POLITECNICO DI TORINO
Corso di Laurea in Ingegneria Gestionale

Tesi di Laurea Magistrale

Analisi degli impatti della fase di follow up sui costi di progetti IT mediante modelli predittivi

Relatore
Prof. Alberto De Marco

Candidato
Martina Mercurio

Anno accademico 2021-2022

Indice

1 Introduzione 6

2 Il Project Management 8

2.1 La storia 8

2.2 I principi 10

2.3 Le tipologie 16

2.3.1 La metodologia Waterfall 16

2.3.2 La metodologia Agile 18

2.3.3 Il processo Hybrid 21

3 IT Project Management 23

3.1 Information Technology 23

3.2 IT Project Management 25

3.3 Il ciclo di vita del software 28

3.4 La manutenzione 31

4 Il Machine Learning 37

4.1 Modelli white-box e black-box 39

4.2 Reti Feed-forward 41

4.3 Reti Feed-forward backpropagation 43

4.4 Reti neurali ricorrenti 45

4.5 Il machine learning per progetti software 46

5 Caso di studio: analisi di progetti software in una società di consulenza IT 48

5.1 L'esecuzione del progetto in azienda 51

5.1.1 Lo sviluppo 52

5.1.2 L'Application Test 53

5.1.3 Il System Test 53

5.1.4 Lo User Acceptance Test 55

5.1.5 Il rilascio in produzione 56

5.1.6 Follow up e Manutenzione 56

5.2 Analisi del Database 57

5.3 Variabili del progetto 59

5.4 Implementazione dei modelli 63

5.4.1 Regressione lineare multipla 68

5.4.2 Regressione polinomiale multipla 69

5.4.3 Albero decisionale 70

5.4.4 Foresta casuale 71

5.5 Analisi dei risultati 72

6 Conclusioni 76

Bibliografia 78

Capitolo 1

Introduzione

Il presente lavoro di tesi nasce durante l'esperienza di stage curriculare svolta all'interno di una società internazionale di consulenza specializzata in innovazione tecnologica e ingegneristica. Nello specifico, il tirocinio è stato esercitato nell'ambito del Project Management nel team di consulenza informatica, fornitore di servizi IT per un noto istituto bancario.

Il proposito che ci si auspica di raggiungere con tale elaborato è quello di ottenere un'ottimizzazione dei processi aziendali dedicati alla gestione dei progetti. L'idea scaturisce dall'attuale mancanza di un modello previsionale per quanto concerne la mole lavorativa associabile ad un progetto nella fase immediatamente successiva al suo rilascio in produzione. L'obiettivo è, quindi, quello di riuscire a prevedere, con un livello di dettaglio adeguato, il peso di una commessa in termini temporali e, conseguentemente, economici, nel cosiddetto periodo di follow up, in modo da poterlo eventualmente includere in fase di stima progettuale. Con una visione più generale, lo scopo diviene, pertanto, la possibilità di poter apportare un valore aggiunto alla gestione globale e totalizzante dei progetti, ottimizzandone i processi di pianificazione e stima.

La prima parte della tesi consisterà in un excursus dottrinale su quelle che rappresentano le best practices nel settore del Project Management, della fornitura di servizi dell'Information Technology e del ciclo di vita del software. Si analizzeranno le varie fasi dell'evoluzione del prodotto in questione sottolineando, attraverso il ricorso ad articoli scientifici, il gap teorico informativo sul tema della manutenzione, evidenziando la carenza di studi e attenzione appropriata verso tale argomento. Si procederà con l'analisi di modelli di machine learning per apprenderne le funzionalità e le casistiche in cui possano essere

utilizzati, fornendo, così, una panoramica teorica sui relativi argomenti. Si proseguirà, poi, analizzando il caso specifico della società in esame, indagando la realtà aziendale ed i suoi processi caratteristici. Il *core* dell'elaborato, nonché il suo obiettivo preminente, sarà quello di procedere con l'individuazione delle principali variabili di interesse ricavabili da database di progetti IT, e la disamina delle possibili relazioni che sussistono tra esse. L'intento è di giungere all'individuazione del modello che meglio si adatti all'interpretazione dei vari fattori, elaborando una previsione accurata degli impatti che questi possono avere sui costi in fase di follow up.

Capitolo 2

Il Project Management

2.1 La storia

Il Project Management è definito, dal Project Management Institute, come l'applicazione di conoscenze, attitudini, strumenti e tecniche alle attività di un progetto al fine di conseguirne gli obiettivi che risultano essere unici e delimitati da vincoli di tempo, costi, qualità e con risorse circoscritte [1]. Così, l'Istituto descrive l'insieme di tecniche volte a organizzare, pianificare, dirigere e controllare le risorse per il raggiungimento di un obiettivo relativamente di breve termine, che è stato fissato per portare a termine traguardi specifici.

Nonostante gli studi e le dottrine dedicate a tale disciplina siano relativamente recenti, le prime evidenze concrete della messa in atto di questa materia possono essere riscontrate, sebbene in modo primordiale, in tempi molto lontani: si può considerare che il ruolo del Project Manager sia stato ricoperto, se pur inconsapevolmente e in maniera non strutturata, già nelle antichissime civiltà. Si pensi, ad esempio, alla costruzione della Sfinge, la celebre figura mitologica risalente al 2.500 a.C., e simbolo della antica civiltà egiziana. È evidente che la sua creazione non sia stata il mero frutto di un'improvvisazione, ma la rappresentazione tangibile di un'idea preliminare ben chiara, attraverso l'impiego di risorse, materiali e umane, ben organizzate.

È solo alla fine dell'800, però, che ha inizio la trasformazione dell'idea di progetto, con approcci che tendono ad assumere la forma più attuale del Project Management quale naturale evoluzione delle teorie di organizzazione aziendale, che si erano andate affermando proprio in quel periodo di crescita di complessità del business mondiale. Attorno agli anni Ottanta dello stesso secolo, infatti, Frederick Taylor

(1856-1915), imprenditore ed ingegnere statunitense, inizia i suoi studi sulla ricerca dei metodi per il miglioramento dell'efficienza nella produzione. Fino ad allora, si riteneva che l'unico modo per incrementare la produttività fosse quello di far lavorare le risorse per più tempo e più duramente. Taylor stravolge questa teoria, dimostrando che l'obiettivo di un progetto può essere raggiunto più facilmente migliorando il processo utilizzato per il raggiungimento dell'obiettivo stesso. Negli anni della Prima Guerra Mondiale, Henry Laurence Gantt (1861-1919), ingegnere meccanico, sviluppa un metodo per ottimizzare la sequenza di lavori necessari alla costruzione della flotta navale. I suoi grafici, i Gantt charts, sono ancora oggi diffusamente utilizzati dai moderni Project Manager.

A partire dagli anni della Seconda Guerra Mondiale vengono alla luce i primi veri progetti strutturati secondo una concezione moderna del Project Management. Tra questi spicca il Progetto Manhattan, lanciato dal governo degli USA con l'obiettivo di preparare ordigni atomici, e considerato da molti l'evento simbolicamente rappresentante la nascita del Project Management. È proprio a partire dalla metà del Secolo scorso che il Project Management inizia ad essere reputato una vera e propria disciplina; da questo momento in poi, vi è un susseguirsi di nuove teorie e metodologie volte a migliorare la totale gestione dei sistemi ampliando le conoscenze su concetti fondamentali come i cambiamenti organizzativi, le strategie di business e le relazioni interpersonali. La sempre crescente complessità dei progetti, infatti, induce gli studiosi ad elaborare nuovi strumenti che facilitino la gestione dei progetti (tra i più famosi il metodo del percorso critico, critical path method, e i grafici PERT). La tendenza era quella di ritenere possibile accrescere la complessità dei programmi di lavoro, ponendo un focus sull'utilizzo attento e mirato delle risorse, e su una qualità finale sempre migliore, senza mai trascurare la necessità di riuscire a mantenere contenuti i costi.

Nel 1969 viene fondato negli Stati Uniti il Project Management Institute, un'organizzazione professionale senza fini di lucro, per la gestione dei progetti, riconosciuta a livello internazionale come l'ente più autorevole in tale materia. PMI ha descritto i suoi obiettivi nel 1975 come "favorire il riconoscimento della necessità di professionalità nella gestione dei progetti; fornire un forum per il libero scambio di problemi, soluzioni e applicazioni di gestione dei progetti; coordinare gli sforzi di ricerca industriale e accademica; sviluppare terminologia e tecniche comuni per migliorare comunicazioni; fornire interfaccia tra utenti e fornitori di sistemi hardware e software; e fornire linee guida

per l'istruzione e lo sviluppo della carriera nel campo della gestione dei progetti"[1].

Al giorno d'oggi il ruolo del Project Manager viene finalmente considerato cruciale e decisivo per la riuscita di un progetto, dando vita ad una richiesta sempre crescente di figure professionali di tal genere in tutto il mondo, sia nel settore pubblico sia in quello privato.

2.2 I principi

Per poter analizzare quali siano i principi fondamentali su cui si basa il Project Management, risulta indispensabile fornire delle delucidazioni in merito al concetto di progetto. Le definizioni esistenti sono molteplici, tuttavia tra le più accreditate si evidenziano le seguenti:

- "Uno sforzo temporaneo intrapreso per creare un prodotto o un servizio univoco" [2]
- "Un insieme di attività tra loro correlate e interdipendenti, volte al raggiungimento di un obiettivo preciso, con un limite di tempo determinato, un budget di risorse stabilite, che vengono avviate alla ricerca di un aumento di valore per l'azienda o per il soddisfacimento delle esigenze del cliente" [2]
- "Uno sforzo complesso, di regola di durata inferiore ai tre anni comportante compiti interrelati eseguiti da varie organizzazioni, con obiettivi, schedulazioni e budget ben definiti" [3]
- "È un insieme di persone e di altre risorse temporaneamente riunite per raggiungere uno specifico obiettivo, di solito con un budget ben predeterminato ed entro un periodo stabilito" [4]
- "Un'unica serie di attività volta a produrre un risultato definito, con una precisa data di inizio e fine, e una precisa allocazione di risorse" [5]

[1] https://www.centocinquanta.it/2019/12/04/la-gestione-dei-progetti\-internazionali-tra-standard-e-multiculturalita/

[2] SDA Bocconi, Divisione Ricerche 1999

[3] Archibald 1994

[4] Graham 1990

[5] Harvard Business School

È evidente che, nelle varie definizioni, vi siano alcuni elementi distintivi e comuni che possano far convergere le varie interpretazioni ad una descrizione che sia esaustiva. I fattori ricorrenti che si evincono sono la presenza di un obiettivo da completare nel rispetto di determinati vincoli; la necessità di realizzare, entro tempi ben definiti, attività coordinate macchinosamente; la limitatezza di risorse umane, finanziarie e strumentali; ed infine, il carattere multifunzionale e pluri-disciplinare rispetto alla struttura organizzativa.

La singolarità dell'obiettivo definisce la peculiarità del progetto a confronto con attività ordinarie, mentre, la limitazione temporale si manifesta non solo nella disponibilità limitata di risorse, ma anche nella necessità di effettuare i vari task entro una precisa finestra di opportunità legata all'oggetto stesso del progetto. La principale caratteristica identificativa di un progetto è la sua novità, è un passo verso l'ignoto, pieno di rischi e incertezze. Non esistono mai due progetti esattamente identici: per quante analogie potranno esservi, ci saranno sempre aspetti che li renderanno unici. Alcuni progetti nascono gradualmente, altri si dissolvono lentamente, rendendo complessa l'individuazione precisa delle date di inizio e di fine; tuttavia, la maggior parte dei progetti, presenta anche delle milestones significative, ovvero degli eventi cruciali rappresentanti i traguardi chiave da raggiungere durante l'intero sviluppo del progetto stesso. Il lasso di tempo che intercorre tra la data di avvio e quella di termine del programma è definito come il ciclo di vita del progetto. Infatti, nonostante sia stata sottolineata l'unicità di ogni progetto, è importante riconoscere le usuali e generalizzate fasi che caratterizzano lo sviluppo del progetto stesso, e che risultano far confluire in un iter universale le varie pianificazioni dei lavori.

La sesta edizione del PMBOK afferma che il ciclo di vita di un progetto definisce l'insieme di fasi che attraversa un progetto, dall'inizio al suo completamento, specificando per ognuna di esse il tipo di lavoro tecnico svolto, le persone coinvolte, l'accuratezza, la completezza, e la successiva approvazione dei deliverable da produrre prima di procedere con la fase successiva. Analizzando questa spiegazione si deduce come questo ciclo sia il risultato di sforzi sia gestionali, sia tecnici, in termini di costo e di personale. Per quanto concerne gli aspetti gestionali, o di Project Management, si concretizzano nell'inizio, nella definizione, pianificazione, esecuzione, controllo e chiusura del progetto; tutti concetti comuni e condivisi nella maggior parte dei lavori. Analogamente, quelli tecnici, che si evolvono in parallelo,

seppur siano inevitabilmente specifici per ogni settore di applicazione e legati alla tipologia di servizio o prodotto, è possibile ricondurli alle macro fasi di fattibilità, definizione dei requisiti di progettazione, sviluppo, implementazione e verifica di test, manutenzione finale.

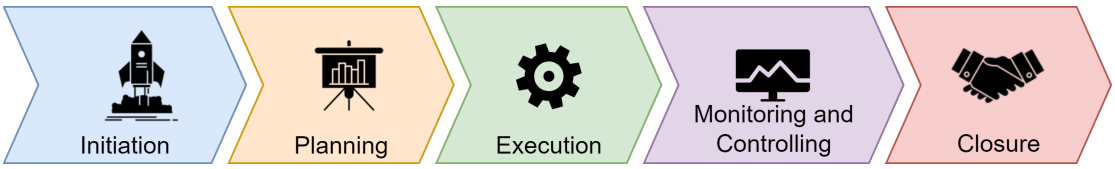

Figura 2.1. Fasi del Progetto

La prima fase è quella di avvio del progetto: probabilmente la più delicata dell'intero processo; vengono stabiliti gli obiettivi, individuate le risorse a disposizione ed identificati i ruoli. In questo step iniziale viene condotto uno studio di fattibilità, una valutazione economica degli effetti del progetto, ovvero un'analisi costi/benefici, ed infine viene decretato lo scopo ultimo del progetto.

La seconda fase è la pianificazione; spesso sottovalutata, è in realtà una fase cruciale che può determinare la riuscita o l'insuccesso di un progetto. In particolare, si svolgono molteplici attività come definire un piano di costi dettagliato, organizzare le risorse coinvolte, definire un'attenta strategia per affrontare i rischi ed impostare la gestione degli stakeholders. È durante questa fase che vengono redatti alcuni documenti fondamentali come il Diagramma di Gantt e la Work Breakdown Structure. L'output di questa fase è il piano di progetto (project plan), documentazione che viene poi sottoposta al vaglio di tutti gli stakeholders per l'approvazione finale.

La terza fase è la fase di esecuzione del progetto. Il team realizza concretamente il piano di lavoro, rendendo tangibile la pianificazione teorica sviluppata in precedenza. Questo step è fortemente correlato con quella che spesso viene considerata la quarta fase dello sviluppo del progetto: la fase del monitoraggio e controllo; giunti a questo punto è indispensabile monitorare costantemente i progressi eseguiti. Il Project Management effettua, infatti, continui check per verificare l'avanzamento del progetto e misurare le prestazioni delle attività. In questo modo è possibile mantenere il controllo sulla direzione del piano programmato, confrontando i rapporti di avanzamento con il progetto originale ed apportare, laddove necessario, le correttive adeguate che vengono registrate come scostamenti dal piano iniziale.

La quinta ed ultima fase è la chiusura del progetto. Giunti al termine del processo, l'enfasi ricade sul rilascio dei risultati finali al cliente, sulla consegna della documentazione del progetto all'azienda, sulla risoluzione dei contratti con i fornitori, sul rilascio delle risorse e sulla comunicazione della chiusura del progetto a tutte le parti interessate. È in questo contesto che risulta di fondamentale importanza condurre un'analisi per tracciare quali siano stati i punti di forza e di debolezza durante l'intero arco temporale di sviluppo del lavoro; si costruirà così un background di conoscenze e competenze empiriche utili per l'elaborazione dei progetti futuri.

A valle dell'esplicitazione della varie fasi, è visibile la dipendenza del successo del progetto dal rispetto di tre principali vincoli: costi, tempi e ambito. La combinazione di queste tre variabili dà vita al noto *triplice vincolo*, ovvero la necessità di trovare un equilibrio tra i diversi fattori al fine di garantire un prodotto finale di qualità. A tal proposito, la figura 2.2 dimostra graficamente la stretta correlazione delle grandezze citate, in relazione con la qualità ottenibile dal tuning delle variabili.

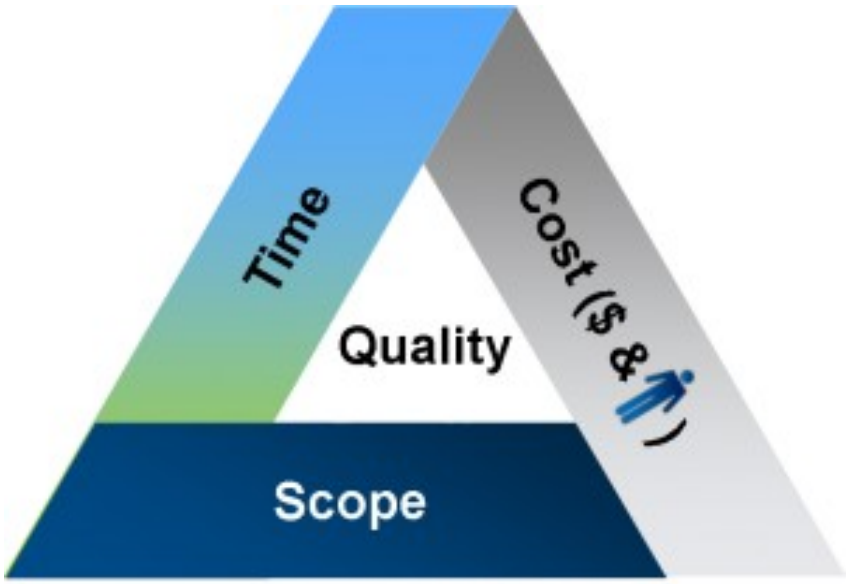

Figura 2.2. Triplice Vincolo (fonte [3])

Il tempo, il primo dei fattori in analisi, va considerato sia dal punto di vista quantitativo che qualitativo e comprende i seguenti aspetti:

- Ore lavorative dedicate al progetto;
- Obiettivi interni;
- Cronologia generale del piano di lavoro;
- Tempo destinato alla strategia e pianificazione.

