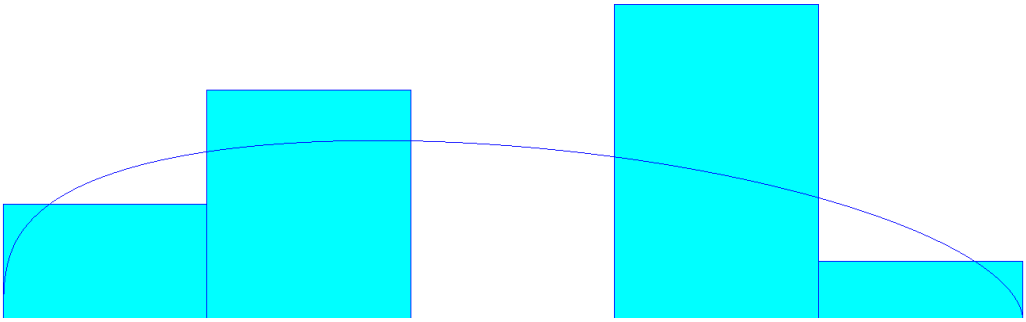

Grafico 6. Distribuzione statistica della produttività.

- Per quanto riguarda il descrittore **SLOC**, sono stati ottenuti i risultati seguenti:

```
       Distribution Summary

Distribution:   Beta
Expression:     162 + 1.26e+003 * BETA(0.844, 1.66)
Square Error:   0.018130

Chi Square Test
  Number of intervals   = 3
  Degrees of freedom    = 0
  Test Statistic        = 0.374
  Corresponding p-value < 0.005

Kolmogorov-Smirnov Test
  Test Statistic        = 0.187
  Corresponding p-value > 0.15

       Data Summary

Number of Data Points   = 25
Min Data Value          = 162
Max Data Value          = 1.42e+003
Sample Mean             = 586
Sample Std Dev          = 317
```

Function	Sq Error
Beta	0.0181
Weibull	0.0209
Exponential	0.022
Erlang	0.022
Triangular	0.0235
Gamma	0.0245
Lognormal	0.0604
Normal	0.0783
Uniform	0.0864

Anche in questo caso, la distribuzione che meglio descrive il SLOC è la distribuzione **Beta**, che mostra un MSE minore rispetto alle altre. Il ragionamento relativo al p-value è il medesimo fatto per la produttività.

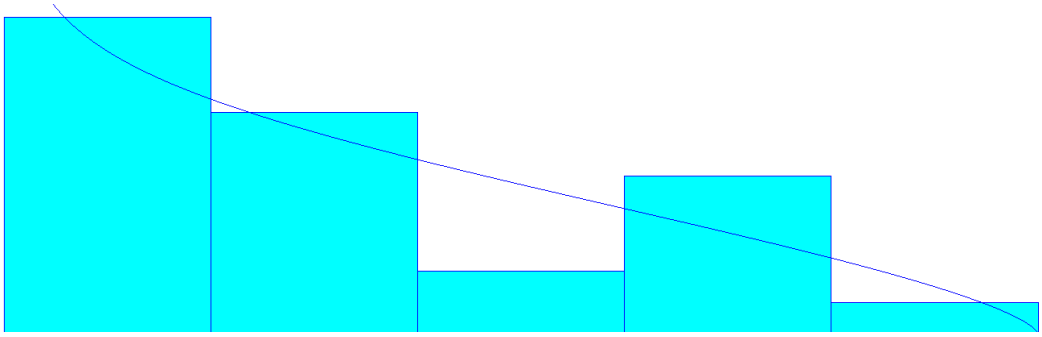

Grafico 7. Distribuzione statistica della SLOC.

- Per quanto riguarda la **complessità funzionale**, i risultati ottenuti sono i seguenti:

```
        Distribution Summary

Distribution:   Erlang
Expression:     0.5 + ERLA(1.11, 4)
Square Error:   0.015082

Chi Square Test
  Number of intervals   = 3
  Degrees of freedom    = 0
  Test Statistic        = 0.538
  Corresponding p-value < 0.005

        Data Summary

Number of Data Points   = 25
Min Data Value          = 1
Max Data Value          = 10
Sample Mean             = 4.92
Sample Std Dev          = 2.16
```

Function	Sq Error
Erlang	0.0151
Triangular	0.0159
Gamma	0.0166
Poisson	0.0187
Weibull	0.0189
Lognormal	0.0222
Normal	0.0239
Beta	0.0262
Uniform	0.0584
Exponential	0.0847

In questo caso, la distribuzione migliore e la distribuzione di **Erlang**, in quanto presenta il MSE minore. Il ragionamento relativo al p-value è il medesimo fatto per la produttività e per il SLOC.