Un aumento della variabile scopo, o una riduzione del budget, come dimostrato nella prima sequenza della figura 2.3, comporteranno inevitabilmente una compensazione a livello temporale, prolungando le scadenze o apportando modifiche al programma.

Il costo, che assume la duplice interpretazione di budget e risorse, include tutti gli strumenti e il supporto necessari allo sviluppo del progetto, la totalità delle voci che possano essere riconducibili all'aspetto economico tra cui:

- Numerosità del team;
- Strumenti e attrezzature;
- Budget complessivo.

La relazione che si instaura tra tempo e costi è inversamente proporzionale: infatti, un incremento dei tempi implicherà una riduzione dei costi, al contrario, una diminuzione del lasso temporale comporterà una maggiorazione del budget necessario; tale corrispondenza si riscontra nella seconda sequenza della figura 2.3.

L'ambito, o scopo, l'ultima delle variabili in esame, è il calibro del progetto ed include i seguenti aspetti:

- Numero dei deliverables;
- Complessità totale del progetto;
- Grado di dettaglio richiesto;
- Standard di qualità desiderato.

Questo fattore implica una relazione di diretta proporzionalità nei confronti delle variabili di costo e tempo. Come si evince dalla terza sequenza della figura 2.3, all'estendersi dello scopo è necessario incrementare conseguentemente anche entrambi gli altri termini per far fronte ad un progetto più grande.

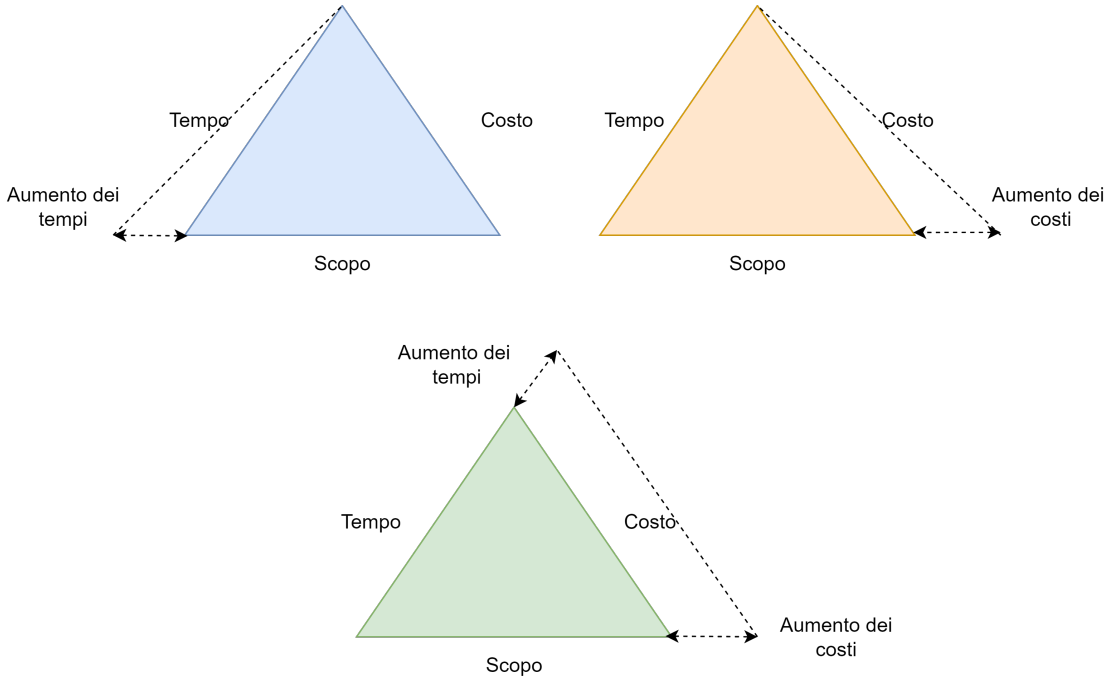

Figura 2.3. Relazioni tra le grandezze del Triplice Vincolo

Il complesso compito del Project Manager è proprio quello di mantenere in equilibrio le tre componenti del triangolo, bilanciando il trade off, rispettando i vincoli ed ottenendo il miglior livello di qualità possibile.

2.3 Le tipologie

La metodologia di gestione del progetto è definita dal Project Management Institute come "un insieme di metodi, tecniche, procedure, regole, modelli e migliori pratiche utilizzate in un progetto" [6]. Si basa comunemente su uno specifico approccio di Project Management, che definisce un insieme di principi e linee guida che definiscono il modo in cui un progetto viene gestito. I modelli procedurali possono essere differenziati in metodi che seguono un classico processo a cascata e metodi agili che seguono un approccio iterativo basato su test.

2.3.1 La metodologia Waterfall

La metodologia Waterfall è stata elaborata dal Dr. Winston W. Royce nel 1970 nel paper "Managing the development of large software systems" e rappresenta 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. Tutte le fasi sono contraddistinte da molteplici attività che generano i famigerati deliverables, da concludere prima che inizi la fase seguente. Vi è, inoltre, la possibilità di assegnare alle diverse fasi, risorse diversificate a seconda delle proprie competenze.

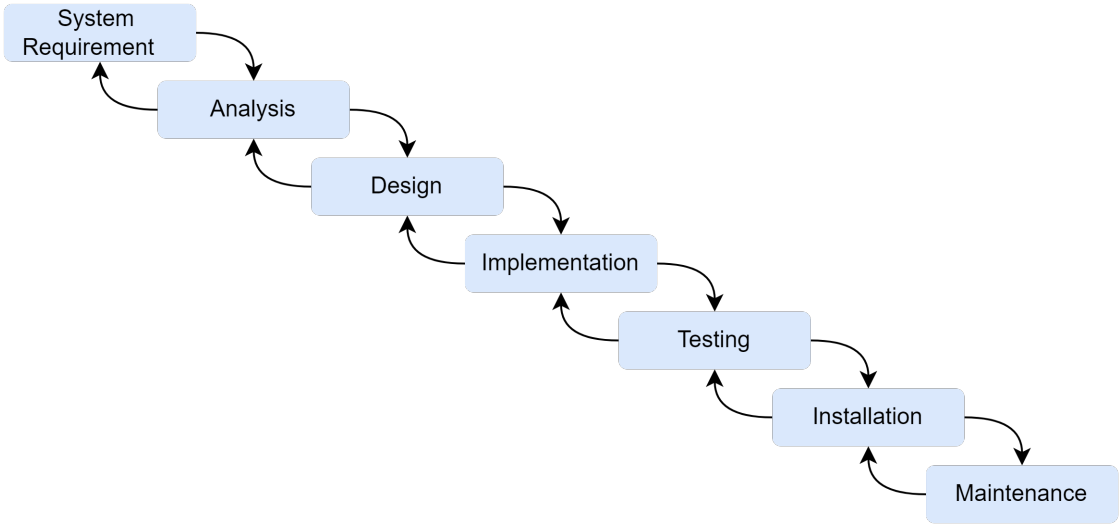

Figura 2.4. Fasi della metodologia Waterfall

[6] Project Management Institute, 2008

La figura 2.4 ben chiarifica il processo a cascata e definisce gli step da superare lungo il percorso che conduce alla realizzazione del progetto. Il flusso prevede l'analisi dei requisiti, la progettazione del concept, la concreta realizzazione, i test e le verifiche di corrispondenza alle specifiche, ed infine il mantenimento.

Un ulteriore aspetto da sottolineare è la possibilità della presenza di iterazioni all'interno del processo. Addentrandosi nell'attività progettuale ed aumentando la granularità dei dettagli, si origina un'iterazione tra ciascuna fase con la precedente e la successiva. Questo fenomeno diviene rilevante nel caso in cui emergano problemi in un determinato step; in tal caso può essere necessario effettuare un riesame dello stadio antecedente o, in alcune circostanze, dell'intero progetto. Quest'ultimo fattore si traduce in criticità qualora il programma debba essere soggetto a modifiche di tipo "disruptive", che possono generare un incremento significativo su tempi e costi.

Nella figura sotto rappresentata è riportato il trend del rischio per i progetti realizzati con la metodologia waterfall 2.5.

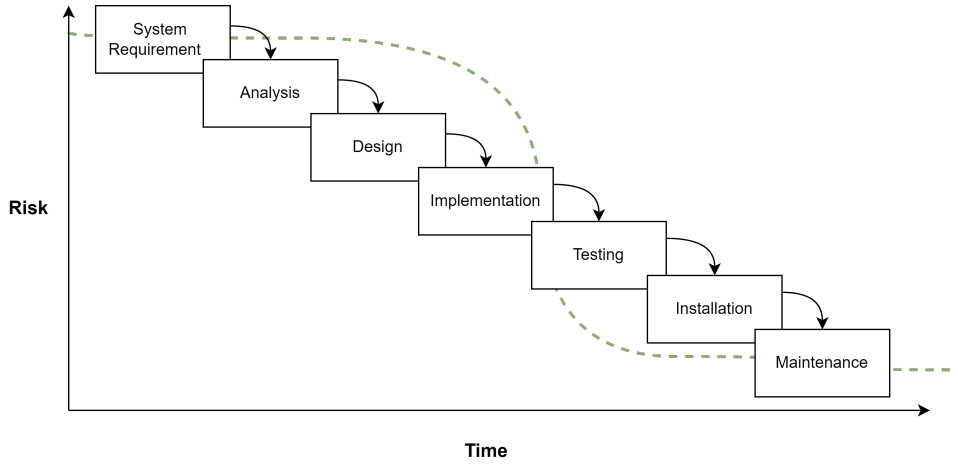

Figura 2.5. Curva di rischio della metodologia Waterfall

L'approccio Waterfall ha acquisito nel tempo sempre più consensi da parte di numerosi Project Manager che lo adoperano nelle principali aziende a livello mondiale; questa metodologia si è affermata largamente per la gestione di progetti in cui si desidera conseguire cospicui standard di qualità, in particolare in ambito software. La profonda attenzione per la pianificazione e l'elaborazione di una documentazione dettagliatamente tecnica, caratterizzano ogni singola fase svolta; questa vigile scrupolosità in fase di progettazione è il sintomo di un atteggiamento preventivo nei confronti di eventuali difetti di progettazione che potrebbero emergere procedendo nei vari step.

Il metodo Waterfall assicura, quindi, un'accurata formalizzazione e descrizione dei requisiti che consentono di individuare ed analizzare, a partire dalle fasi preliminari, i potenziali e futuri difetti. Tale tipo ti approccio è considerato lineare, sequenziale e facilmente monitorabile e controllabile.

Questa metodologia, tuttavia, si contraddistingue per una limitata flessibilità: risulta particolarmente complesso, infatti, far fronte a rettifiche che susseguono la pianificazione iniziale. Altro tema da non sottovalutare è la prassi per cui i deliverables vengono visionati dal cliente solo al loro completamento, amplificando così, le conseguenze di un possibile errore riscontrato in fase finale. Queste proprietà rendono il metodo Waterfall ideale per progetti durevoli e con una bassa incertezza tecnologica.

2.3.2 La metodologia Agile

L'insoddisfazione da parte di alcuni esperti del settore nei confronti dei vecchi processi, considerati eccessivamente costosi, lenti ed incapaci di far conseguire i risultati desiderati, spinge, nel 2001, un gruppo di sviluppatori a stilare il manifesto per lo sviluppo Agile. Con questa metodologia rivoluzionaria si sono stravolte le pratiche consolidate del vecchio Project Management, ritenute oramai eccessivamente strutturate, segmentate e burocratiche, introducendo così una visione orientata alla gestione del cambiamento e al soddisfacimento del cliente.

I punti chiave su cui si basa tale cambiamento sono i seguenti:

- Considerare gli individui e le interazioni piuttosto che i processi e gli strumenti;
- Porre il focus sul software funzionante piuttosto che sulla documentazione esaustiva;
- Incentrarsi sulla collaborazione con il cliente piuttosto che sulla negoziazione di contratti;
- Rispondere al cambiamento piuttosto che seguire un piano ben delineato anticipatamente.

All'interno del manifesto si presentano le principali caratteristiche della metodologia Agile, tra queste la *Scrum12* è la più diffusa, in particolare nell'ambito software.

I principi cardine su cui poggia Scrum sono la premura che i bisogni del cliente vengano soddisfatti e la disponibilità ad accogliere il cambiamento. Per il rispetto del primo concetto è fondamentale il ruolo del Product Owner che deve ben interpretare le esigenze e le priorità degli stakeholders e del proprio team. Per quanto concerne il trattamento del cambiamento, è reso possibile ed efficace grazie al continuo confronto con il committente e allo sviluppo del software in maniera frammentata ed incrementale. Questa assidua cooperazione consente che i requisiti arrivino regolarmente e il team sia in grado di adattarsi alle varie circostanze [4].

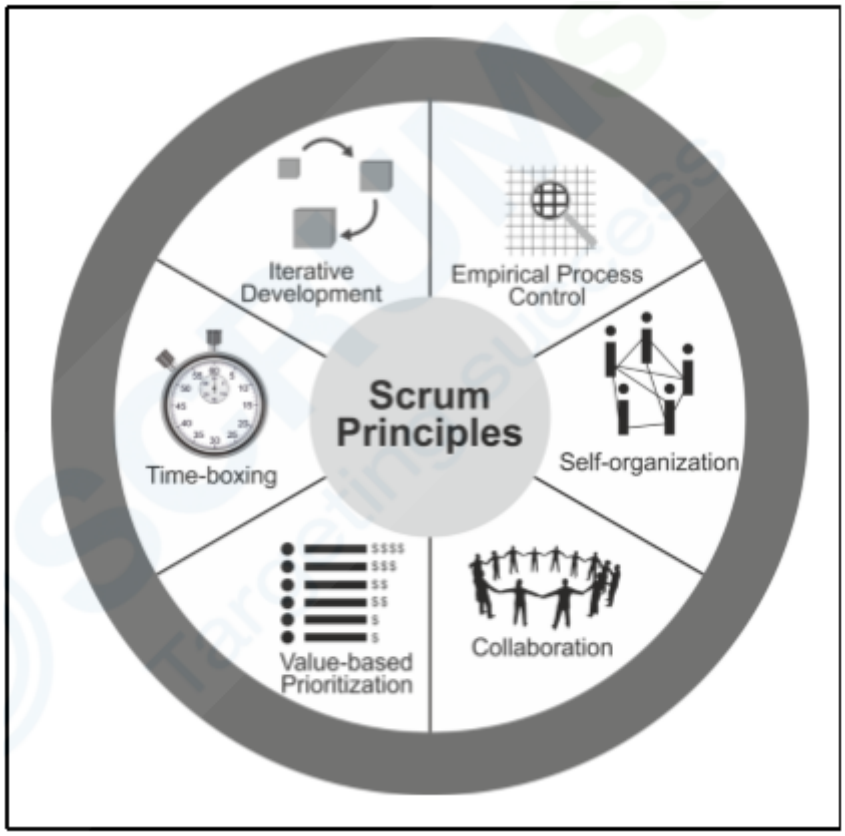

Figura 2.6. I principi Scrum (fonte [4])

Rimanendo fedeli all'idea secondo cui è fondamentale la soddisfazione del cliente attraverso la consegna frequente di piccole ed utili parti di codice, Scrum si struttura con cosiddetti *Sprint*, processi di tipo incrementale ed iterativo in cui viene suddiviso il progetto, al termine dei quali vengono consegnate piccole parti funzionanti del software; a differenza del modello a cascata, quindi, la consegna avviene in maniera continua ad intervalli regolari.

La prima fase del progetto è l'individuazione della visione globale che viene identificata a seguito del primo team meeting, in cui si definiscono i requisiti e si stila un'iniziale pianificazione del programma. In questo step si determina la figura guida del framework, lo Scrum Master, si indicano gli stakeholders, si crea lo Scrum Team e si descrivono le User Story. Queste ultime, conosciute anche come *Epic*, rappresentano la lista delle attività che devono essere portate a termine durante ciascuno Sprint; è necessario, pertanto, che vengano stimate, ordinate ed approvate per costituire il cosiddetto *Prioritized Product Backlog* del progetto. Il Product Backlog assieme allo Sprint Backlog e all'Increment sono definiti *artefatti*, ovvero le modalità con cui Scrum visualizza il lavoro e il valore raggiunto fino a qual momento. Ogni artefatto include un commitment, con il fine di migliorare la trasparenza e attraverso il quale è possibile misurare i progressi del team.

Terminata la fase di programmazione, si dà avvio all'implementazione dei deliverables dello Sprint, che vengono monitorati quotidianamente con lo strumento degli stand up meeting, al fine di verificare giornalmente lo stato di avanzamento dei lavori e far emergere eventuali criticità. Ogni Sprint si conclude con uno Sprint Review Meeting nel quale si richiede agli stakeholders l'accettazione del lavoro prodotto fino a quel momento. Nel caso in cui questo avvenga, si procede con la consegna al cliente dei deliverables, con la formalizzazione della conclusione dello Sprint mediante apposita documentazione e con una attenta riflessione su quanto appreso.

Nel mondo degli affari di oggi, le esigenze aziendali e i requisiti in continua evoluzione rappresentano una sfida per i progetti e la loro gestione in termini di portata, costi e tempi. I processi moderni, inoltre, sono sempre più complessi e interconnessi, intrinsecamente correlati a diverse tipologie di clienti, partnership e concorrenti. Queste sfide sottolineano la necessità di un approccio flessibile ed adattabile per poter produrre progetti, prodotti e servizi in modalità sempre più celere, mantenendo alti gli standard di qualità e rispettando il pieno soddisfacimento delle esigenze del client [7]: l'Agile risulta essere, dunque, la risposta migliore a tali richieste.