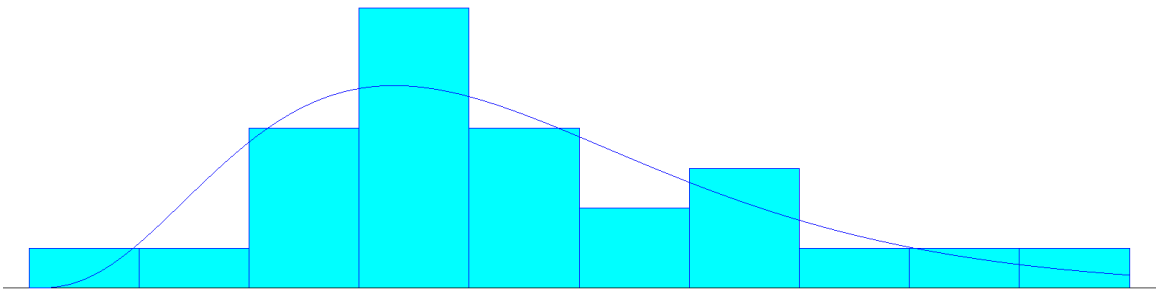

Grafico 8. Distribuzione statistica della complessità funzionale.

- Andando ad analizzare i dati relativi alla **durata** dell'attività progettuale, i risultati raggiunti sono i seguenti:

```
       Distribution Summary

Distribution:   Beta
Expression:     12.5 + 98 * BETA(0.794, 0.998)
Square Error:   0.048227

Chi Square Test
  Number of intervals   = 4
  Degrees of freedom    = 1
  Test Statistic        = 2.3
  Corresponding p-value = 0.144

       Data Summary

Number of Data Points   = 25
Min Data Value          = 13
Max Data Value          = 110
Sample Mean             = 54.1
Sample Std Dev          = 28.9
```

Function	Sq Error
Beta	0.0482
Uniform	0.049
Weibull	0.049
Gamma	0.0491
Exponential	0.0494
Erlang	0.0494
Normal	0.0502
Triangular	0.0503
Lognormal	0.0506
Poisson	0.0835

In questo caso, la distribuzione migliore è la distribuzione di tipo **Beta**, in quanto presenta un MSE minore delle altre. Il p-value ottenuto nel test del Chi Quadro è superiore al valore di significatività (0.05) e, di conseguenza, si può affermare che il test non è statisticamente significativo poiché il risultato potrebbe essere determinato da un effetto casuale del campionamento.

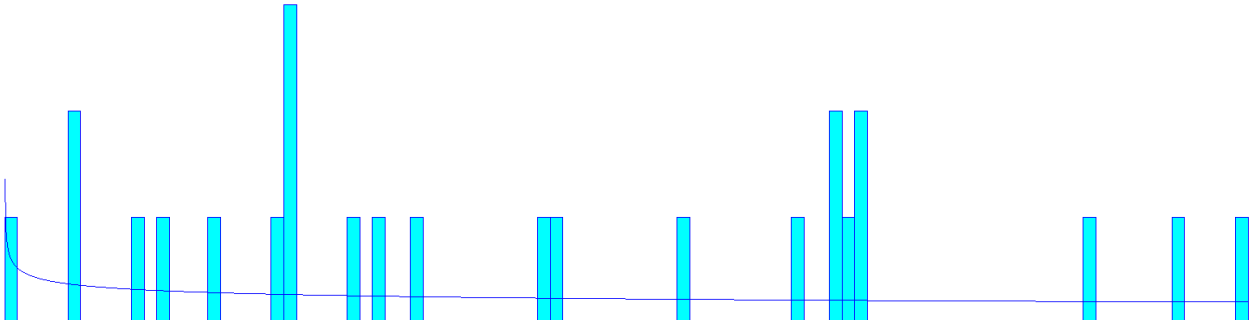

Grafico 9. Distribuzione statistica della durata.

- Per quanto riguarda l'**effort**, i risultati dell'analisi sono i seguenti:

```
       Distribution Summary

Distribution:   Exponential
Expression:     3 + EXPO(44.2)
Square Error:   0.027233

Chi Square Test
  Number of intervals   = 3
  Degrees of freedom    = 1
  Test Statistic        = 0.858
  Corresponding p-value = 0.384

Kolmogorov-Smirnov Test
  Test Statistic        = 0.0931
  Corresponding p-value > 0.15
```

Function	Sq Error
Triangular	0.0438
Erlang	0.0449
Gamma	0.045
Weibull	0.045
Lognormal	0.0465
Beta	0.0466
Normal	0.0475
Uniform	0.0498
Exponential	0.0507
Poisson	0.0867

La distribuzione migliore è quella di tipo **esponenziale**, poiché presenta un MSE minore rispetto alle altre distribuzioni prese in esame. Anche in questo caso, il p-value è superiore alla soglia di significatività scelta (0.05) e, di conseguenza, valgono le stesse conclusioni fatte precedentemente per la durata.

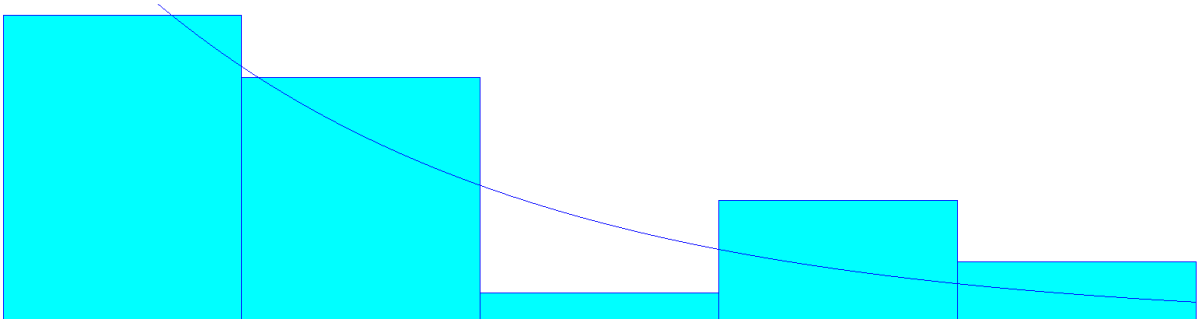

Grafico 10. Distribuzione statistica dell'effort.

- Infine, per quanto riguarda il **numero di defect**, i risultati dell'analisi sono i seguenti:

```
       Distribution Summary

Distribution:   Triangular
Expression:     TRIA(-0.5, 7.72, 44.5)
Square Error:   0.043845

Chi Square Test
  Number of intervals   = 4
  Degrees of freedom    = 2
  Test Statistic        = 1.95
  Corresponding p-value = 0.398

       Data Summary

Number of Data Points   = 25
Min Data Value          = 0
Max Data Value          = 44
Sample Mean             = 17.2
Sample Std Dev          = 11.5
```

Function	Sq Error
Erlang	0.0272
Exponential	0.0272
Gamma	0.0341
Weibull	0.0345
Triangular	0.0476
Beta	0.0499
Lognormal	0.0696
Normal	0.079
Uniform	0.096

La distribuzione migliore è quella di tipo **triangolare**, poiché presenta un MSE minore rispetto alle altre distribuzioni prese in esame. Poiché il p-value è superiore alla soglia di significatività scelta (0.05), valgono le stesse conclusioni raggiunte per durata ed effort.

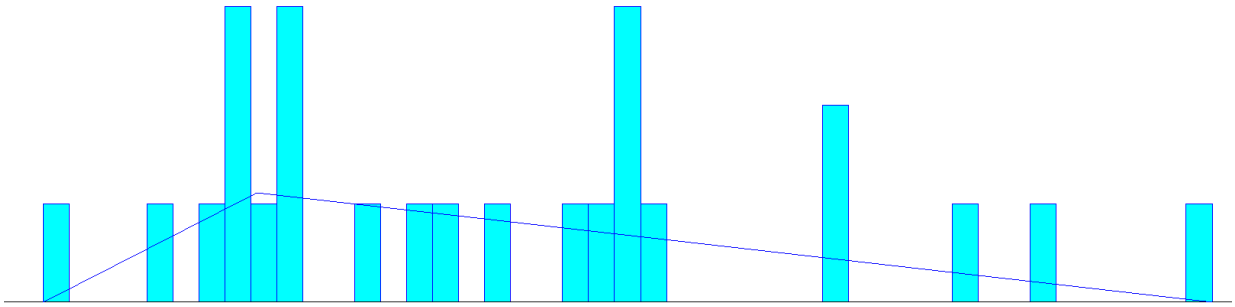

Grafico 11. Distribuzione statistica del numero di defects.

5.3.2 ANALISI DELLE RELAZIONI TRA I DESCRITTORI DI PROGETTO

Come già accennato in precedenza, lo step successivo del processo di rafforzamento della procedura di stima di progetto consiste nell'identificazione e nell'analisi della correlazione tra i descrittori individuati.

Le regole da seguire per la corretta interpretazione dei risultati della matrice di correlazione sono le seguenti:

- Se il coefficiente di correlazione $\rho > 0$, le due variabili sono **positivamente correlate**. Ciò significa che, all'aumentare del valore di una variabile, anche il valore dell'altra tende ad aumentare.
- Se il coefficiente di correlazione $\rho = 0$, le due variabili sono **incorrelate**.
- Se il coefficiente di correlazione $\rho < 0$, le variabili sono **negativamente correlate**. Ciò significa che, se il valore di una variabile diminuisce, quello dell'altra tende ad aumentare.

Per quanto riguarda la correlazione diretta, si distinguono le seguenti tre casistiche:

- Se $0 < |\rho| < 0.3$, si ha **correlazione debole**.
- Se $0.3 < |\rho| < 0.7$, si ha **correlazione moderata**.
- Se $|\rho| > 0.7$, si ha **correlazione forte**.

Per effettuare l'analisi di **correlazione** tra i descrittori, è stata utilizzata la funzione di **Data Analysis** di Microsoft Excel, che fornisce in output la **matrice di correlazione** dei dati presi in esame. Di seguito, l'immagine relativa al suddetto output.

	Productivity	*SLOC*	*Functional Complexity*	*Duration*	*Effort*	*Number of Defects*
Productivity	1					
SLOC	-0,118472745	1				
Functional Complexity	-0,306651568	0,739755549	1			
Duration	-0,453525311	0,70149397	0,841332117	1		
Effort	-0,470016485	0,692108719	0,873753373	0,896586797	1	
Number of Defects	0,130005462	-0,380376729	-0,42500122	-0,408761171	-0,26945	1

Tabella 2. Matrice di correlazione dei descrittori di progetto.

Prima di procedere con l'analisi dei risultati, è doveroso premettere che i valori dei coefficienti di correlazione sono influenzati dalla numerosità del campione preso in esame. Se si prendesse un database più ampio, potrebbe succedere che alcune variabili presentino una correlazione diversa da quella ottenuta nel presente caso di studio. Tuttavia, la procedura da seguire rimane la stessa.