[7] Macheridis, 2009; Shenhar, 2004; Weinstein, 2009

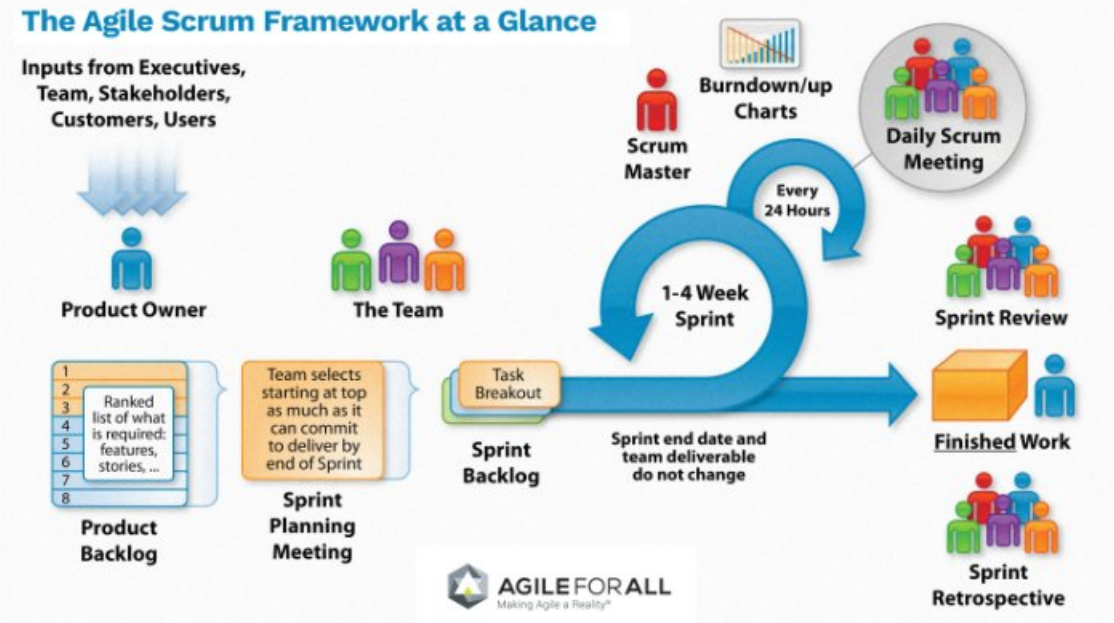

Figura 2.7. Metodologia Scrum

2.3.3 Il processo Hybrid

In ambito informatico, il termine ibrido o Hybrid è utilizzato per indicare un compromesso raggiunto tramite la combinazione di metodologie eterogenee, con l'obiettivo di soddisfare bisogni contrastanti di un processo di produzione del software in specifici contesti e situazioni. In particolare, il processo Hybrid, più frequentemente, indica l'infiltrazione di metodi Agile in processi di tipo tradizionale come il Waterfall.

Con l'espansione della tecnologia ed il diffondersi della necessità della produzione di software iniziano a sorgere i primi dubbi sulla validità dei processi utilizzati per la gestione dei progetti fino a quel momento in uso. La pianificazione di progetti software era infatti continuamente ancorata ai principi tradizionali. Come descritto in precedenza, il manifesto Agile segna la prima vera discontinuità importante, modificando l'approccio culturale al processo di produzione del software; sposta il focus dalla burocrazia del processo alla rapidità di azione, dal processo al prodotto/cliente, dalla forma alla sostanza, mantenendo la concentrazione su qualità, costo, valore delle risorse. Tuttavia il metodo Agile non rappresenta la soluzione a tutti i problemi: esistono situazioni, per esempio, in cui il paradigma Agile non può essere adottato per tutto il ciclo di sviluppo, ma solo per la sua parte implementativa. In tempi recenti si è quindi fatta avanti

l'idea di ibridare il processo di produzione Waterfall con le pratiche Agile, con l'obiettivo dichiarato di far coesistere i due mondi, cercando di ottenere da ciascuno dei due il maggior vantaggio possibile e limitandone al minimo gli svantaggi. Questo metodo è stato definito *"Wagile"*; sono stati proposti diversi approcci al processo ibrido, nominalmente diversi ma nella sostanza piuttosto simili.

Vale la pena di osservare che spesso la necessità di adottare una modalità Hybrid è dovuta all'incertezza sulla completezza dei requisiti quando si opera con un approccio Waterfall. La difficoltà di avere ben chiara, preliminarmente, l'ampiezza e la complessità delle richieste del cliente è indicata da tutti gli analisti come la causa prima del fallimento o della forte criticità nei progetti IT.

Come con tutti i modelli ibridi, entrambe le parti devono scendere a compromessi. Lo sviluppo a cascata deve rinunciare ad alcune certezze di aspettative fisse, in favore della flessibilità. Lo sviluppo Agile deve accettare il compromesso di essere creativi ma con molta meno libertà, lavorando con una scadenza fissa e con una pianificazione adeguata a prevedere i costi, stimare lo sforzo e valutare i rischi.

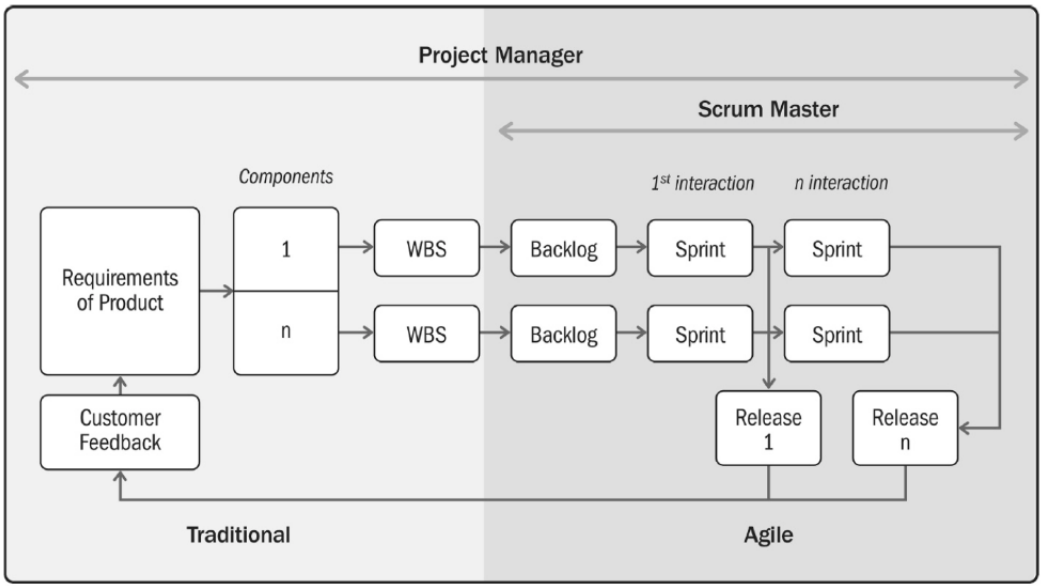

Figura 2.8. Metodo Hybrid Wagile (fonte[6])

Capitolo 3

IT Project Management

3.1 Information Technology

Con il termine Information Technology si fa riferimento alle metodologie e tecnologie necessarie per l'utilizzo, privato o pubblico, di qualsiasi elaboratore come computer, mainframe, o server, sistema di networking o di archiviazione, strumenti di telecomunicazione e altri dispositivi ed infrastrutture per proteggere, scambiare, archiviare, creare ed elaborare qualsiasi formato di dati elettronici.

A partire dagli anni Ottanta, l'Information Technology si è affermata come risorsa strategica fondamentale per i processi di business aziendale, facilitando l'adeguamento ai cambiamenti del mercato, migliorando l'assetto organizzativo e funzionale delle aziende e dando la possibilità di generare un importante vantaggio competitivo. È diventato indispensabile, quindi, per le imprese moderne disporre delle capacità di gestione e delle abilità di applicazione di questa risorsa innovativa, al fine di essere fortemente concorrenziali sul mercato dei competitors.

Ad un livello più concreto e tangibile, la tecnologia dell'informazione è definita come l'ingegnosità di applicare la tecnologia per la risoluzione di problemi aziendali ed organizzativi su larga scala. Sempre più società, infatti, desiderano implementare soluzioni intuitive e sofisticate: l'IT risulta essere l'elemento in grado di soddisfare questo bisogno. All'interno delle imprese, alcune tra le esigenze che implicano il coinvolgimento del settore IT possono essere le seguenti:

- Sovraccarico di dati: le aziende spesso devono elaborare enormi quantità di dati, questo richiede software avanzati e capacità analitiche ricercate;

- Utilizzo di sistemi wireless: sempre più datori di lavoro offrono la possibilità di lavorare in modalità smart working mettendo a disposizione smartphone, tablet o laptop con hotspot wireless e capacità di roaming;
- Servizi cloud: la maggior parte delle aziende non gestisce più le proprie "server farm" per archiviare enormi quantità di dati, preferisce, invece, lavorare con servizi cloud, piattaforme di hosting di terze parti che conservano tali dati;
- Larghezza di banda per la riproduzione di video: le soluzioni di videoconferenza sono diventate sempre più popolari, è necessaria, quindi, una maggiore larghezza di banda di rete per supportarle a sufficienza.

Il dipartimento IT ideale è dipinto, pertanto, come la divisione in cui si è in grado di automatizzare e creare processi per ogni attività quotidiana, e vi è un allineamento trasparente con gli obiettivi dell'azienda; i tre pilastri principali alla base di un reparto IT sono i seguenti [7]:

- IT Governance: è la combinazione di politiche e processi che assicurano l'efficacia della gestione dei sistemi IT rimanendo in linea con l'esigenza dell'organizzazione;
- IT Operations: questa categoria include il supporto tecnico, la manutenzione della rete, i test di sicurezza e le attività di gestione dei dispositivi;
- Hardware ed Infrastrutture: tale area di interesse è correlata a tutti i componenti fisici dell'infrastruttura IT; include la configurazione e la manutenzione di apparecchiature come server, router, o laptop.

Forte della crescente diffusione del web, oggi, il settore IT è diventato sempre più fondamentale in ogni ambito, accorciando le distanze tra le persone, creando nuove opportunità di crescita e nuove professioni, migliorando l'efficienza e la produttività in quasi tutti i settori. È in questo contesto in continua evoluzione e sempre più intrinsecamente dipendente dalla tecnologia che si collocano le aziende fornitrici di servizi IT, diventate famigerate al giorno d'oggi non solo per gli aspetti tecnologici di cui si occupano, ma anche per la stretta ed attenta relazione con il cliente che sono in grado di instaurare.

3.2 IT Project Management

La gestione dei progetti IT è il processo di gestione del piano, dell'organizzazione e della responsabilità per il raggiungimento degli obiettivi dell'information technology.

L'entità della gestione di questi progetti spesso non si traduce nella mera applicazione delle conoscenze, dell'allineamento delle competenze e dell'utilizzo di strumenti e tecniche regolari per portare a termine un progetto. I Project Manager IT, infatti, affrontano le sfide delle integrazioni interdipendenti, dei rapidi aggiornamenti tecnologici e delle modifiche alle versioni che possono verificarsi durante la sequenza temporale del progetto.

Nell'ambito dell'information technology, i progetti si rivelano unici nel loro genere, caratterizzati da emergenza, atipicità, breve termine ed incertezza. In questo contesto il ruolo del Project Manager deve fornire una reattività soddisfacente al cliente, riuscendo, al tempo stesso, a ottenere il supporto della direzione, senza mai sospendere la guida e il controllo del team, e la gestione dei rapporti con le parti terze.

Esistono diversi modelli di sviluppo di progetti IT, e quindi nello specifico di sviluppo software, che si adattano alle differenti e svariate situazioni e vincoli del contesto. Tuttavia, sulla base del modo in cui si susseguono i task e si realizzano i deliverables, possono essere definite diverse tipologie:

- Modello predittivo: questa è la metodologia più comune e tradizionale. L'ambito, la pianificazione e le stime vengono elaborate prima dell'avvio del progetto, ed ogni fase del suo sviluppo deve essere iniziata e completata nell'ordine previsto. Un'estensione del modello a cascata è rappresentato dal V-Model, affermatosi con la diffusione dei progetti di sviluppo software. Questo metodo, contrariamente all'andamento discendente lungo una linea retta della tipologia Waterfall, risale con una tipica forma a V da cui ne deriva il nome. Tale metodica mette in evidenza la relazione tra ogni fase del ciclo di vita dello sviluppo del software e la sua fase di testing o collaudo. Durante l'implementazione di ogni attività vengono progettati e descritti anche i test necessari per la verifica di quanto realizzato; questa operazione consente di individuare eventuali errori o anomalie prima della fase successiva,

evitandone la propagazione. Gli stessi test verranno poi adoperati nelle successive fasi di testing descritte nella parte destra della figura 3.1;

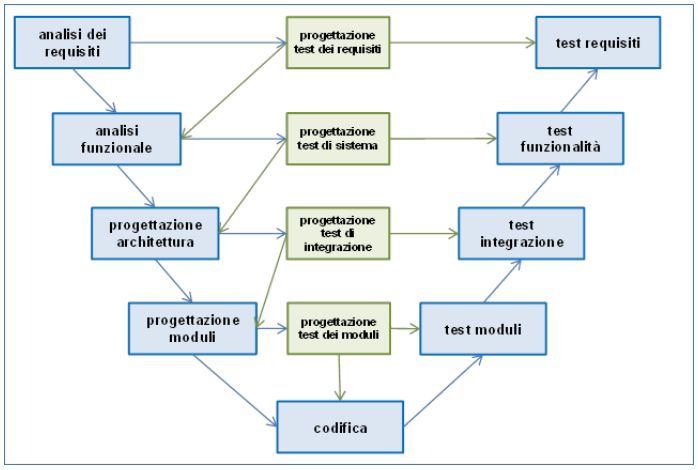

Figura 3.1. V-Model

- Modello evolutivo: questo metodo tenta di superare i maggiori limiti del modello a cascata, basandosi sulla prototipazione di soluzioni rapide e semplificate. La prima versione del software sviluppato è considerata un throw-away, una prova cestinabile ma utile a fornire al progettista un feedback di analisi e verifica; sarà la seconda versione, sviluppata con il metodo a cascata, a consentire la realizzazione del prodotto finito. Questa metodica elimina i potenziali errori della fase di analisi ma non riduce i tempi di sviluppo dell'intero ciclo;

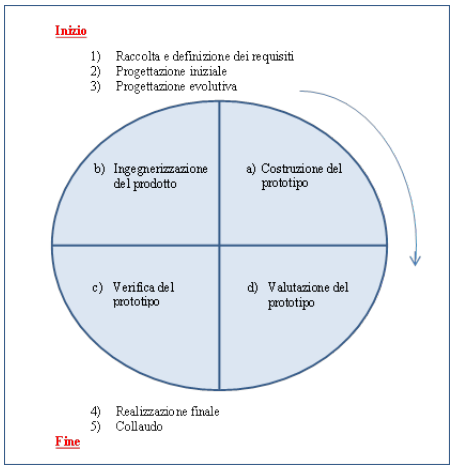

Figura 3.2. Modello Evolutivo

- Modello incrementale: deriva dalla trasformazione del modello evolutivo. Questo metodo prevede che si proceda con la realizzazione di un archetipo parziale che possa essere valutato dal cliente da cui si ottiene un primo riscontro; si prosegue in maniera incrementale rilasciando gradatamente prototipi successivi fino al conseguimento del prodotto desiderato. Con questo modello si riducono le tempistiche, ma aumenta il rischio di perdere il controllo delle attività: può accadere, infatti, che l'avvio della progettazione del deliverable successivo avvenga prima della valutazione della versione incrementata precedentemente; per questo motivo continua ad essere indispensabile una contaminazione del modello a cascata;

- Modello a spirale: rappresenta l'evoluzione del metodo di sviluppo classico; adottando tale metodologia le fasi del ciclo di vita tradizionale vengono percorse fino ad ottenere una prima versione estremamente semplificata del programma; questa viene utilizzata per comprendere al meglio i requisiti richiesti dagli utenti, le specifiche e la progettazione, portando alla realizzazione di una nuova versione più evoluta. Il processo si interrompe quando si raggiunge una versione soddisfacente, solitamente, come si evince dalla figura 3.3, conseguita con non più di tre cicli.

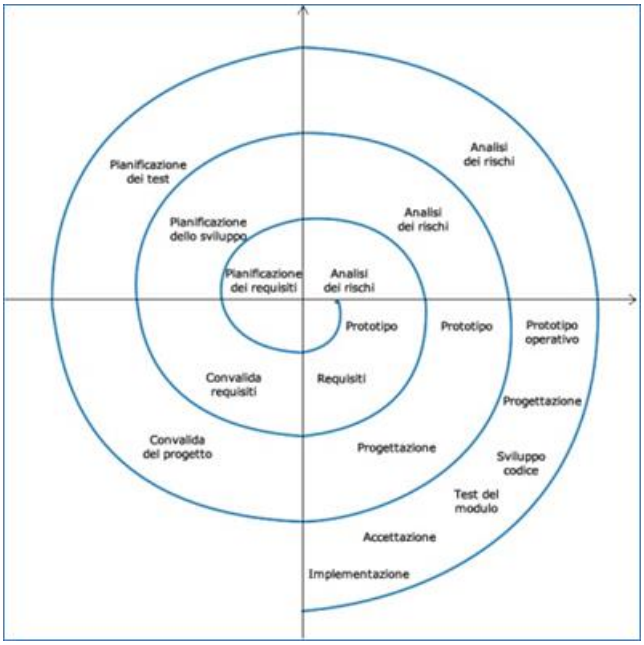

Figura 3.3. Modello a spirale

Indipendentemente dalla metodologia di gestione scelta, rimane ben chiara l'esigenza comune del saper gestire e correlare l'attività propria del progetto, con le relazioni con il cliente, con la direzione e con il team di sviluppo; tutti elementi che interagiscono fortemente nell'intero sistema progettuale e che, se ben coordinati, possono condurre al successo del progetto [8].

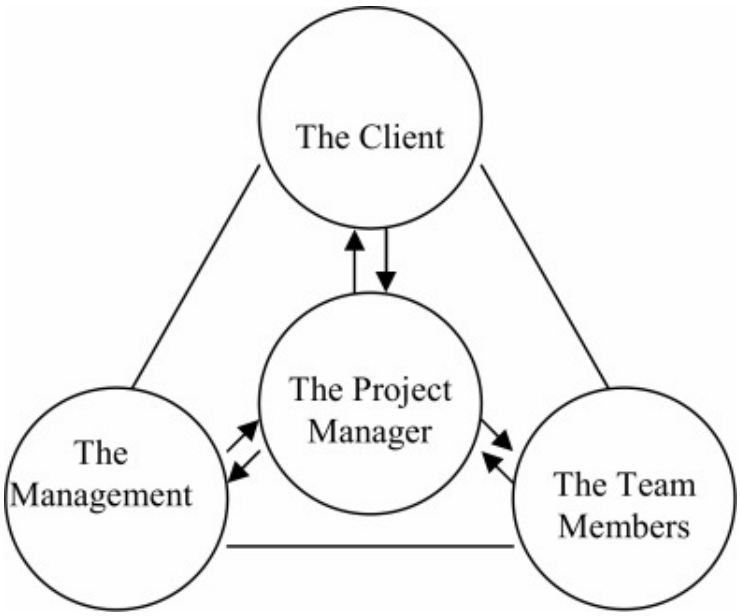

Figura 3.4. Relazioni punti chiave IT Project Management (fonte [8])

3.3 Il ciclo di vita del software

Con l'espressione "ciclo di vita del software" si intendono le fasi percorse da un progetto software dal suo avvio fino alla realizzazione e conseguente manutenzione. Nella figura 3.5 sotto riportata vengono raffigurati i principali step che compongono questo ciclo.

Il progetto ha inizio con la raccolta dei requisiti che viene spesso definita come "requirement elicitation", intendendo lo stadio basato sulla capacità di estrarre dal cliente quante più informazioni possibili. Questo processo può avvenire con la combinazione di svariati metodi: osservazioni sul campo, interviste, gruppi di lavoro. Una volta individuate le caratteristiche desiderate, è indispensabile effettuarne un'analisi accurata per ottenere una descrizione completa e formalizzata; l'obiettivo è quello di ottenere un livello di dettaglio chiaro ed adeguato sui requisiti funzionali, non funzionali, e sui vincoli che il progetto dovrà rispettare.

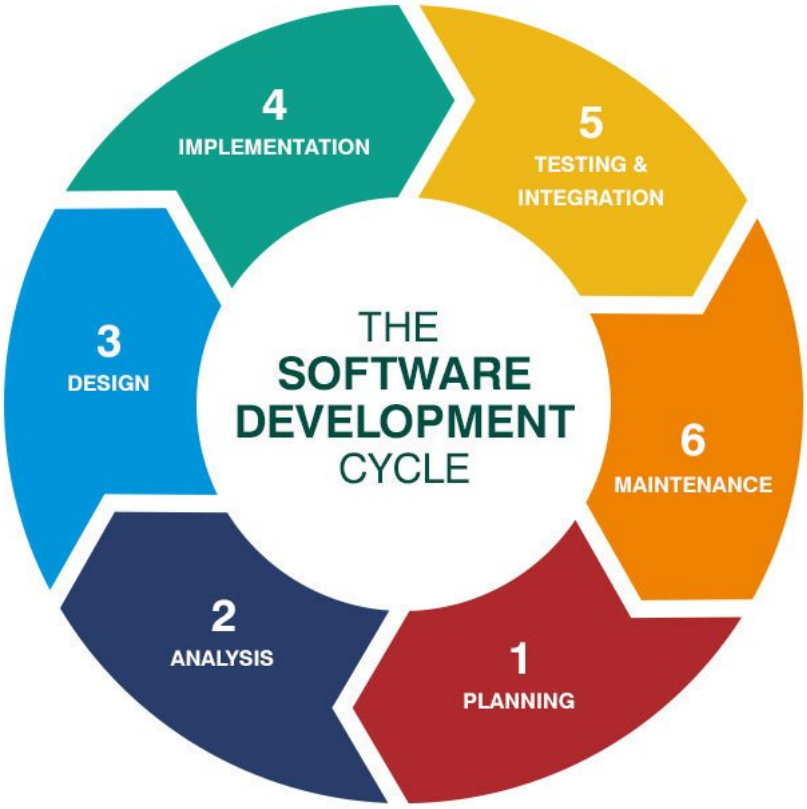

Figura 3.5. Le fasi del ciclo di vita del software

Dall'output della fase di analisi ne deriva la progettazione minuziosa delle istruzioni operative sull'elaborazione del prodotto. Nel caso specifico di uno sviluppo software, la progettazione di un'applicazione è la ricerca della soluzione implementativa migliore, conformemente agli obiettivi e ai vincoli prestabiliti. La progettazione, differentemente dall'analisi che si sofferma ad un livello più astratto, non può prescindere dalla tecnologia utilizzata: progettisti e team di sviluppo definiscono l'architettura del sistema, ovvero i componenti software, l'interfaccia e le relazioni tra le parti. Gli strumenti di uso comune in questa fase sono il diagramma di flusso e la sua evoluzione nel UML activity diagram: entrambi in grado di fornire una scomposizione delle attività per poterle implementare con gli opportuni linguaggi di programmazione selezionati.

Se queste prime due fasi possono ritenersi parzialmente analoghe ad ogni progetto di realizzazione di qualsivoglia prodotto, è a partire dalla fase di implementazione che si concretizzano le reali differenze, momento in cui i prodotti si instradano in percorsi ad hoc muovendosi eterogeneamente.

La fase di coding consiste nello sviluppo e codifica del codice sorgente, aspetto sostanziale per ogni progetto informatico. I programmatori, con l'utilizzo di database e linguaggi di programmazione selezionati, realizzano la soluzione concreta del problema nel cosiddetto ambiente di sviluppo. Generalmente, il software viene suddiviso in parti, i moduli, che vengono implementate separatamente per poi essere assemblate ed integrate totalmente una volta giunti al completamento dello sviluppo. Questa prima versione del software viene definita versione *alfa*, un modello ancora acerbo rispetto al prodotto finale; da questo prototipo deriva, infatti, la seconda versione generata a seguito di un primo debugging, ovvero a seguito della rimozione dei principali ed evidenti errori.

La fase di test, o collaudo, viene eseguita da attori terzi, definiti tester, nell'infrastruttura di supporto indicata come ambiente di testing. Questo step verifica e valida la capacità del software di attenersi e soddisfare i requisiti predeterminati dal cliente, analizzando quindi la correttezza del programma dal punto di vista funzionale. Anche il collaudo può essere effettuato sui singoli moduli o sul sistema integrato nella sua totalità; può essere un collaudo funzionale, di rottura, di regressione, di sicurezza o ancora di accessibilità, di accettazione. Gli errori individuati in questa fase si rivelano abitualmente molto più insidiosi rispetto a quelli evidenziati in fase di debugging; nel caso in cui l'applicativo non rispetti le specifiche desiderate, verrà inoltrato al team di sviluppo che si occuperà della sua correzione. Occorre effettuare una distinzione tra i problemi riscontrati, differenziando tra malfunzionamenti e difetti. Il malfunzionamento si verifica quando, sotto determinate condizioni, il comportamento del software si discosta da quello atteso; il difetto, invece, si riscontra nel caso in cui una porzione di codice eseguita con specifici input, genera un malfunzionamento. Generalizzando, vi è quindi un malfunzionamento ogni qual volta si esegue un codice che contiene un difetto nei dati in input, tale da generare un errore visibile. La fase del collaudo tenta, pertanto, di individuare il maggior numero di malfunzionamenti possibile per poter risalire e sanare i difetti. Nel processo di realizzazione del software si procede, in un primo momento, attenendosi ad una check list di test-case da eseguire manualmente o automaticamente, e in un secondo momento, alla riproduzione di alcuni scenari di collaudo in cui si ripercorrono i passaggi che svolgerebbe verosimilmente un utente. In questo step si ricercano quindi le anomalie che possono far discostare il comportamento dell'applicativo da quelle che sono le richieste del cliente, verificando, pertanto, il lavoro da un punto di

vista prevalentemente funzionale e prestazionale.

Raggiunto il livello di qualità ambito, può avvenire il rilascio, o deployment, del programma che consiste nell'installazione del prodotto software nell'infrastruttura eseguibile ed usufruibile dagli utenti. Nel caso di sviluppo software l'entrata in ambiente di produzione rappresenta il collocamento del programma su uno o più server web, a partire dal quale ha inizio la sua vita operativa nel contesto per cui è stato progettato;

Per concludere il ciclo di vita si giunge all'ultima fase, quella della manutenzione, che comprende tutte le attività che comportano modifiche da apportare al prodotto in tempi successivi al rilascio. Si può classificare in ordinaria ed evolutiva: la prima tipologia rappresenta l'insieme di interventi correttivi necessari ad eliminare errori nel funzionamento del programma che emergono nel tempo; la seconda, invece, comprende la totalità delle variazioni volte ad arricchire e migliorare le funzionalità del programma per via di nuove necessità operative o dell'adeguamento a nuove innovazioni tecnologiche. Ogni modifica di questo tipo comporta inevitabilmente nuovi costi legati alle ulteriori fasi di sviluppo, collaudo di regressione, e rilascio in produzione.

3.4 La manutenzione

Con il riconoscimento della longevità dei sistemi di informazione, nasce la necessità di comprendere le trasformazioni longitudinali che tali sistemi sperimentano. Poiché i sistemi applicativi sono modelli che vivono in ambienti dinamici, la loro utilità può essere mantenuta solo se questi sistemi vengono modificati quando cambiano i rispettivi ambienti. Indipendentemente dal fatto che queste modifiche vengano apportate per correggere difetti nel codice esistente, per adattarsi all'ambiente o per aggiungere funzionalità, tali attività vengono collettivamente denominate attività di manutenzione del software. Belady e Lehman [9] definiscono l'evoluzione del software come "..il comportamento dinamico dei sistemi di programmazione man mano che vengono mantenuti e migliorati nel corso della loro vita".

La manutenzione e l'evoluzione del software sono un'area notevolmente poco studiata ed approfondita; gli esperti del settore, infatti, stimano che un'elevatissima percentuale di tutti i costi relativi a un

software relativamente moderno siano costi di manutenzione regolare, di cui la maggior parte delle aziende non tiene conto. Questo aspetto sconvolge, non di rado, il budget, inficiando la realizzazione complessiva del progetto.

Le moderne soluzioni software sono spesso componenti tecnologici molto complessi che richiedono aggiornamenti e manutenzione regolari. Questi lavori di manutenzione possono spesso essere dispendiosi in termini di tempo e devono essere pianificati con largo anticipo. Il software customizzato è ancora più costoso da manutenere in quanto richiede un team di supporto dedicato sempre disponibile. Inoltre, un'efficace manutenzione del software richiede un'adeguata documentazione, per consentire agli sviluppatori un'agevolazione nell'individuazione e conseguente risoluzione dei problemi.

L'intero processo di manutenzione del software coinvolge vari gradi di complessità, requisiti e competenze professionali, inevitabilmente implicanti costi differenti. A seconda delle esigenze dell'azienda e delle condizioni di mercato esistenti, un numero variabile di servizi può essere incluso in un pacchetto di manutenzione standard. Tendenzialmente, questi servizi possono essere classificati in tre tipologie:

- Correttivi
- Adattivi
- Migliorativi

Le procedure di manutenzione correttiva del software coinvolgono tutto il lavoro svolto per risolvere i problemi che potrebbero verificarsi nel programma; possono essere disfunzioni che si manifestano durante la fase di test iniziale o in seguito al rilascio in produzione (nel qual caso i programmi di risoluzione dei problemi devono essere rilasciati in pacchetti di aggiornamento inviati agli utenti). Questi, di solito, rappresentano circa il 20% del totale dei costi dedicati alla manutenzione del software [1].

Le procedure di manutenzione adattiva coinvolgono tutti gli aggiornamenti e le modifiche apportate al programma per adeguarlo agli attuali standard del settore. Indipendentemente dal suo grado di avanguardia, è fondamentale che la tecnologia del software subisca

[1] https://galorath.com/software-maintenance-costs

continui rinnovamenti per stare al passo con le innovazioni del suo campo. I consumatori, solitamente, optano per le opzioni più convenienti e complete, che rendono la manutenzione adattiva una parte vitale del mantenimento tecnologico, costituendo circa il 25% di tutti i costi di manutenzione del software [2].

Gli interventi migliorativi sono condizionati dalle esigenze dell'azienda; sono procedure abitualmente eseguite per includere o rimuovere eventuali funzioni del programma sulla base della discrezione del consiglio aziendale e del suggerimento dei consumatori. Il miglioramento regolare del sistema può garantire la massima soddisfazione dell'utente, gravando per circa il 35% dei costi dedicati al sistema manutentivo [3].

L'articolo [10] sostiene che l'onere correlato alla manutenzione del software sia aumentato nel corso degli anni, sino a raggiungere, come mostrato in figura 3.6, il 90% dello sforzo totale speso su un sistema durante la sua evoluzione.

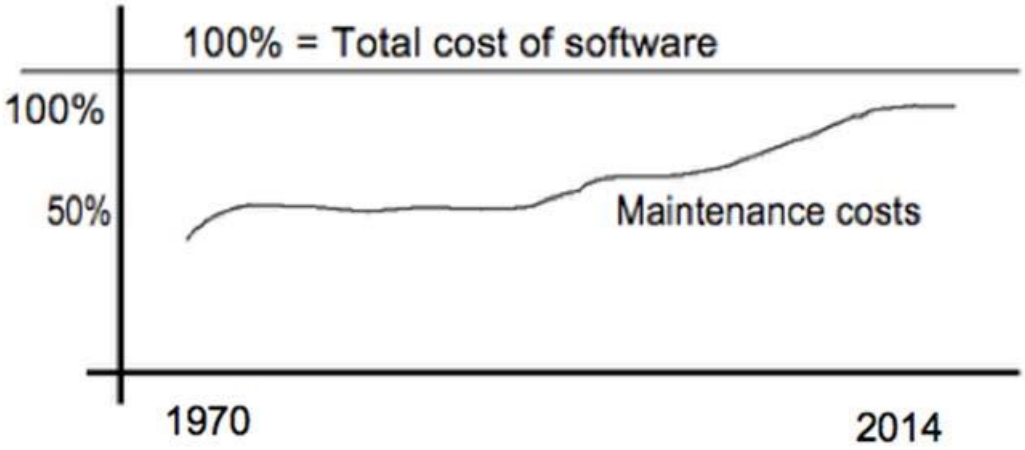

Figura 3.6. Evoluzione dei costi legati alla manutenzione

La ricerca sui costi di manutenzione del software ha molto in comune con la ricerca sui costi di sviluppo di un nuovo applicativo, poiché entrambe implicano la creazione di codice funzionante attraverso gli sforzi di sviluppatori dotati di esperienza, strumenti e tecniche appropriati. La manutenzione del software, tuttavia, è fondamentalmente

[2] https://galorath.com/software-maintenance-costs

[3] https://galorath.com/software-maintenance-costs/

diversa dallo sviluppo di nuovi sistemi in quanto deve interagire con un sistema esistente.

È dimostrato che i costi di manutenzione del software siano influenzati dalla complessità del programma in termini di dimensione del modulo, dimensione della procedura e complessità della ramificazione. Sebbene il legame tra la difficoltà nella comprensione del software e il costo della sua manutenzione sia interessante, le prove empiriche precedenti, che collegano la complessità del software ai costi di manutenzione del software, sono relativamente deboli [11], poiché molti di tali tentativi si basano su esperimenti che coinvolgono piccole porzioni di codice. Tali prove sono preziose, ma diversi ricercatori hanno notato che tali risultati devono essere applicati con cautela ai sistemi applicativi commerciali su larga scala, valutando singolarmente le peculiarità di ogni specifica casistica [12].

Lo studio analizzato nell'articolo [13] sembra giungere alla conclusione secondo cui testando l'impatto delle dimensioni del modulo del codice sullo sforzo del programmatore, risulti una relazione inversamente proporzionale tra le due variabili; l'analisi si conclude, tuttavia, con la tesi secondo cui non sia possibile affermare nulla di definitivo su tale legame, a causa dell'ottenimento di risultati scarsamente saldi e non universalmente validi.

Floris e Harald, nello studio da loro condotto, introducono, tra i fattori cruciali nell'aumento del costo manutentivo, la mancanza di documentazione completa a supporto della spiegazione del codice implementato precedentemente. Ipotizzano che le conseguenze di tale incompletezza possano generare ritardi nella comprensione del programma e successiva lentezza nella risoluzione delle anomalie riscontrate: ".. Perché i membri del team di produzione potrebbero aver lasciato l'azienda, essersi ritirati o essere sostituiti da un'altra persona che non è a conoscenza del processo produttivo" [14].

Più recentemente, Gibson e Senn hanno analizzato l'impatto della struttura del software utilizzando un esperimento di laboratorio [15]. Hanno evidenziato come le versioni più strutturate della stessa porzione di codice richiedano, in media, meno tempo per essere manutenute, e che le valutazioni soggettive dei programmatori sulla complessità dei sistemi esistenti non siano molto accurate; un risultato, questo, che attribuiscono all'incapacità dei manutentori di separare la complessità delle nuove attività dalla quella dei sistemi esistenti. Tuttavia, i risultati ottenuti dai loro esperimenti non si sono rivelati globalmente validi.

Secondo quanto riportato nell'articolo di C. Ebert e H. Soubra [17], per fornire in modo più affidabile e gestire meglio i propri progetti, le aziende sono interessate a misurare e stimare lo sforzo di manutenzione per le consegne di modifiche software. La maggior parte delle aziende utilizza analogie o ipotesi, ma presto si rende conto che l'accuratezza è insufficiente e non c'è ripetibilità. La misurazione delle dimensioni funzionali viene utilizzata nell'industria come base su cui costruire stime. Si utilizzano come input le richieste di modifica e la loro analisi nei confronti del sistema in cui la rettifica deve essere incorporata; sulla base di questa analisi iniziale degli impatti del cambiamento, un modello aiuta a ricavare lo sforzo richiesto per implementare i cambiamenti sottostanti. Fattori come il tipo di software, la distribuzione del team o la ripetibilità del progetto aiutano nella regolazione della stima dello sforzo.