Dall'output ottenuto, si evince che la **produttività** presenta una **correlazione debole negativa** con l'indicatore SLOC, una **correlazione negativa moderata** con la complessità funzionale, la durata e l'effort del progetto e una **correlazione debole positiva** con il numero di defects legati al progetto. Il risultato coincide con ciò che ci si potrebbe aspettare in quanto è logico pensare che, all'aumentare della produttività, la durata e l'effort del progetto tendano a diminuire. Ciò che potrebbe sorprendere è invece la relazione tra produttività e numero di defects, che mostra come non è detto che un gruppo di lavoro più produttivo porti allo sviluppo di un software con un numero minore di bug. Al contrario il risultato mostra come, all'aumentare della produttività, il numero di defects tenda a crescere, anche se in misura molto leggera.

Per quanto riguarda la dimensione del codice prodotto (**SLOC**), essa presenta una **correlazione forte positiva** con la complessità funzionale e con la durata del progetto, una **correlazione moderata positiva** con l'effort di progetto e una **correlazione moderata negativa** con il numero di defects. Anche in questo caso, il risultato coincide con le aspettative poiché è normale che un codice di grandi dimensioni richieda un effort e delle tempistiche maggiori rispetto ad uno di dimensioni più ridotte e che sia associato ad un prodotto con delle funzionalità più complesse. Tuttavia, per quanto riguarda la relazione con il numero di defects legati al progetto, ci troviamo nuovamente di fronte ad un risultato inaspettato: infatti,

emerge che un aumento delle dimensioni del codice prodotto coincida con una diminuzione del numero dei bug del software sviluppato.

Per quanto riguarda la **complessità funzionale**, si nota che essa presenta una **correlazione forte positiva** con la durata e l'effort del progetto come da aspettativa. Al contrario, presenta una **correlazione moderata negativa** con il numero di defects di progetto che dimostra come un elevato livello di complessità non risulti necessariamente in un elevato numero di errori nel codice. Questo risultato può essere legato al fatto che uno sviluppatore, di fronte ad un progetto complesso, abbia la tendenza a mantenere una soglia di attenzione maggiore e a limitare così il numero di errori.

Proseguendo l'analisi dei risultati, si evince che la **durata** del progetto mostra una **correlazione forte positiva** con l'effort di progetto, segno del fatto che questi due descrittori tendono a crescere di pari passo. Anche in questo caso, però, c'è una **correlazione moderata negativa** con il numero di defects di progetto, che rafforza le considerazioni fatte in precedenza.

Infine, l'ultima relazione rimasta da analizzare è quella tra l'**effort** e il numero di defects: la **correlazione** è **debole negativa**, segno del fatto che vengono sì confermati i risultati precedenti, ma che un aumento dell'effort porta ad una diminuzione del numero di defects inferiore rispetto ad un aumento della durata.

Di seguito, verranno presentati due esempi di analisi basata sulla regressione lineare legati alla stima predittiva dell'effort e della durata.

5.4 IL MODELLO DI REGRESSIONE LINEARE: STIMA DELLA DURATA DI PROGETTO

La **regressione lineare** è un metodo statistico di stima del valore atteso condizionato di una variabile dipendente Y, dati i valori di altre variabili indipendenti X (predittori). Nel caso di studio analizzato, è stato utilizzato un modello di **regressione lineare multipla**, in cui ci sono più variabili indipendenti utili a spiegare la variabile indipendente. La cui formula della retta di regressione è la seguente:

$$\boldsymbol{Y_i = \beta_0 + \beta_1 X_{1i} + \beta_2 X_{2i} + \ldots\ldots + \beta_k X_{ki} + u_i}$$

Dove i coefficienti β_i sono i coefficienti angolari delle variabili indipendenti X_i, il coefficiente β_0 è il valore atteso di Y quando tutte le X sono uguali a zero, u_i è l'errore statistico.

Per effettuare la regressione lineare della durata del progetto, sono state identificate la produttività e la complessità funzionale come variabili indipendenti. Lo strumento utilizzato per l'analisi è la funzione di **Data Analysis** di Microsoft Excel, che ha fornito in output il seguente risultato.

SUMMARY OUTPUT

Regression Statistics	
Multiple R	0,866048387
R Square	0,750039808
Adjusted R Square	0,727316154
Standard Error	15,0727991
Observations	25

ANOVA

	df	*SS*	*MS*	*F*	*Significance F*
Regression	2	14997,676	7498,838	33,00701	2,38001E-07
Residual	22	4998,164002	227,1893		
Total	24	19995,84			

	Coefficients	*Standard Error*	*t Stat*	*P-value*	*Lower 95%*	*Upper 95%*	*Lower 95,0%*	*Upper 95,0%*
Intercept	29,30887473	17,60641456	1,66467	0,110158	-7,204594237	65,82234371	-7,204594237	65,82234371
Productivity	-10,04103788	5,210100122	-1,92723	0,066963	-20,84612421	0,764048442	-20,84612421	0,764048442
Functional Complexity	10,36469508	1,497406333	6,921765	5,98E-07	7,259264414	13,47012575	7,259264414	13,47012575

Tabella 3. Regressione lineare legata alla durata di progetto.

I **coefficienti di regressione** β_i esprimono, sulla base del modello stimato, di quanto aumenta o diminuisce la variabile dipendente Y, in questo caso la durata, per ogni incremento unitario delle variabili dipendenti X_i (complessità funzionale e produttività). Dal risultato ottenuto, si nota come siano state confermate le conclusioni fatte in fase di studio della correlazione tra le variabili.

Per andare a misurare la bontà dell'adattamento del modello di regressione multipla, si utilizza il **coefficiente di determinazione** $\boldsymbol{R^2}$, che varia tra 0 e 1 ed esprime la frazione di varianza spiegata dal modello sul totale della varianza del fenomeno osservato. In questo caso, si è ottenuto $\boldsymbol{R^2 = 0,75}$ che mostra come il modello di regressione utilizzato spieghi il 75% della variabilità complessiva della durata e che sia, di conseguenza, molto affidabile.

Si possono, inoltre, effettuare i due seguenti tipi di test:

- **Test F**: serve per capire se il modello spiega una quota significativa della varianza di Y e l'effetto dell'inserimento di un nuovo predittore sulla varianza spiegata. In questo caso, essendo il **p-value** osservato minore del p-value teorico (0.05), il modello utilizzato spiega una quota significativa della varianza del fenomeno.
- **Test t**: serve a valutare la significatività statistica di un predittore all'interno del modello. In questo caso, il p-value associato alla complessità funzionale è minore di 0.05 e il p-value associato alla produttività è maggiore di 0.05. Ciò significa che la complessità funzionale spiega una quota significativa della varianza di Y, mentre la produttività non è statisticamente significativa.

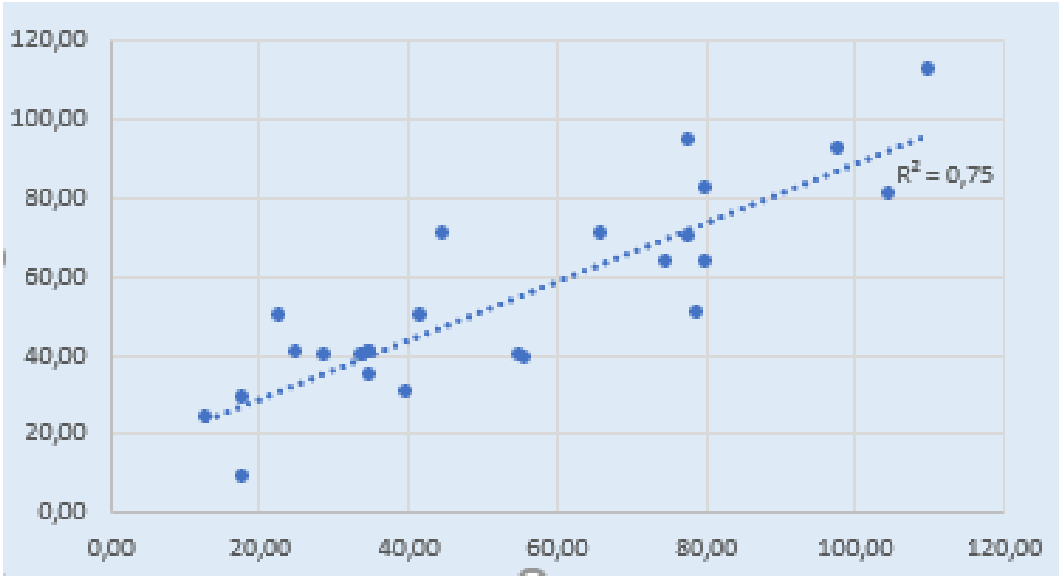

Grafico 12. Regressione lineare della durata.

5.5 IL MODELLO DI REGRESSIONE LINEARE: STIMA DELL'EFFORT DI PROGETTO

Nel caso di studio preso in esame, la stima dell'effort è proporzionale a quella del costo di progetto in quanto, a ciascun giorno di effort, corrisponde una tariffa fissa giornaliera.

In questo caso, per effettuare la regressione lineare dell'effort del progetto, sono state identificate la produttività, la complessità funzionale e la durata come variabili indipendenti. Lo strumento utilizzato per l'analisi è la funzione di **Data Analysis** di Microsoft Excel, che ha fornito in output il seguente risultato.

Multiple R	0,929563746							
R Square	0,864088757							
Adjusted R Square	0,844672865							
Standard Error	15,63919177							
Observations	25							
ANOVA								
	df	*SS*	*MS*	*F*	*Significance F*			
Regression	3	32655,0893	10885,03	44,5042	2,80858E-09			
Residual	21	5136,270704	244,5843					
Total	24	37791,36						
	Coefficients	*Standard Error*	*t Stat*	*P-value*	*Lower 95%*	*Upper 95%*	*Lower 95,0%*	*Upper 95,0%*
Intercept	-7,416547499	19,38442223	-0,3826	0,705859	-47,72866034	32,89556534	-47,72866034	32,89556534
Productivity	-7,757758708	5,844422666	-1,32738	0,198635	-19,911901	4,396383584	-19,911901	4,396383584
Functional Complexity	8,085783067	2,769624934	2,919451	0,008195	2,326032709	13,84553343	2,326032709	13,84553343
Duration	0,648200027	0,221212189	2,930218	0,007997	0,188164095	1,108235958	0,188164095	1,108235958

Tabella 4. Regressione lineare legata all'effort di progetto.