L'analisi dell'evoluzione del software è lo studio del cambiamento longitudinale. Più a lungo un sistema software rimane produttivo, più il cambiamento diventa inevitabile. Questo cambiamento non è né necessariamente buono né cattivo, ma sicuramente indispensabile. Utilizzando la volatilità del software e i modelli di manutenzione del ciclo di vita per quantificare l'evoluzione del programma, è possibile comprendere più a fondo quale sia la causa del cambiamento nei sistemi informativi, controllarne i fattori principali su cui è fattibile effettuare interventi, e anticipare le conseguenze degli aspetti su cui risulta impossibile intervenire anticipatamente. Con una migliore comprensione del cambiamento longitudinale nei sistemi software, è possibile esaminare le domande su come e quando prepararsi al cambiamento, quali tipi di sistemi sono maggiormente sottoposti a modifiche, e come questi mutamenti possono essere gestiti nel modo più efficace ed efficiente possibile. La previsione dei costi del sistema dell'intero ciclo di vita può essere inclusa nelle decisioni di gestione del software.

Sebbene i ricercatori riconoscano l'importanza delle attività di manutenzione del ciclo di vita e dei loro risultati, sono state condotte relativamente poche ricerche empiriche sulla fase successiva al rilascio in produzione [18]; nessuno dei lavori sviluppati che compongono il know how attuale, difatti, realizza stime che possano essere utilizzate dai responsabili della manutenzione del software per impiegare al meglio le proprie risorse.

È proprio da questo gap informativo che scaturisce l'idea di approfondire un tema di grande attualità; è evidente che, attualmente, non

si sia giunti all'elaborazione di teorie e allo sviluppo di metodologie, concretamente servibili, e che possano essere globalmente condivise e corroborate per quanto concerne la fase finale dell'intero processo del software. Questa tesi, pertanto, si pone l'obiettivo di studiare la previsione delle spese legate alla fase di manutenzione per poterle includere nella stima iniziale e renderla, così, più dettagliata e fedele a quelli che saranno i costi da sostenere per l'intero ciclo di vita del progetto. Nei capitoli successivi si andrà, quindi, a fornire una panoramica delle dinamiche aziendali in cui è nata questa idea, confrontando le cosiddette best practices, prettamente teoriche, appena descritte, con il processo reale e caratteristico della società in esame; successivamente si canalizzerà l'indagine focalizzandosi in particolar modo su una determinata parte della fase di manutenzione: quella immediatamente successiva alla fase di rilascio, definita, nell'azienda in questione, fase di *follow up*.

Capitolo 4

Il Machine Learning

Il machine learning è un componente importante del crescente campo della scienza dei dati. Attraverso l'uso di metodi statistici, gli algoritmi vengono addestrati per effettuare classificazioni o previsioni, scoprendo informazioni chiave all'interno dei progetti di data mining. Queste informazioni guidano, successivamente, il processo decisionale all'interno delle applicazioni e delle aziende, incidendo idealmente sulle metriche di crescita chiave.

Definire in maniera semplice le caratteristiche e le applicazioni del machine learning non è sempre possibile: questo settore è molto vasto e prevede differenti modalità, tecniche e strumenti per essere realizzato. Le diverse metodologie di apprendimento e di sviluppo degli algoritmi, inoltre, danno vita ad altrettante possibilità di utilizzo che allargano il campo di applicazione dell'apprendimento automatico rendendone difficile una definizione specifica. Si può, tuttavia, definire con il termine machine learning l'insieme degli svariati meccanismi che permettono ad una macchina intelligente di migliorare le proprie capacità e prestazioni nel tempo. Il dispositivo, quindi, è in grado di imparare a svolgere determinati compiti migliorando, tramite l'esperienza, le proprie capacità, le proprie risposte e funzioni. Alla base dell'apprendimento automatico vi sono algoritmi che, basandosi su nozioni primitive, rendono possibile la scelta consapevole di una decisione piuttosto che un'altra e la capacità di ripetere azioni apprese nel tempo.

Secondo quanto riportato nell'articolo [1] il sistema di apprendimento

[1] https://ischoolonline.berkeley.edu/blog/what-is-machine-learning/

di un algoritmo di machine learning è costituito da tre componenti principali:

- Un processo decisionale: in generale, gli algoritmi di apprendimento automatico vengono utilizzati per fare una previsione o una classificazione. Sulla base di alcuni dati di input, che possono essere etichettati o non etichettati, l'algoritmo produrrà una stima di un modello nei dati;
- Una funzione di errore che serve a valutare la previsione del modello. Se sono presenti esempi noti, una funzione di errore può effettuare un confronto per valutare l'accuratezza del modello;
- Un processo di ottimizzazione del modello: se il modello può adattarsi meglio ai punti dati nel set di addestramento, i pesi vengono regolati per ridurre la discrepanza tra l'esempio noto e la stima del modello. L'algoritmo ripeterà questo processo di valutazione e ottimizzazione, aggiornando i pesi in modo autonomo fino al raggiungimento di una soglia di accuratezza.

A seconda del tipo di algoritmo implementato, cioè in base alle modalità con cui la macchina impara ed accumula dati ed informazioni, si possono individuare differenti sistemi di apprendimento automatico: supervisionato, non supervisionato, e per rinforzo:

- Apprendimento supervisionato, noto anche come apprendimento automatico supervisionato, è una sottocategoria del machine learning. È definito dall'uso di set di dati etichettati per addestrare algoritmi in grado di classificare i dati o prevedere i risultati in modo accurato. La centralità di questa tipologia risiede nella capacità di costruire un database di informazioni e di esperienze. Il dispositivo, trovandosi di fronte ad un problema, fa ricorso alle situazioni inserite nel proprio sistema, le analizza e decide quale risposta dare sulla base, appunto, del suo bagaglio di esperienze già codificate. L'obiettivo è, quindi, quello di identificare, sulla base degli esempi di input e output che vengono forniti inizialmente, una regola generale che colleghi i dati di ingresso con quelli di uscita, in modo da poter poi riutilizzare tale regola per altri compiti simili;
- Apprendimento non supervisionato, noto anche come apprendimento automatico non supervisionato, utilizza algoritmi di machine learning per analizzare e raggruppare set di dati senza etichetta. Questo metodo sfrutta un approccio più indipendente che

prevede che le informazioni inserite all'interno della macchina non siano codificate, e non vi è, pertanto, la possibilità di trarre informazioni da esempi precedentemente resi noti. È il dispositivo stesso a schedare tutte le informazioni in possesso, organizzarle ed imparare il loro significato, il loro utilizzo e il risultato a cui esse portano nelle differenti situazioni;

- Apprendimento automatico di rinforzo è un modello di machine learning simile all'apprendimento supervisionato, ma l'algoritmo non viene addestrato utilizzando dati di esempio. Questo modello impara man mano che procede utilizzando tentativi ed errori. Rappresenta il sistema di apprendimento più complesso che prevede che la macchina sia dotata di sistemi e di strumenti in grado di migliorare il proprio apprendimento e di comprendere le caratteristiche dell'ambiente limitrofo e dinamico.

4.1 Modelli white-box e black-box

Una delle principali criticità legata all'utilizzo di algoritmi di machine learning è l'abilità di saper fornire agli stakeholders una spiegazione esaustiva riguardo ai risultati della previsione ottenuti. Per tale ragione, ogni qualvolta ci si trovi a dover scegliere il modello di apprendimento automatico da adoperare, è indispensabile valutare attentamente il trade-off tra la precisione che l'algoritmo può garantire, e l'interpretabilità dei risultati che ne possono derivare. È possibile categorizzare i modelli in due macro-tipologie principali:

- Black-box model: il cosiddetto modello a scatola nera in grado di fornire un elevato livello di accuratezza ma risulta di difficile comprensione il funzionamento interno dell'algoritmo; gli utenti, infatti, possono solo osservare la relazione input-output. Inoltre, non forniscono una stima dell'importanza di ciascuna caratteristica nelle previsioni del modello, né è facile capire come interagiscano i diversi fattori. In questa tipologia rientrano, per esempio, le reti neurali;

- White-box model: i modelli a scatola bianca sono notevolmente più semplici da interpretare, ma forniscono una capacità predittiva inferiore e non sono sempre in grado di modellare la complessità intrinseca del set di dati. In questa categoria rientrano modelli come la regressione lineare e gli alberi decisionali.

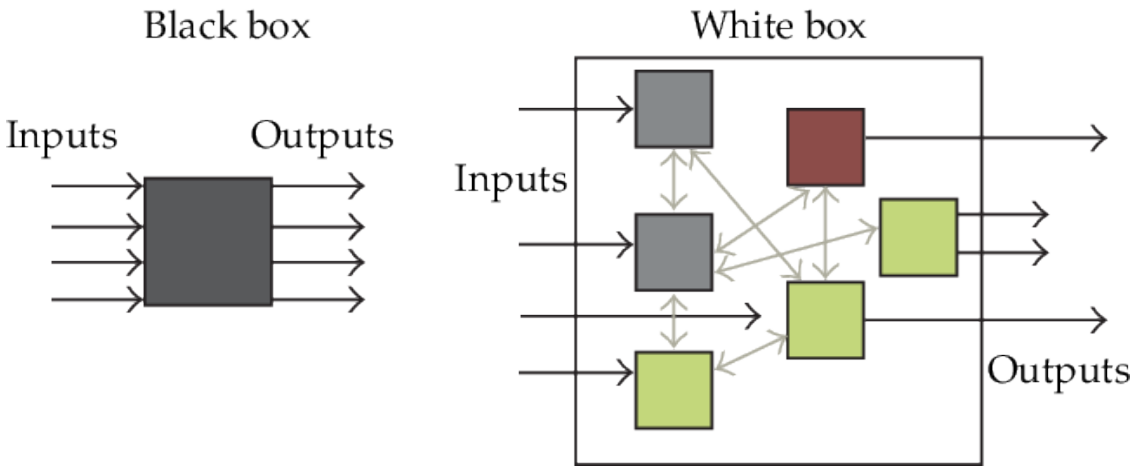

Figura 4.1. Operazioni nei modelli Black-box e White-box

Il grado di spiegabilità di un modello è spesso legato a due proprietà della funzione di risposta, la quale definisce la relazione input-output del modello: la *linearità* è la proprietà per cui, in una funzione di risposta lineare, l'associazione tra una caratteristica e il target si comporta in modo lineare. Se una funzione cambia in modo lineare, ci si aspetta che anche l'obiettivo cambi in modo lineare a una velocità simile; la *monotonia*, caratteristica che prevede che in una funzione di risposta monotona, la relazione tra una caratteristica e l'obiettivo sia sempre in una direzione di aumento o di diminuzione.

L'ambito dell'interpretabilità del modello è intrinsecamente legato alla sua complessità. I modelli lineari mostrano lo stesso comportamento nell'intero spazio delle caratteristiche e sono, quindi, interpretabili globalmente. La relazione tra l'input e l'output è spesso limitata in termini di complessità e le interpretazioni globali si basano su interpretazioni locali. Per modelli più complessi, il comportamento globale del modello è più difficile da definire e sono necessarie interpretazioni locali di piccole regioni delle funzioni di risposta. 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.

È possibile concludere che 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. Al contrario, i modelli black-box, generalmente più accurati, devono essere investigati a livello locale poichè solo in quella circostanza presentano caratteristiche di linearità e monotonia.

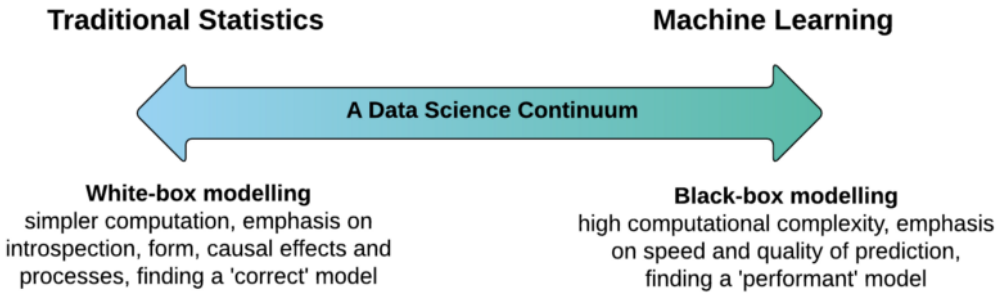

Figura 4.2.

Black-box & White-box

4.2 Reti Feed-forward

Il modello Feed-forward è la forma più semplice di rete neurale; è un algoritmo di classificazione ispirato al meccanismo utilizzato dalle reti neurali biologiche. Queste reti di modelli sono chiamate Feed-forward perché l'informazione viaggia solo in avanti all'interno della rete, attraverso i vari nodi. Esso consiste in un numero di semplici unità di elaborazione simili a neuroni, organizzate in strati: uno strato di input, uno o più strati nascosti e uno strato di output. Ogni nodo, o neurone artificiale, si connette a un altro e ha un peso e una soglia associati, i quali codificano la conoscenza della rete. Se l'output di un singolo nodo è superiore al valore di soglia specificato, quel nodo viene attivato, inviando i dati al livello successivo della rete. In caso contrario, nessun dato viene trasferito al livello seguente.

Il processo di apprendimento è costituito da un periodo di addestramento, in cui la rete, tramite un insieme di coppie input-output, chiamato training set, 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. Dato il set di addestramento, l'algoritmo di apprendimento regola, iterativamente, i parametri, ovvero i pesi e le distorsioni, in modo che il modello possa mappare accuratamente gli input sugli output. L'algoritmo riduce i pesi, tramite un piccolo moltiplicatore del tasso di apprendimento, se l'output previsto è maggiore dell'output atteso, e

li aumenta in caso contrario [2]. La rete fornisce, pertanto, in maniera ripetuta un output e il valore di scostamento tra esso e l'output desiderato. L'obiettivo è quello di minimizzare questa differenza finchè non si ottiene il valore ambito. A questo punto, la fase di addestramento termina ed inizia una fase di test che viene eseguita utilizzando dei vettori, detti test set. In figura 4.3 è riportato un esempio di rete Feed-forward con un livello nascosto.

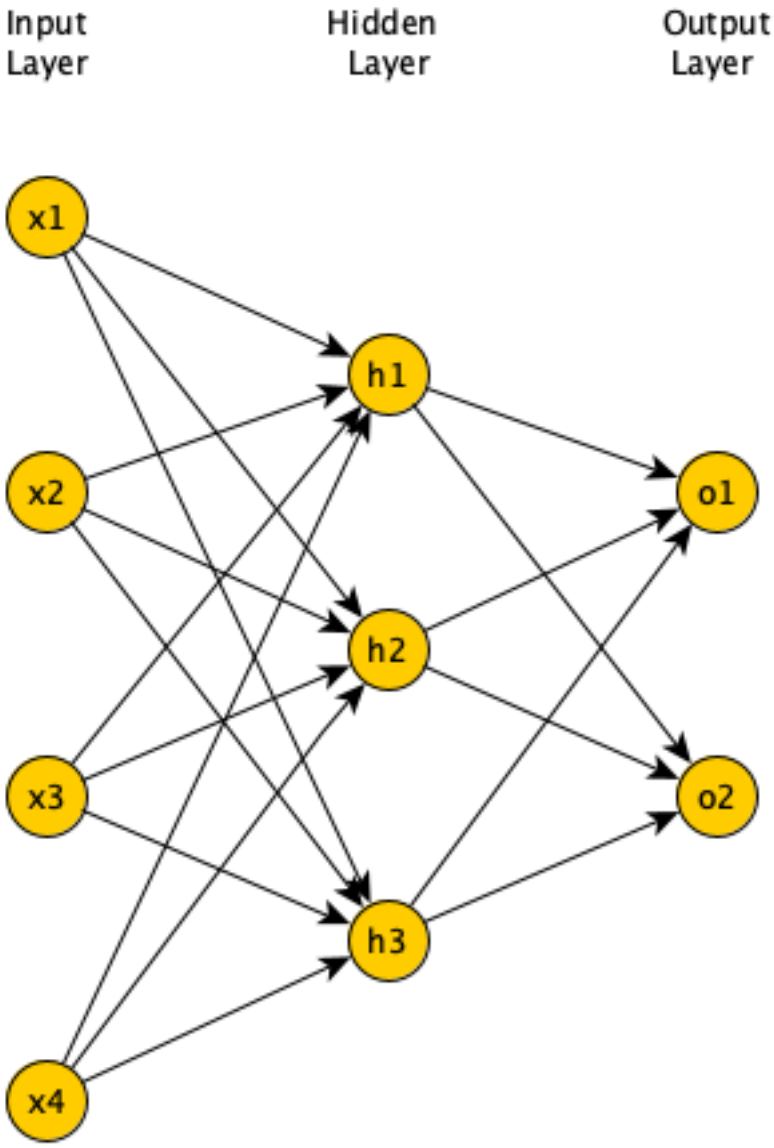

Figura 4.3. Rete neurale Feed-forward con un livello nascosto

[2] https://training.galaxyproject.org/training-material/topics/statistics/tutorials/FNN/tutorial.html

Nella figura sopra riportata 4.3 sono rappresentati i principali attori della rete in questione. Il processo è costituito da tre tipi di layers e le uniche connessioni esistenti sono tra un layer e quello successivo. L'Input Layer è composto 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 compreso tra 0 e 1, e, se è connesso ad un altro neurone, un peso w, solitamente un valore di tipo double. Per ottenere un neurone del layer successivo si utilizza la seguente formula:

$$newneuron = \sigma(w_1 a_1 + ... + w_n a_n + b)$$

dove b rappresenta il bias, il cui scopo è quello di approssimare, e σ è la funzione di attivazione utile a normalizzare l'output in modo tale che sia nuovamente compreso tra 0 e 1.

4.3 Reti Feed-forward backpropagation

La backpropagation è un metodo di addestramento supervisionato di reti neurali artificiali che valuta il contributo all'errore di ciascun neurone dopo l'elaborazione di un insieme di dati. L'obiettivo della backpropagation è modificare i pesi in modo da addestrare la rete a mappare correttamente input arbitrari in output. L'algoritmo mette a punto i pesi di una rete neurale in base al tasso di errore ottenuto nell'iterazione precedente. Una corretta parametrizzazione dei pesi consente di ridurre i tassi di errore e rendere affidabile il modello aumentandone la generalizzazione. Il minimo della funzione di errore nello spazio peso viene calcolato utilizzando il metodo della discesa del gradiente [19], riducendo la differenza della somma al quadrato tra il target e le uscite effettive della rete, i quali vengono poi utilizzati per effettuare il training della rete stessa.