Anche in questo caso, in base al risultato ottenuto, sono state confermate le conclusioni raggiunte in fase di studio della correlazione tra le variabili.

Per quanto riguarda la bontà dell'adattamento del modello di regressione multipla, il **coefficiente di determinazione** $\boldsymbol{R^2}$ ottenuto è pari a 0.86 e ciò dimostra come il modello di regressione utilizzato spieghi il 86% della variabilità complessiva dell'effort e che mostri, di conseguenza, un adattamento anche superiore a quello legato alla durata. Questo risultato è ulteriormente confermato dal dato relativo al **test F**, che presenta un **p-value** inferiore a 0.05.

Infine, andando ad osservare i dati relativi al **test t**, si nota come la complessità funzionale e la durata presentino un **p-value** inferiore a 0.05 e che spieghino, di conseguenza, una quota significativa della varianza dell'effort. Al contrario, anche in questo caso la produttività ha un p-value maggiore di 0.05 e si rivela essere non statisticamente significativa.

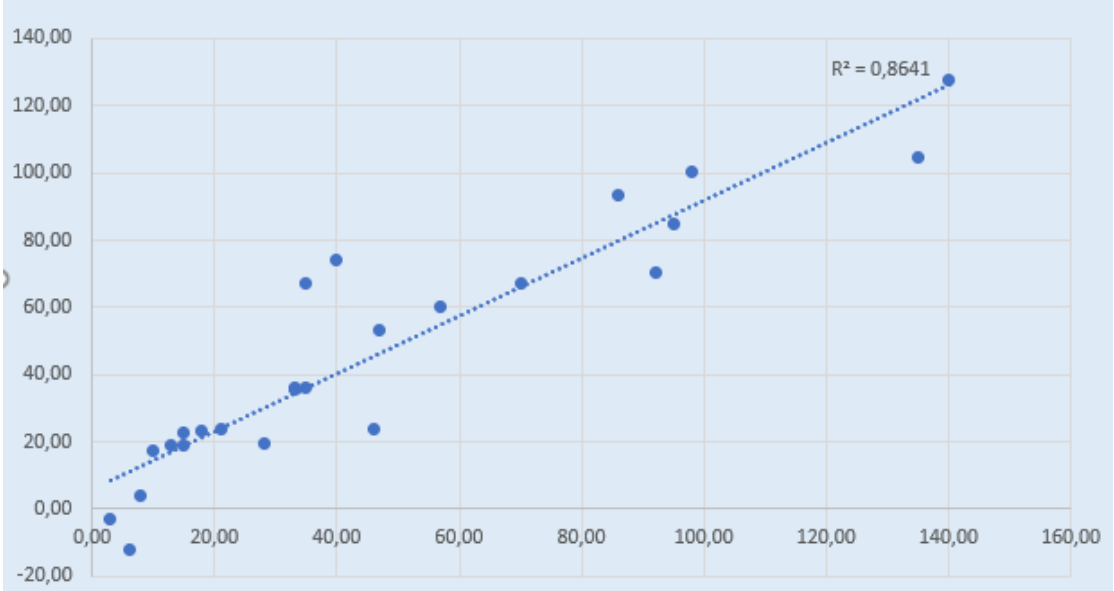

Grafico 13. Regressione lineare dell'effort.

5.6 SENSITIVITY ANALYSIS

Una volta scelto l'algoritmo di machine learning da utilizzare per la stima e da confrontare con la metodologia di regressione lineare e una volta ottenuto un risultato in output, è possibile effettuare una **sensitivity analysis** utile a capire quale sia il metodo di stima più adatto al database preso in esame.

La sensitivity analysis è utile a testare l'effetto delle fluttuazioni delle variabili di input sul comportamento del modello. Di conseguenza, permette di capire se il modello è adatto per il processo preso in esame, la qualità del modello, quali sono i fattori che contribuiscono a rendere variabile l'output, la regione all'interno dello spazio dei fattori di input in cui la variazione del modello è massima e le interazioni tra i vari fattori. Esistono due diversi approcci alla sensitivity analysis:

- **Drastic sensitivity analysis**: studia l'effetto di cambiamenti drastici alle variabili di input come, ad esempio, variare un fattore dal suo minimo al suo massimo. Questo approccio fornisce un range di valori di input che indicano le regioni all'interno delle quali c'è una capacità predittiva ottimale.
- **Marginal sensitivity analysis**: studia l'effetto di cambiamenti infinitesimali alle variabili di input. Si effettua aggiungendo un rumore artificiale con una distribuzione normale al modello, al fine di andare a capire gli effetti dei cambiamenti delle variabili di input sull'efficienza del modello. Inoltre, permette di identificare i valori limite di capacità predittiva degli indici utilizzati nel modello.

La sensitivity analysis si può effettuare su una sola variabile o su gruppi di variabili e i risultati ottenuti per le due metodologie di stima prese in esame vanno confrontati al fine di trovare la metodologia ottimale. Per valutare e confrontare la qualità predittiva delle metodologie di stima prese in esame, si possono utilizzare i seguenti indici di errore, il loro valore medio e la loro deviazione standard:

- L'**errore quadratico medio** (MSE), che mostra la discrepanza quadratica media che c'è tra i valori dei dati osservati ed i valori dei dati stimati.

- Il **balanced relative error** (BRE), che è definito come il rapporto tra il valore assoluto dell'errore e l'errore minimo osservato.
- Il **magnitude of error relativo alla stima** (MER), che è definito come il rapporto tra il valore assoluto dell'errore e il valore osservato.
- Il **magnitude of relative error** (MRE), che è definito come il rapporto tra il valore assoluto dell'errore e il valore stimato.

Index name	Basic formula	Mean	Deviation
Mean square error	$MSE = (\widehat{Y} - Y_i)^2$	$MMSE = \frac{\sum(\widehat{Y}_i - Y_i)^2}{N}$	$DMSE = \left(\frac{\sum(MMSE - Y_i)^2}{N}\right)^{1/2}$
Balanced relative error	$BRE = \frac{\lvert\widehat{Y}_i - Y_i\rvert}{\min(\widehat{Y}_i - Y_i)}$	$MBRE = \frac{\sum\frac{\lvert\widehat{Y}_i - Y_i\rvert}{\min(\widehat{Y}_i - Y_i)}}{N}$	$DBRE = \left(\frac{\sum(MBER - Y_i)^2}{N}\right)^{1/2}$
Magnitude of error relative to estimate	$MER = \frac{\lvert\widehat{Y}_i - Y_i\rvert}{\widehat{Y}_i}$	$MMER = \frac{\sum\frac{\widehat{Y}_i - Y_i}{\widehat{Y}_i}}{N}$	$DMER = \left(\frac{\sum(MMER - Y_i)^2}{N}\right)^{1/2}$
Magnitude of relative error	$MRE = \frac{\lvert\widehat{Y}_i - Y_i\rvert}{Y_i}$	$MMRE = \frac{\sum\frac{\lvert\widehat{Y}_i - Y_i\rvert}{Y_i}}{N}$	$DMRE = \left(\frac{\sum(MMRE - Y_i)^2}{N}\right)^{1/2}$

$\widehat{Y}_i$ – actual ith value; Y_i – estimated ith values; N – number of samples, $i = 1, N$.

Tabella 5. Indici di valutazione delle metodologie di stima.

Una volta raccolti tutti i dati relativi ai suddetti indici, si può effettuare un'analisi comparativa dei risultati ottenuti per entrambe le metodologie di stima: minori sono i valori di questi indici di errore, migliore è la capacità predittiva della metodologia di stima presa in esame.

CONCLUSIONI

La stima dei tempi e dei costi di un progetto software rimane un problema complesso che sta attirando l'attenzione di molti ricercatori. Uno degli svantaggi legati all'utilizzo di algoritmi di machine learning è la struttura chiusa e poco chiara del processo di computazione. Risulta, infatti, difficile per un project manager che non sia uno specialista in algoritmi matematici capire quali siano i meccanismi di funzionamento di queste metodologie di stima predittiva. Inoltre, la credibilità di una stima dipende fortemente dalla trasparenza del metodo utilizzato, dai dati, dalle definizioni e dalle assunzioni che vengono fatte a monte del processo.

Al fine di effettuare una stima predittiva dei tempi e dei costi in uno stadio iniziale del processo di sviluppo, è necessario identificare i descrittori di progetto che possono essere utilizzati come predittori e le relazioni che intercorrono tra di essi. Tuttavia, bisogna tenere conto del fatto che le caratteristiche del prodotto nelle fasi iniziali del progetto tendono a cambiare nelle fasi successive in particolare nei progetti di sviluppo software. Inoltre, questo tipo di progetti si presta a cambiamenti dei requisiti in corso d'opera che possono cambiare la complessità e le dimensioni del prodotto nel tempo, rendendo ancora più difficoltosa ed imprecisa una stima nelle fasi iniziali del progetto.

Sebbene la raccolta accurata dei dati legati ai progetti passati sia fondamentale per poter effettuare una stima affidabile, è importante sottolineare che la tecnologia, i tools e le caratteristiche del progetto che viene stimato possono essere molto diverse rispetto a quelle dei progetti completati in passato e ciò fa sì che il risultato del modello di stima possa diventare imprevedibile. Di conseguenza, nel momento in cui si accetta l'assunzione che il database storico dei progetti presenti caratteristiche comparabili con eventuali progetti nuovi, è importante tenere conto anche di questa limitazione.

Al fine di definire un framework ben delineato a monte della procedura di stima, una volta terminati la fase di pulizia del database dagli outliers e lo studio legato alle relazioni tra i principali descrittori di progetto, è necessario andare ad

identificare l'algoritmo o gli algoritmi di machine learning di cui si vuole studiare l'efficacia predittiva. Solo al termine della sensitivity analysis e di un'analisi comparativa tra i vari metodi che si vogliono testare, sarà possibile individuare l'algoritmo più adatto ad effettuare la stima dei costi e dei tempi dei progetti software legati al database in esame.