L'analisi si basa normalmente sull'assunzione di una funzione di attivazione semilineare (differenziabile e monotona) quale il sigmoide. L'algoritmo si struttura in due fasi principali: una fase feed-forward, in cui viene applicato un vettore di ingresso e il segnale si propaga attraverso gli strati di rete, modificato dai pesi e dalle polarizzazioni correnti e dalle funzioni di attivazione non lineari. Emergono, così,

valori di output corrispondenti, e questi possono essere confrontati con gli output target per il dato vettore di input utilizzando una funzione di perdita. In seconda battuta si procede con una fase di feedback in cui il segnale di errore viene rinviato (backpropagato) attraverso i livelli di rete per modificare i pesi in modo da ridurre al minimo l'errore nell'intero training set, riducendo il più possibile la superficie di errore nello spazio dei pesi.

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:

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

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

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

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

Ciascuna derivata parziale legata ai pesi e ai bias viene memorizzata 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, si esegue la media degli output per ogni peso e ogni bias. L'output del vettore finale 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:

$$w^l = w^l - learningrate * \frac{\partial C}{\partial w^L}$$

$$b^l = b^l - learningrate * \frac{\partial C}{\partial b^L}$$

In figura 4.4 è riportato uno schema riepilogativo del funzionamento della Feed-forward backpropagation neural network.

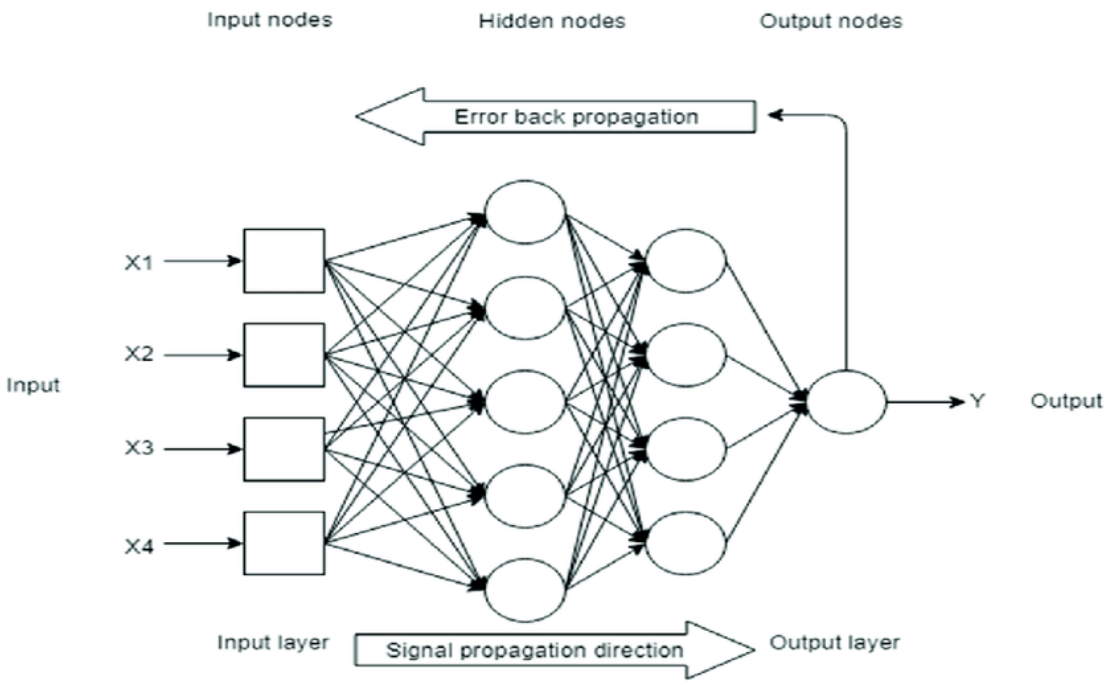

Figura 4.4.

Feed-forward backpropagation neural network

4.4 Reti neurali ricorrenti

Una rete neurale ricorrente (RNN) è un tipo di rete neurale artificiale che utilizza dati sequenziali o dati di serie temporali. Le reti neurali ricorrenti oltre a sfruttare i dati di addestramento per apprendere, si contraddistinguono per la loro "memoria" in quanto prendono informazioni da input precedenti per influenzare l'input e l'output correnti. La RNN può ricevere in input e produrre in output uno o più vettori, i quali sono influenzati sia dai pesi legati agli input sia da vettori "nascosti" che rappresentano il contesto correlato ai precedenti input e output e vengono aggiornati in modo ricorrente. L'utilizzo degli stati nascosti permette, inoltre, di identificare le relazioni tra i neuroni vicini anche in una situazione in cui si hanno una serie di input diversi. La maggior parte delle reti neurali ricorrenti presentano l'ulteriore vantaggio di essere bidirezionali: ciascun neurone, infatti, 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.

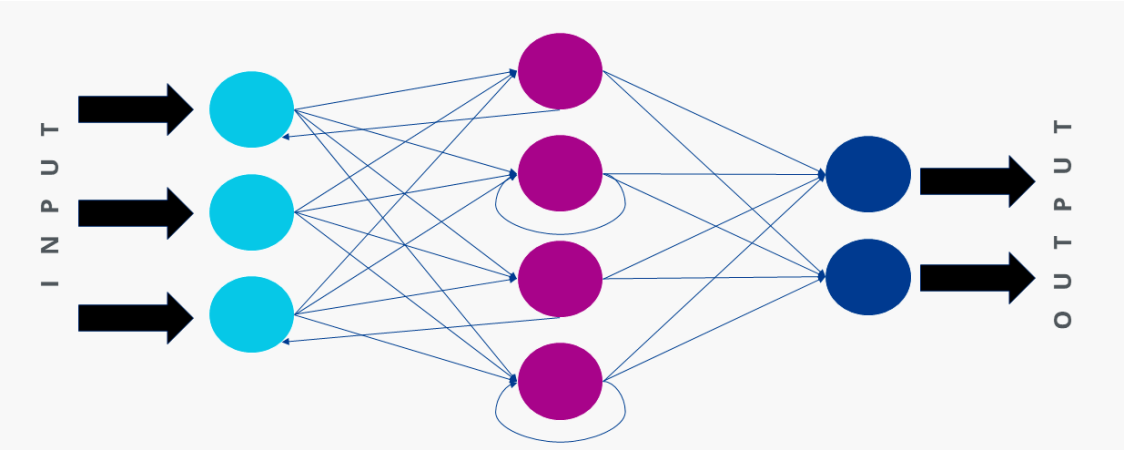

Figura 4.5.

Rete neurale ricorrente bidirezionale

4.5 Il machine learning per progetti software

Come discusso precedentemente, il monitoraggio e il controllo di un progetto software è particolarmente complesso e più critico di quanto non lo sia per un prodotto più fisicamente tangibile. Per questo motivo, la pianificazione di un progetto software è una delle attività più delicate dell'intero processo ed eventuali carenze rischiano di generare inefficienza nella sua gestione.

La possibilità di adoperare modelli di machine learning per la previsione dei costi da sostenere, sia in fase di implementazione che durante il corso della vita del prodotto, consentirebbe una valutazione più accurata delle prospettive del progetto e agevolerebbe riflessioni sui principali fattori coinvolti. Tuttavia, la scarsità di dati empirici e la difficoltà nel costruire un framework ben definito con relazioni ben chiare tra i descrittori chiave, uniti alla prassi per cui le stime dei costi più comuni si basano su un approccio top-down piuttosto che bottom-up, rendendo più difficile la raccolta dei dati, rappresentano le maggiori criticità per l'utilizzo dell'apprendimento automatico in questo settore.

Nel capitolo successivo si analizzerà il database a disposizione e si applicheranno alcuni algoritmi di machine learning per approfondire il possibile impiego di stime predittive dei costi. In particolare, il focus sarà, come anticipato nei paragrafi precedenti, sulla fase di interesse di follow up, immediatamente successiva al rilascio in

produzione. L'applicazione dell'apprendimento automatico potrebbe, così, agevolare l'esplorazione economica di una fase poco investigata e finanziariamente sottovalutata.

Capitolo 5

Caso di studio: analisi di progetti software in una società di consulenza IT

Il caso di studio preso in esame è stato analizzato ed elaborato durante l'esperienza di stage curriculare vissuta in una società di consulenza ingegneristica. L'azienda opera nei settori di tecnologia dell'informazione, elettronica, software embedded, reti di telecomunicazioni e ingegneria; nello specifico, la software factory in cui si è svolto il tirocinio si dedica alla creazione di attività per il supporto di processi aziendali e all'implementazione di soluzioni innovative per un noto istituto bancario italiano.

Le fasi che si susseguono nella realizzazione dei vari progetti sono le seguenti:

- Analisi funzionale;
- Pianificazione dei costi e dei tempi;
- Esecuzione del progetto;
- Consegna del prodotto al cliente.

La direzione sistemi informativi della banca fornisce, all'azienda di consulenza, le specifiche tecniche del progetto che desidera venga realizzato; questa analisi dettagliata, che comprende i requisiti qualitativi e quantitativi, è definita *Analisi funzionale* e consente l'individuazione degli indicatori chiave utili alla valutazione dei risultati ottenuti. Questa AFU è il primo contatto diretto con il cliente a seguito

del quale viene presa in carico l'attività progettuale. A questo punto i consulenti si dedicano alla stesura della stima per quanto concerne i costi e l'effort necessari alla realizzazione del progetto, procedendo prevalentemente per analogia con i lavori precedenti. La stima viene vagliata, in un primo momento, dal Project Manager che, confermandola, acconsente all'invio al cliente; l'acquirente può così valutare ed accettare la proposta, concordando l'avvio ufficiale dell'attività.

Ciascuna attività attraversa alcune fasi preliminari:

- Analisi e pianificazione: l'obiettivo di questa fase è quello di analizzare i requisiti del progetto per poter individuare i work packages in cui viene suddivisa l'attività, ed effettuarne una stima meticolosa. I consulenti si occupano della stesura dei casi d'uso per poter determinare la dimensione e la complessità del codice e consentire la definizione accurata delle richieste del cliente. Gli use case diagrams, basandosi sugli standard UML, pongono in evidenza le interazioni tra il sistema con gli applicativi esterni, mostrano gli attori coinvolti e la sequenza di azioni da svolgere per la completa funzionalità del software, le potenziali condizioni da verificarsi, la motivazione ultima per cui l'utente desidera interfacciarsi con il programma, gli eventuali errori e le possibili risposte del sistema. La totalità dei casi d'uso interpreta le funzionalità da sviluppare fornendo, attraverso la metodologia dei function points, un'indicazione sulla dimensione e sulla laboriosità del progetto.

  La tecnica dei FP si applica indipendentemente dal linguaggio di programmazione adottato, si dimostra, pertanto, universalmente utilizzabile per la maggior parte dei progetti software. Questa metodica si basa sulla tipologia dei requisiti forniti, sul numero di informazioni in input e in output, e sulla quantità di memoria necessaria; elaborando queste informazioni genera una misurazione quantitativa ottenendo il numero totale di fuction points corrispondenti al progetto.

  Dal punto di vista gestionale è fondamentale identificare, in questa fase, le milestones del progetto, per poter pianificare nel tempo le varie attività e potervi allocare strategicamente le risorse a disposizione.

- Stima dei tempi e dei costi: a seguito della definizione dei work

packages e della valutazione delle relative durate, è necessario verificare le possibili relazioni di interdipendenza operativa tra le varie attività per poter stabilire l'ordine in cui queste devono essere realizzate. Per ogni work package viene stimato l'effort richiesto per la sua realizzazione e vengono identificate e quantificate le difficoltà ad esso correlate, associandogli, conseguentemente, un numero di story points rappresentativo di tali parametri stabiliti. Gli story points sono calcolati sulla base di due fattori prevalenti, i function points derivanti dall'analisi funzionale e l'analogia con progetti sviluppati antecedentemente.

Le similitudini riscontrate in attività sviluppate nel passato costituiscono un grande bagaglio di esperienze da cui attingere, risulta quindi fondamentale l'archiviazione e l'aggiornamento di un database che mostri sempre una validità attuale ed un'affidabilità corrente, che possa essere sfruttato come totalità di lessons learned. L'archiviazione dei documenti connessi ad attività sviluppate in precedenza, consente di revisionare le stime effettuate e confrontarle con ciò che concretamente è accaduto: in particolar modo, si rivela utile per l'analisi dell'effort necessario al progetto, analizzando gli scostamenti che si sono verificati tra quello pianificato e quello effettivo; permette, inoltre, di affinare il calcolo degli story points in modo tale da includere la contingency a cui ricorrere in caso di forti criticità.

A seguito della validazione della stima da parte del Project Manager, avviene una conversione in termini di costi e di tempi necessari al raggiungimento dell'obiettivo. Questa trasformazione si realizza mediante un fattore di conversione che, sulla base della produttività media emersa nello sviluppo dei progetti passati, associa 1 story point a 1,5 giorni di lavoro per ogni risorsa impiegata. Il valore di questo fattore viene rivisitato e aggiornato periodicamente in base alle performance e all'esperienza acquisita dalle risorse, e presenta generalmente un andamento decrescente, ad eccezione dei momenti in cui vengono inserite nuove risorse che necessitano di un periodo di inserimento e hanno una produttività iniziale minore.

Il processo di stima si conclude con la definizione dei costi di progetto, che vengono ricavati moltiplicando il numero totale dei giorni pianificati per una tariffa fissa giornaliera.

Un tipico esempio di stima è riportato nella figura 5.1.

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 5.1. Stima di un'attività

5.1 L'esecuzione del progetto in azienda

Il processo che porta allo sviluppo del progetto segue le macro fasi standard analizzate nei paragrafi precedenti, caratterizzandosi, tuttavia, in particolari step; a seguito dell'individuazione dei requisiti e dell'elaborazione di una stima, si procede come segue:

- Sviluppo
- Application Test
- System Test
- User Acceptance Test
- Produzione
- Follow up e Manutenzione

In concomitanza al concreto sviluppo delle attività, i programmatori elaborano la documentazione tecnica (ATE) a supporto dell'implementazione del software, la quale verrà consegnata al cliente al completamento del progetto.

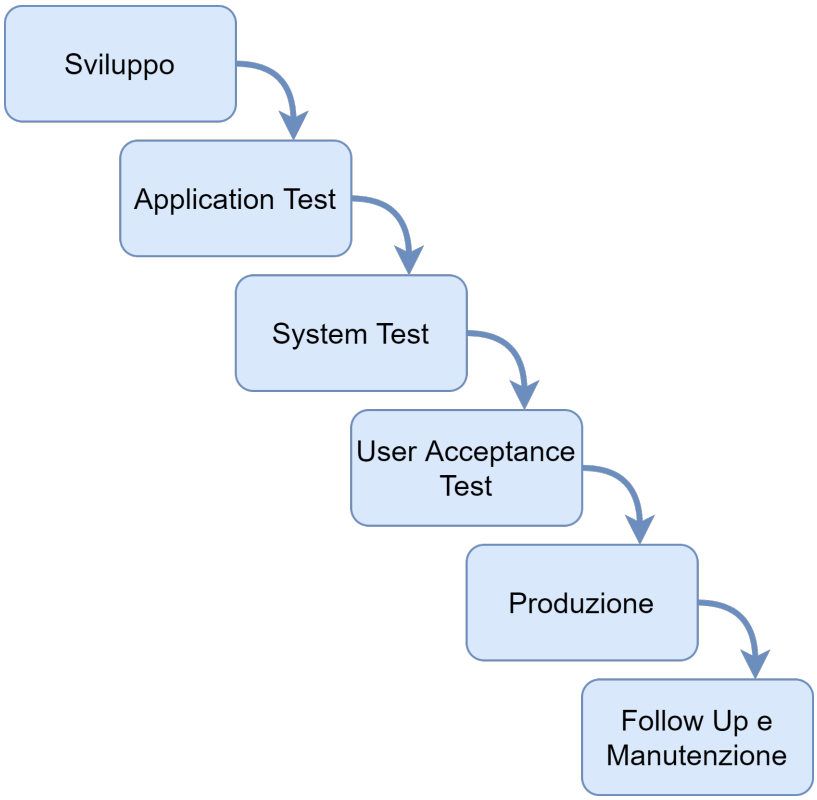

Figura 5.2. Fasi di esecuzione del progetto

5.1.1 Lo sviluppo

Prima che abbia inizio la fase di sviluppo, sia il cliente che il fornitore, creano il cosiddetto *branch*, il nuovo ramo progettuale che rappresenta la nuova attività da portare a termine. Dal lato del fornitore il branch è realizzato all'interno dell'ambiente locale di sviluppo, che sarà accessibile agli sviluppatori grazie ad un'utenza personale e certificata tramite una firma digitale, per consentire all'istituto bancario di poter supervisionare e controllare chi scrive il codice.

Il *core* dello sviluppo del software è indubbiamente la stesura del codice sorgente, lo strumento che permette l'esecuzione delle funzionalità e dei servizi richiesti dal cliente. L'implementazione avviene all'interno di un ambiente locale costituito da server fisici e cloud in cui sono presenti gli applicativi necessari alla stesura del codice. In questa fase è necessario che i consulenti siano lungimiranti verso quella che risulta essere una delle criticità maggiori di questa fase: la capacità di scrivere un codice che possa funzionare anche quando sarà integrato con gli altri sistemi preesistenti.

La programmazione totale viene suddivisa nella creazione di singoli moduli che verranno assemblati successivamente per comporre il prodotto finale. Ogni modulo è testabile dallo sviluppatore che ne verifica la correttezza delle istruzioni attraverso un test definito Unit Test.

5.1.2 L'Application Test

L'Application Test è realizzato all'interno di un ambiente definito *Svil* e rappresenta il primo livello di validazione interna del software. Il primo passaggio che si effettua in questa fase è la compilazione dei test J-Unit, parti di codice stilate in ambiente locale dagli sviluppatori stessi che fungono da auto verifica del proprio lavoro sviluppato. Sulla base di quelle che sono le specifiche recepite dal programmatore, egli stesso scrive i test basilari che il suo prodotto dovrà superare affinché sia conforme ai requisiti, e li avvia nel momento in cui avviene la compilazione del codice.

Nella fase di Application test si evidenzia l'eventuale presenza di importanti incompatibilità strutturali in relazione agli standard di sviluppo aziendali, intercettando, così, gravi ed evidenti errori di programmazione. Questa procedura è eseguita tramite applicativi e strumenti dedicati che, compilando il codice, eseguono i test sulle principali funzioni implementate. Nel caso in cui vengano riscontrate delle anomalie, lo sviluppatore effettua un debug manuale sulle singole istruzioni e ne corregge gli errori emersi. Questa tipologia di test si caratterizza per una particolare velocità di esecuzione, in quanto viene applicata sulle funzioni ritenute fondamentali. Al termine della fase di Application, il codice verificato viene incapsulato all'interno di una *Unità di cambiamento* creata dal cliente; questa UDC racchiude tutte le evolutive software legate ad un singolo progetto e rappresenta l'ultimo passo prima della fase successiva del ciclo di vita del progetto.

5.1.3 Il System Test

Conclusa la programmazione in locale ed effettuato l'Application Test, è possibile dare avvio alla fase di System Test con cui factory preposte verificano, in maniera puntuale, la soddisfazione dei requisiti definiti inizialmente nelle specifiche consegnate dal cliente.

Prima di procedere con lo svolgimento dei test, è fondamentale stabilire il System Test Plan: un piano che definisce scopo e obiettivi,

descrive le aree potenzialmente critiche, il contesto, i risultati intermedi e quello finale atteso, la pianificazione temporale e sequenziale dei test, le metodologie di esecuzione, i parametri di sospensione e di ripresa, i criteri di accettazione e di rifiuto, l'ambiente di verifica, ed infine le responsabilità ed i ruoli dei vari attori.

Il nucleo della fase di System Test è incentrato sull'esecuzione, manuale o automatica, di un determinato numero di *test cases* correlati alle funzionalità della *user interface*. Questi test si suddividono, generalmente, in test standard, effettuati per qualsiasi tipologia di software, e test più specifici che indagano le funzionalità del codice in esame valutando che i risultati corrispondano a quelli attesi.

Il trasferimento del codice validato dall'ambiente di *Svil* a quello di *System* avviene tramite l'UDC che comprende alcuni changeset, ovvero l'insieme di direttive utili allo svolgimento dei test: i nomi dei file da rilasciare, i relativi owner, e tutte le informazioni necessarie al superamento di questa fase. Uno tra gli aspetti fondamentali di questo step è la risoluzione dell'*Integration Test*, che verifica la corretta compatibilità e funzionalità delle varie parti sviluppate, spesso da owner differenti, una volta assemblate tra loro per comporre il prodotto finale. Ulteriore fattore centrale di questa fase è il cosiddetto *No Regression Test*, utile ad individuare possibili anomalie riscontrabili nel momento in cui vi sia interazione tra il nuovo codice e le architetture preesistenti. Solo nel caso in cui non vi siano criticità è possibile procedere con un test massivo dell'intero codice, in cui viene validata non solo l'effettiva integrazione tra il nuovo prodotto e le varie periferiche esterne, ma anche tutte le funzionalità che il software deve rispettare per poter soddisfare i requisiti richiesti. Si procede fornendo al sistema degli input prestabiliti e se ne verifica la coerenza e correttezza degli output (metodo del black-box testing) anche dal punto di vista grafico e delle tempistiche di recovery. I test vengono svolti in maniera iterativa: ogniqualvolta un test fallisce, si apre un *defect*, si formalizza quindi la presenza di una disfunzione che dovrà essere risolta dai programmatori addetti allo sviluppo. Il processo termina quando il codice risponde nel modo atteso e si raggiungono, così, i risultati desiderati. È essenziale sottolineare che in questa fase si testa unicamente la correttezza formale del prodotto e non l'esperienza vissuta dal tester in quanto utilizzatore finale del software e quindi sostituto del cliente ultimo.

Conclusa questa fase di esecuzione dei test, è di primaria importanza la stesura della documentazione che attesti le problematiche

riscontrate e il funzionamento del software testato, prima che questo possa giungere alla fase di User Acceptance Test.

5.1.4 Lo User Acceptance Test

La fase di User Acceptance Test è effettuata da un campione selezionato di utenti di business, definiti *Beta Tester*, i quali si occupano di svolgere una serie di prove di utilizzo alle evolutive in via di rilascio. Gli addetti si dedicano alle verifiche degli aspetti riguardanti l'interfaccia grafica e la user experience dell'utilizzatore ultimo; sono, pertanto, completamente indifferenti alle logiche sottostanti e al linguaggio di programmazione utilizzati in fase di sviluppo.

Questo step si raggiunge solo in caso di esito positivo di tutti i test precedentemente svolti e rappresenta l'ultimo scoglio da superare prima che il prodotto possa essere rilasciato in produzione. L'importanza della fase di UAT consegue dal fatto che gli sviluppatori devono confrontarsi e realizzare funzionalità inesistenti e, se pur dettagliatamente descritte, sicuramente soggette ad interpretazione. Alcuni aspetti dei requisiti, inoltre, risultano comprensibili solo nell'ottica della fruibilità da parte del cliente finale; si dimostra quindi indispensabile che chi ha effettuato le richieste funzionali convalidi il software anche da questo punto di vista.

L'ambiente di UAT, e più in generale tutte le fasi di test, consentono, laddove necessario, di apportare modifiche al software ad un costo nettamente inferiore rispetto alle spese che si dovrebbero sostenere nel caso in cui le rettifiche dovessero essere elaborate dopo il rilascio in produzione; post release è inevitabile che vi sia il coinvolgimento di terze entità architetturali e il potenziale danno all'immagine del fornitore si appesantirebbe notevolmente.

Anche per la fase di User Acceptance Test è essenziale che avvenga la stesura della documentazione che riporti dettagliatamente la descrizione di ciò che è avvenuto in questa fase relativamente ai vari test cases effettuati. È ragguardevole, inoltre, che si pianifichino al meglio i tempi per poter risolvere eventuali problematiche sollevate a seguito delle verifiche, entro la data del rilascio, includendo in fase di stima possibili buffer temporali.

5.1.5 Il rilascio in 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 hanno superato con esito positivo i test e, pertanto, non devono subire ulteriori modifiche. Questo ambiente presenta delle caratteristiche strutturali e funzionali identiche a quelle dell'ambiente di produzione e permette di rilasciare tutte le UDC prossime al rilascio nello stesso momento. Il passaggio avviene sincronizzando i due ambienti ed effettuando, al momento della release, una semplice modifica ai riferimenti dei database; si procede, inoltre, con la realizzazione di una copia dell'ambiente che verrà utilizzato, nel prossimo rilascio, come ambiente di pre-produzione. 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. Da questo istante il prodotto è ufficialmente fruibile dal cliente finale, il quale, accedendo all'internet banking, potrà visionare ed utilizzare concretamente le evolutive apportate al sito.

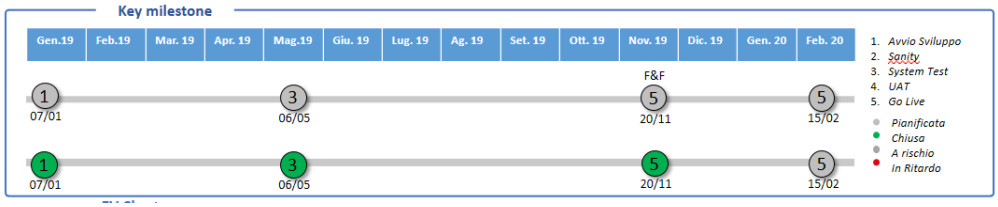

Figura 5.3. Fasi salienti del processo in azienda

5.1.6 Follow up e Manutenzione

A seguito del rilascio in produzione del prodotto, nasce l'esigenza di far fronte ai problemi che potrebbero sorgere lato utente con l'utilizzo del sito; ha inizio, pertanto, la fase di manutenzione che sarà necessaria sino a quando il software avrà vita. È possibile suddividere questa fase in due parti: il periodo definito di *Follow up*, e quello propriamente chiamato di *Manutenzione*. Il follow up è un lasso temporale di circa un anno ed è possibile descriverlo come un periodo di garanzia, nel quale qualsiasi problematica possa emergere deve essere risolta dagli stessi sviluppatori che hanno programmato il software. Nel caso in cui il cliente riscontri delle anomalie o dei problemi, vengono

aperti degli *incident*, ovvero dei ticket, che devono essere analizzati e risolti da chi si è occupato della stesura del codice. Solo al termine di questo intervallo di tempo, avviene un passaggio di consegna delle competenze necessarie legate al progetto verso il gruppo dedicato alla vera e propria manutenzione; sarà, pertanto, il gruppo di Application Maintenance che si dedicherà in toto alla risoluzione delle problematiche legate al sito. Se la sezione di AM è specificatamente preposta allo svolgimento di questo compito, al contrario, per il gruppo di sviluppatori che si occupano dell'implementazione delle varie attività, questi aspetti rappresentano solo una piccola parte dei loro compiti. Da qui nasce l'ambizione di voler quantificare questo impegno, che si dimostra essere sempre più laborioso, per potergli attribuire il giusto valore e consentire una valutazione accurata delle spese, potendole, così, inserire in fase di stima nel momento in cui si effettua la proposta economica al cliente.

5.2 Analisi del Database

L'azienda in questione, presso cui è stata svolta l'esperienza di stage, utilizza come metodo di stima il processo di analogia. Nonostante questa metodologia eviti il ricorso ad esperti, l'utilizzo di modelli ed algoritmi parametrici e complessi, e risulti complessivamente semplice, presenta alcuni limiti da analizzare.

In primo luogo, è necessario svolgere un processo prolisso di raccolta di dati storati che spesso si dimostra impreciso e caratterizzato da errori in fase di analisi. In secondo luogo, è possibile che si presentino casi di falsa analogia, ovvero è plausibile che due progetti possano sembrare molto simili in linea teorica, e rivelarsi, solo successivamente, sostanzialmente e notevolmente differenti. È un metodo intrinsecamente incerto e soggettivo che potrebbe portare a valutazioni discordanti se effettuate da soggetti diversi.

Tuttavia, il mutamento ad un utilizzo preponderante di algoritmi di intelligenza artificiale richiede, a monte, uno studio approfondito sull'identificazione dei principali descrittori del progetto, con la conseguente costruzione di un framework ben delineato, i cui risultati in output devono essere accuratamente analizzati ed interpretati.

Il primo step di questo processo è una raccolta meticolosa dei dati utili, che devono essere studiati e filtrati per plasmare ed efficientare il database su cui effettuare l'analisi. A seguito dell'individuazione

delle variabili da esaminare, si prosegue con l'osservazione delle relazioni emerse tra i descrittori, interpretando i risultati della stima elaborata tramite algoritmi di tipo black-box. Si verifica la validità e l'affidabilità degli output ricavati confrontandoli con quelli ottenuti da una metodologia classica come, per esempio, la regressione lineare; questo accertamento avviene con l'applicazione della sensitivity analysis, analisi in grado di fornire indicazioni sull'errore standard associato ad entrambi i metodi (processo illustrato in figura 5.4).

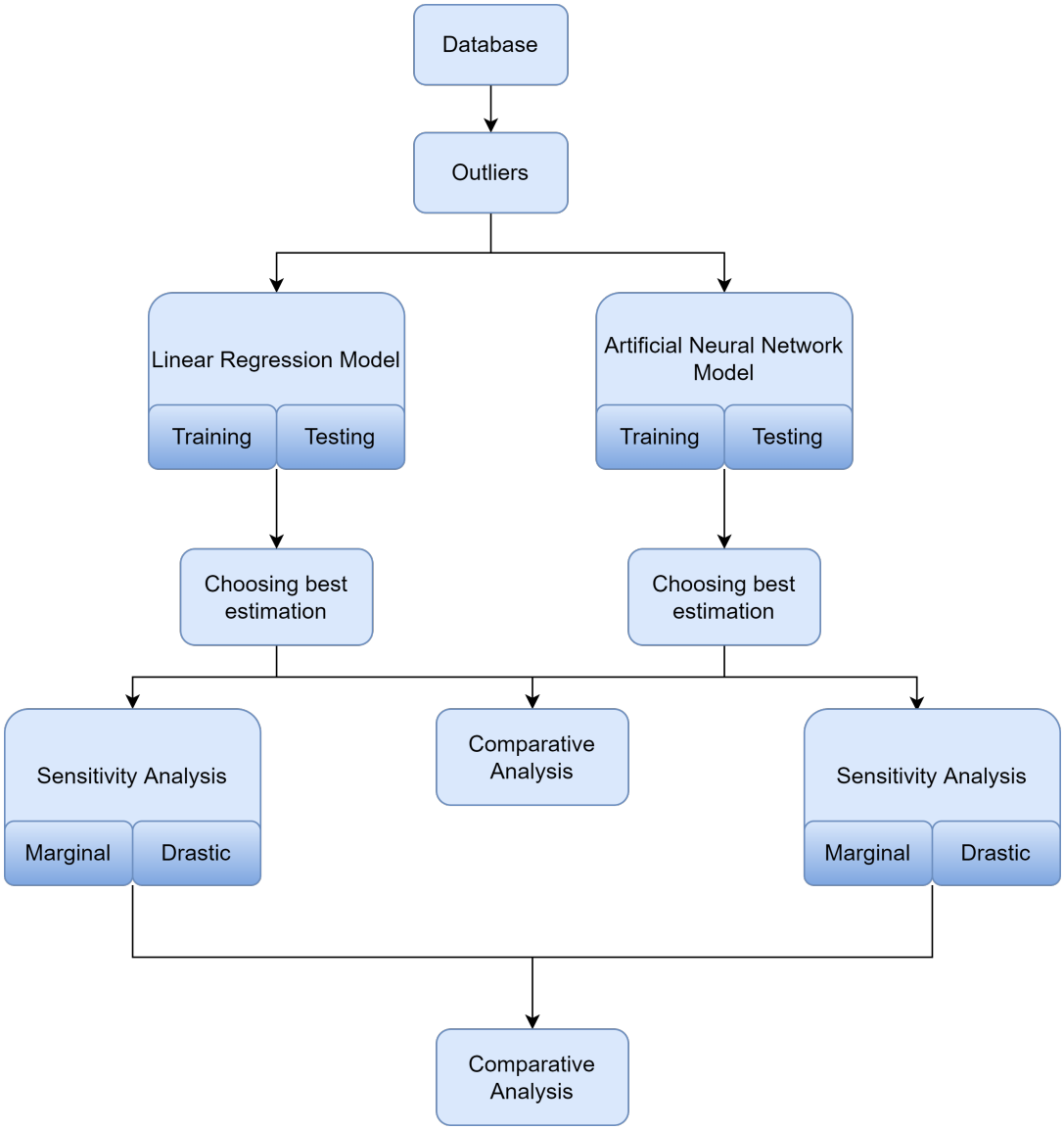

Figura 5.4. Modello per ottenere il framework per la stima

Ad ogni modo, il processo di stima in azienda è elaborato sino alla fase del rilascio in produzione, escludendo da tale previsione la fase di follow up, che, come spiegato nella sezione 5.1.6, risulta ancora in carico al gruppo il quale si è occupato dello sviluppo del progetto;

pertanto, questo onere in termini di tempo e, soprattutto, di costi non è incluso nella quantificazione economica. Di seguito verranno analizzate le variabili che si è deciso di prendere in considerazione per approfondire l'analisi; verranno illustrati i risultati ottenuti con l'utilizzo di differenti algoritmi, analizzandone le informazioni in uscita e i relativi errori; in tal modo sarà possibile esprimersi su quale sia il modello che meglio si adatta alla spiegazione delle variabili in gioco e delle loro relazioni. L'obiettivo è, quindi, quello di riuscire a proporre l'utilizzo del modello migliore per l'elaborazione di una previsione della fase di follow up fondata su algoritmi affidabili e innovativi di intelligenza artificiale, che potrà integrare e completare la stima dei progetti, rendendola, così, esaustiva, accurata e totalizzante.

5.3 Variabili del progetto

Il primo passo per raggiungere l'obiettivo è l'identificazione delle variabili direttamente coinvolte nell'analisi, ovvero gli indicatori chiave che consentono di verificare e valutare le relazioni significative che genereranno il modello.

Il database esaminato è composto da circa 500 attività che sono state sviluppate dal team di progetti dell'azienda e che, al momento dello studio, risultano essere completate. I descrittori da cui si è scelto di iniziare l'analisi, per una disamina preliminare, sono i seguenti:

- ACWP: Actual cost of work performed. Rappresenta una delle principali metriche, assieme al BCWS e all'EV, utilizzate per monitorare e verificare l'andamento del progetto nel settore del Project Management. Consente di quantificare le performance dell'attività identificando i problemi di costo prima che diventino delle vere e proprie criticità, ed aiuta il team a focalizzarsi sul completamento dei propri task intervenendo ove necessario. Per ogni attività analizzata si è tenuto traccia di tutti e tre gli indicatori, tuttavia, visto il focus del tema sugli aspetti economici, si è preferito concentrarsi sull'indicazione dei costi sostenuti nel corso dello sviluppo del progetto.

- FTE Gruppo: rappresenta il numero totale di risorse occupate sulla singola attività. In tutti i lavori è presente un Owner, ovvero il responsabile del progetto, con cui collaborano fino a 5 persone; il valore del FTE gruppo varia, pertanto, da 1 fino ad

un massimo di 6. Si è deciso di includere questo fattore nell'analisi complessiva, per poter valutare se vi sia un legame tra i costi sostenuti nella fase di follow up e il numero di programmatori che hanno sviluppato il task; si potrà stabilire, così, se vi sia correlazione tra le problematiche nate post rilascio e la numerosità del gruppo di lavoro.

- Numerosità defect: è il numero di anomalie riscontrate nelle fasi di testing del software. A differenza dei prodotti più classici e tangibili, è evidente che vi sia una complessità maggiore per l'identificazione dei difetti e per la valutazione degli standard di qualità raggiunti. L'unico indicatore che possa ritenersi affidabile e valido per la valutazione della bontà dell'output di progetto è il numero di segnalazioni ricevute dal fornitore da parte dei clienti finali. Per ogni attività vi è una tracciabilità dei defect ricevuti e che sono stati assegnati ai programmatori facenti parte del gruppo di lavoro competente per quel determinato task. Nell'analisi seguente si cercherà una relazione tra le problematiche sorte pre e post produzione.