Ad oggi, non è ancora stato identificato un metodo di stima ideale per qualsiasi tipo di progetto software, poiché la scelta dipende dalla struttura funzionale dell'azienda e dalle metodologie di project management utilizzate al suo interno. Per questo motivo, è importante impostare un lavoro di analisi a monte al fine di identificare la metodologia di stima di tempi e costi da utilizzare e di ridurre il più possibile gli errori legati alla previsione. Questa tesi ha cercato di dare una panoramica generale degli step da seguire, ma in futuro saranno necessari studi più approfonditi per perfezionare il processo di scelta dell'algoritmo di machine learning utile ad effettuare stime predittive di progetti software.

INDICE DELLE FIGURE

INDICE DEI GRAFICI

INDICE DELLE TABELLE

BIBLIOGRAFIA

Relich Marcin, Pawlewski Pawel, 2017, *A case-based reasoning approach to cost estimation of new product development*.

Pham Do Huan, Do Nhon, 2019, *A consulting system for estimating costs of an information technology hardware project based on law of public investment*.

Sommestad Teodor, Sandgren Sofia, 2009, *A framework for Assessing the Cost of IT Investments*.

Hamdan Khaled, El Khatib Hazem, 2017, *A Software Cost Ontology System for Assisting Estimation of Software Project Effort for Use with Case-Based Reasoning*.

Berlin Stanislav, Raz Tzvi, Moshe Zviran, 2008, *Comparison of estimation methods of cost and duration in IT projects*.

Usman Muhammad, Mendes Emilia, Britto Ricardo, 2014, *Effort Estimation in Agile Software Development: A Systematic Literature Review.*

Tayebi Moosavi Rohollah, Ostadzadeh Shervin Mazaheri Samaneh, 2009, *Employing MDA in PRINCE2 Framework*.

Stamelos Ioannis, Angelis Lefteris, Morisio Maurizio, 2002, *Estimating the development cost of custom software*.

Dasheng Xue, Shenglan Hao, 2017, *Estimation of Project Costs Based on Fuzzy Neural Network*.

Irgens Chris, Landryovà Lenka, 2004, *Evaluating a Software Costing Method Based on Software Features and Case Based Reasoning*.

Wille Cornelius, Fiegler Anja, Neumann Robert, Dumke Reiner, 2011, *Evidence-Based Evaluation of Effort Estimation Methods*.

Patel Kirit, 2009, *Information Technology in Using Project Management Methodologies*.

Calefato Fabio, Lanubile Filippo, Mallardo Teresa, 2010, *A Controlled Experiment on the Effects of Synchronicity in Remote Inspection Meetings*.

Arain Manzoor Faisal, 2008, *IT-based approach for effective management of project changes: A change management system (CMS).*

Benton Bruce, 2009, *Model-Based Time and Cost Estimation in a Software Testing Environment.*

Hamdan Khaled, El Khatib Hazem, Shuaib Khaled, 2010, *Practical Software Project Total Cost Estimation Methods.*

Choetkiertikul Morakot, Dam Khanh Hoa, Tran Truyen, Ghose Aditya, 2015, *Predicting Delays in Software Projects using Networked Classification.*

Zhenyou Li, 2014, *Predicting Project Effort Intelligently in Early Stages by Applying Genetic Algorithms with Neural Networks.*

Hewagamage Champa, 2011, *Redesigned Framework and Approach for IT Project Management.*

Azzeh Mohammad, 2013, *Software Cost Estimation Based on Use Case Points for Global Software Development.*

Hamza Haitham, Kamel Amr, Shams Khaled, 2013, *Software Effort Estimation using Artificial Neural Networks: A Survey of the Current Practices.*

Benala Rao Tirimula, Dehuri Satchidanada, 2013, *Software Effort Prediction Using Learning (Clustering) and Functional Link Artificial Neural Networks.*

Jie Jun, 2018, *Tailoring a Project Management Methodology that Suits One's Needs.*

Frey Thorsten, 2014, *Governance Arrangements for IT Project Portfolio Management,* Springer Gabler.

De Marco Alberto, Narbaev Timur, 2013, *An Earned Schedule-based regression model to improve cost estimate at completion.*

De Marco Alberto, Narbaev Timur, 2013, *Combination of Growth Model and Earned Schedule to Forecast Project Cost at Completion.*

De Marco Alberto, Narbaev Timur, Rosso Marta, 2016, *Nonlinear cost estimates at completion adjusted with risk contingency.*

De Marco Alberto, Sciuto Francesco, Warburton Roger D.H, 2017, *Earned Schedule Formulation using nonlinear cost estimates at completion.*

De Marco Alberto, Sciuto Francesco, Warburton Roger D.H, 2013, *Earned Schedule Formulation using nonlinear cost estimates at completion.*

Stock James H., Watson Mark W., 2015, *Introduction to Econometrics,* Always Learning Pearson.

SITOGRAFIA

https://www.pmi.org/learning/library/agile-project-estimation-techniques-6110

https://www.humanwareonline.com/project-management/center/tecniche-di-stima/

https://twproject.com/it/blog/tecniche-di-stima-di-un-progetto/

https://towardsdatascience.com/machine-learning-interpretability-techniques-662c723454f3