- Family & Friends: con la definizione di "Family & Friends" l'istituto bancario, cliente dell'azienda presso cui è stato svolto il tirocinio, indica una particolare cerchia ristretta di utenti finali. Questa singolare categoria di clienti, tra le varie peculiarità, ha quella di poter fruire di alcune evolutive del sito in tempi precedenti rispetto a tutti gli altri consumatori. Vi sono infatti dei casi in cui lo sviluppo o le migliorie apportate al software sono considerate particolarmente delicate, in queste casistiche prima del classico rilascio al pubblico, vi è una fase di produzione visibile solo a questa determinata classe di utenti. I casi in cui la peculiarità del progetto richiede che vi sia un'accurata attenzione nel momento in cui il suo utilizzo sia aperto a tutti gli utenti, si prevede che vi sia un'anteprima di produzione presso i cosiddetti family&friends; d'altronde, questi fruitori sono, sovente, attori interni alla banca stessa, i primi che si interfacceranno con il programma e ne potranno verificare eventuali problematiche anticipatamente rispetto alla completa divulgazione. Tale variabile è di tipo booleano e indica, per ogni attività, se il rilascio sia stato effettuato in modalità F&F oppure secondo le regole tradizionali.

- FTE Performance: Full Time Equivalent Performance è la misura della produttività delle risorse assegnate al progetto. Questo

valore è particolarmente complesso da assegnare poiché richiede in input fattori come la conoscenza, l'esperienza, il tempo investito, e ottiene in output un codice o un software; determinare la produttività di un processo ingegneristico è uno sviluppo poco intuitivo: è, infatti, evidente che la relazione tra input e output non sia linearmente quantificabile né particolarmente immediata per essere valutata con semplicità. Bisogna tener conto anche della pratica molto diffusa nel mondo dello sviluppo attuale, che prevede il riutilizzo di parti di codice già implementate precedentemente: la conseguenza tangibile, questa, della presenza di diffuse analogie tra i vari progetti. Nel caso in esame, la produttività di ciascun gruppo di lavoro è stata calcolata assegnando ad ogni risorsa coinvolta un fattore di performance stabilito dal Project Manager, ed effettuandone la somma per tutti gli attori del progetto; tale assegnazione è avvenuta secondo criteri soggettivi e basandosi sul bagaglio dell'esperienza dei progetti passati.

- Duration: 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.

- Complessità progetto: questa metrica è di particolare e difficoltosa identificazione. Un alto livello di complessità progettuale è legato, solitamente, ai costi sostenuti, ai tempi impiegati, alle tecnologie e i file coinvolti, alle varie interdipendenze delle parti correlate e alla differenziazione all'interno del progetto. Il Project Manager può fare affidamento, oltre che alla sua esperienza, anche sul numero di function points attribuiti ad ogni task; su tale assunto, nel caso in esame, avviene la suddivisione in 3 fasce di differente complessità su cui si realizza la classificazione delle attività.

- Costi di Follow Up: è la variabile target del modello, ovvero la variabile dipendente. Poiché tutte le attività in analisi risultano essere già concluse, si hanno i riferimenti alle spese sostenute in fase di follow up per ogni progetto, valori tracciati in questo caso, ovviamente, a posteriori. L'analisi che verrà condotta tenterà di stabilire le dipendenze e le relazioni tra questo fattore e tutti quelli presi in esame, per poter individuare un modello che, sulla base degli elementi a disposizione, riesca a prevedere anticipatamente tali costi.

PROJECT	Owner	Risorsa 1	Risorsa 2	Risorsa 3	Risorsa 4	Risorsa 5	FTE Performance	ACWP	Duration	FTE Gruppo	Stato Attività	Complessità Progetto	Numerosità Defect	F&F	Costi Follow up
Project 1	x	x	x	x			5	0.00 €	x	4	Completata	3	14	0	0.00 €
Project 2	x						3	0.00 €	x	1	Completata	1	13	1	0.00 €
Project 3	x	x	x	x	x	x	3	0.00 €	x	6	Completata	1	24	1	0.00 €
Project 4	x	x					6	0.00 €	x	2	Completata	2	25	1	0.00 €
Project 5	x	x	x				8	0.00 €	x	3	Completata	2	20	0	0.00 €
Project 6	x						8	0.00 €	x	1	Completata	2	35	0	0.00 €
Project 7	x	x					6	0.00 €	x	2	Completata	3	14	0	0.00 €
Project 8	x						5	0.00 €	x	1	Completata	3	8	1	0.00 €
Project 9	x	x	x	x	x	x	3	0.00 €	x	6	Completata	3	23	0	0.00 €
Project 10	x	x	x	x			5	0.00 €	x	4	Completata	3	20	1	0.00 €
Project 11	x	x					6	0.00 €	x	2	Completata	3	1	1	0.00 €
Project 12	x	x	x				6	0.00 €	x	3	Completata	3	5	1	0.00 €
Project 13	x	x	x	x	x		8	0.00 €	x	5	Completata	1	15	0	0.00 €
Project 14	x	x					5	0.00 €	x	2	Completata	2	23	0	0.00 €
Project 15	x	x	x				3	0.00 €	x	3	Completata	3	9	0	0.00 €
Project 16	x						5	0.00 €	x	1	Completata	3	8	1	0.00 €
Project 17	x	x					6	0.00 €	x	2	Completata	2	31	1	0.00 €
Project 18	x	x	x	x	x		8	0.00 €	x	5	Completata	2	2	1	0.00 €
Project 19	x						3	0.00 €	x	1	Completata	2	20	1	0.00 €
Project 20	x	x	x	x	x	x	8	0.00 €	x	6	Completata	1	26	1	0.00 €

Figura 5.5. Struttura database

5.4 Implementazione dei modelli

Per la fase di implementazione dei modelli è stato utilizzato il software *Anaconda*, un distributore di linguaggi di programmazione Python ed R adottato per la semplificazione del calcolo scientifico come data science, analisi predittiva, elaborazione di dati su larga scala e machine learning. Nello specifico l'ambiente interattivo Jupyter ha consentito lo sviluppo di codici in grado di vagliare ed elaborare il database a disposizione.

A seguito del caricamento della base dati illustrata in figura 5.5 è possibile ottenere una prima matrice di correlazione; questa tipologia di misura statistica esprime la relazione tra due variabili ed indica la tendenza che esse hanno a variare insieme, ovvero, a "covariare". Per esprimere il legame esistente tra due variabili, in termini di entità e direzione, si utilizza il coefficiente di correlazione. Tale coefficiente è standardizzato e può assumere valori che vanno da -1.00 (correlazione perfetta negativa) a $+1.00$ (correlazione perfetta positiva).

- Coefficiente di correlazione > 0 indica una correlazione positiva tra le due variabili in analisi; all'aumentare del valore di una, si assiste all'incremento dell'altra;
- Coefficiente di correlazione $= 0$ evidenzia una sostanziale mancanza di correlazione tra i due fattori;
- Coefficiente di correlazione < 0 dimostra la presenza di una correlazione negativa; il trend delle due variabili risulta, quindi, opposto: alla diminuzione del valore di una variabile, si assiste all'aumento dell'altra.

In particolare vi possono essere tre diverse entità di correlazione:

- Debole: quando il valore assoluto del coefficiente di correlazione è compreso tra 0 e 0.3;
- Moderata: quando il valore assoluto si pone nell'intervallo 0.3 - 0.7;
- Forte: quando il valore assoluto di tale indicatore è maggiore di 0.7.

Prima di procedere con l'analisi dei risultati ottenuti, è doveroso premettere che i valori dei coefficienti di correlazione sono, ovviamente, influenzati e determinati dalla numerosità del campione preso in esame; non è da escludere che una differente basi dati possa far emergere risultati diversificati. In figura 5.6 è riportata la matrice di correlazione ottenuta in prima battuta.

	FTE Performance	ACWP	Duration	FTE Gruppo	Complessità progetto	Numerosità defect	F&F	Costi Follow Up
FTE Performance	1.000000	-0.324685	0.014677	-0.141577	-0.705408	-0.437506	-0.673390	-0.907031
ACWP	-0.324685	1.000000	-0.101309	0.413469	0.232551	0.152283	0.252166	0.304689
Duration	0.014677	-0.101309	1.000000	-0.216513	-0.008168	-0.059449	-0.004736	-0.033729
FTE Gruppo	-0.141577	0.413469	-0.216513	1.000000	0.128359	0.070750	0.146474	0.172801
Complessità progetto	-0.705408	0.232551	-0.008168	0.128359	1.000000	0.609125	0.883584	0.877531
Numerosità defect	-0.437506	0.152283	-0.059449	0.070750	0.609125	1.000000	0.525455	0.534343
F&F	-0.673390	0.252166	-0.004736	0.146474	0.883584	0.525455	1.000000	0.820080
Costi Follow Up	-0.907031	0.304689	-0.033729	0.172801	0.877531	0.534343	0.820080	1.000000

Figura 5.6. Matrice di correlazione

Da una prima analisi è possibile evidenziare la presenza di variabili particolarmente correlate tra loro: FTE Performance risulta inversamente e fortemente correlata alla Complessità del progetto e ai Costi di follow up; queste due relazioni possono essere l'esemplificazione del ragionamento secondo cui più i programmatori risultano essere performanti, meno saranno i costi da sostenere in fase di garanzia per i bug individuati nei progetti; allo stesso modo più complessa si dimostra essere l'attività, tendenzialmente inferiori riusciranno ad essere le performance degli sviluppatori.

L'indicatore della Numerosità dei defect sembra avere una relazione coerente con l'andamento del rilevatore della Complessità del progetto e della variabile target, entrambi caratterizzati da un'entità moderata; questi legami esplicano coerentemente la possibilità del verificarsi di numerosi defect nel caso progetti complessi, e l'eventualità di dover sostenere maggiori costi di follow up qualora il progetto presenti numerose anomalie in fase di testing.

La variabile Family & Friends, infine, mostra un legame forte e concorde con le variabili Complessità del progetto e Costi di follow up. La spiegazione del primo aspetto può essere facilmente ricollegabile a quanto spiegato precedentemente, ovvero all'esigenza secondo cui i progetti che presentano particolari complessità e richiedono, quindi, specifiche attenzioni, rientrano nella categoria dei F&F; per quanto

concerne la seconda relazione è plausibile presupporre che progetti spinosi abbiano poi ripercussioni anche sulle risorse da impiegare in fase di prima manutenzione.

La matrice di correlazione viene spesso raffigurata graficamente con una heat map, sfruttando l'intensità di colore come segnalatore di forti relazioni (figura 5.7).

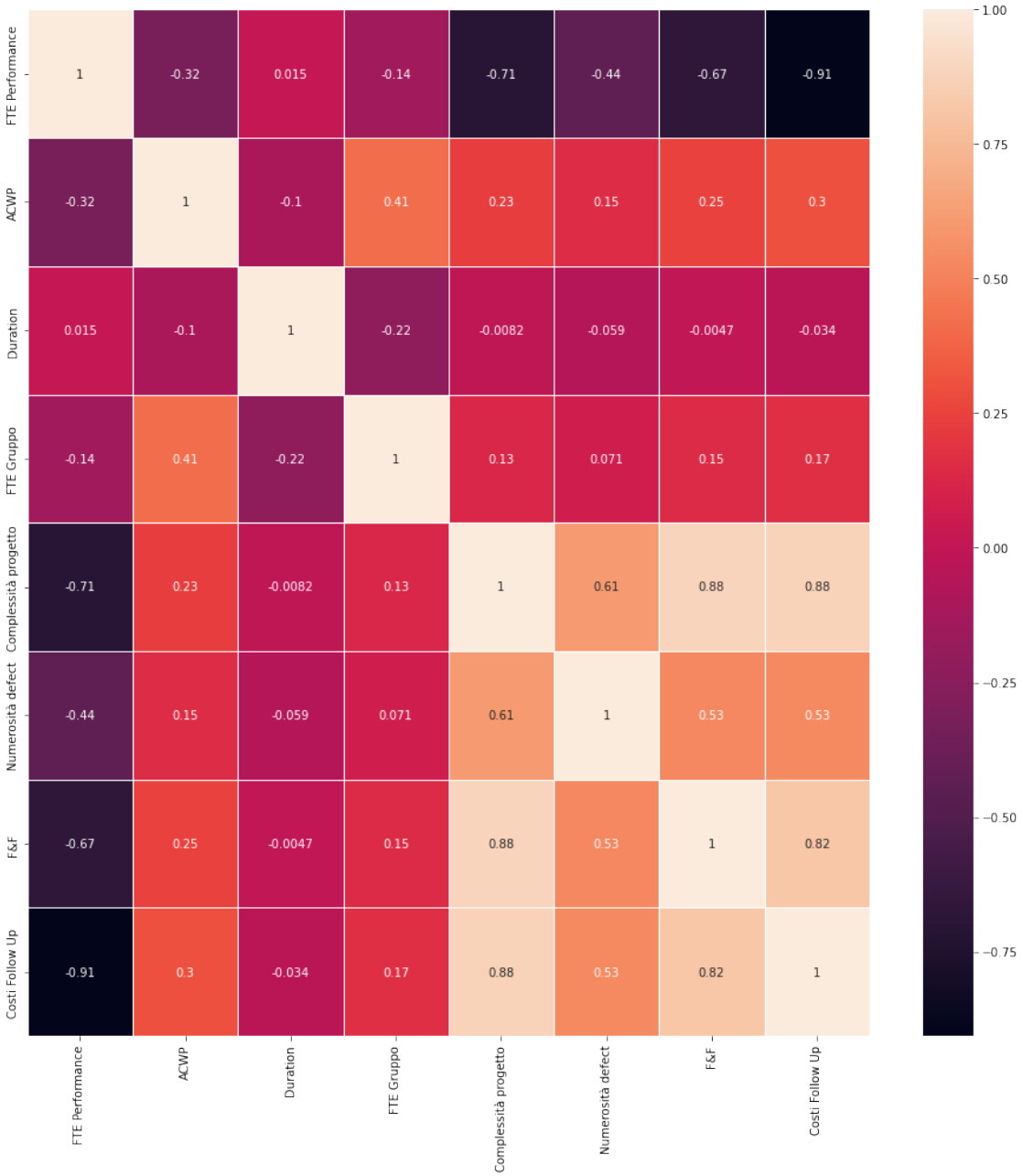

Figura 5.7. Heat map

Questa prima indagine rappresenta solo un primo step parziale e poco preciso; è necessario, infatti, affinare la base dati per ottenere

risultati che siano affidabili e strutturati in maniera accurata.

Per uno studio attento e valido è necessario procedere con l'identificazione degli outliers, valori che, risultando anomali rispetto alla totalità delle osservazioni effettuate, potrebbero influire negativamente sull'analisi generale. Si è deciso di rifarsi al metodo di Hidiroglou e Berthelot, basato sull'identificazione dei quantili: in particolare si è stabilito di tracciare l'intervallo di accettazione come lo spazio dei valori compresi tra il quantile q_{10} e q_{90}, il giusto compromesso tra l'intento di esclusione di valori atipici, e la selezione di un'area di studio sufficientemente ampia. Visto il contenuto numero di osservazioni del database, si è scelto di impostare gli outliers individuati pari agli estremi dell'intervallo di accettazione, in modo da farli rientrare nell'intervallo ma posti al limite della validità consentita.

A seguito della puntualizzazione dell'intervallo di accettazione, si ripete la creazione della matrice di correlazione, notando l'avvenuta accentuazione di alcune relazioni come quelle che intercorrono tra le variabili F&F, Complessità progetto e Numerosità defect con la variabile target Costi Follow up. Come è evidente dalla figura 5.8 non emergono nuove relazioni ma si intensificano quelle analizzate in precedenza.

	FTE Performance	ACWP	Duration	FTE Gruppo	Complessità progetto	Numerosità defect	F&F	Costi Follow Up
FTE Performance	1.000000	-0.473301	-0.005859	-0.154467	-0.467097	-0.407071	-0.528639	-0.599104
ACWP	-0.473301	1.000000	0.156224	0.396642	0.312650	0.287772	0.335218	0.367990
Duration	-0.005859	0.156224	1.000000	0.073100	0.002357	0.005587	-0.014604	-0.014084
FTE Gruppo	-0.154467	0.396642	0.073100	1.000000	0.208253	0.229703	0.216388	0.206493
Complessità progetto	-0.467097	0.312650	0.002357	0.208253	1.000000	0.945728	0.883584	0.939255
Numerosità defect	-0.407071	0.287772	0.005587	0.229703	0.945728	1.000000	0.770980	0.905115
F&F	-0.528639	0.335218	-0.014604	0.216388	0.883584	0.770980	1.000000	0.846012
Costi Follow Up	-0.599104	0.367990	-0.014084	0.206493	0.939255	0.905115	0.846012	1.000000

Figura 5.8. Matrice di correlazione post eliminazione outliers

Tuttavia, la correlazione non solo non è in grado di verificare la presenza o l'effetto di altre variabili che non siano le due prese in esame, ma interpreta limitatamente le relazioni senza investigarne la tipologia. Per tale motivo si procede con l'approfondimento dell'analisi.

Al fine di perfezionare quanto si andrà a vagliare, è necessario escludere dallo studio i fattori che poco risultano incidere nell'analisi: per

tale motivo si decide di eliminare le variabili ACWP, FTE Gruppo e Duration che, contrariamente a quanto si potesse ipotizzare inizialmente, non risultano particolarmente correlate con alcuna variabile; i costi sostenuti, la durata dell'attività e il numero di risorse coinvolte nel suo sviluppo, non solo non implicano particolari relazioni di causa-effetto con le problematiche nate post produzione, ma non presentano alcuna dipendenza con nessun fattore esaminato.

Per proseguire l'indagine si imposta esplicitamente la variabile Costi follow up come variabile target e si suddivide il database, in maniera randomica, in *Training Set* e *Test Set*; si stabilisce, così, che l'80% della base dati verrà utilizzata per addestrare l'algoritmo, e il restante 20% sarà sfruttato per testare quanto appreso.

Di seguito verranno presentati i risultati ottenuti applicando i modelli di Regressione lineare multipla, Regressione polinomiale multipla, Albero decisionale e Foresta Casuale; si analizzeranno, nello specifico, gli indicatori chiave *Accuracy* e *RMSE* (Root Mean Square Error) per i rispettivi training e testing set. L'Accuracy, formalmente calcolata come frazione delle previsioni che il modello ha ottenuto correttamente, indica la precisione della predizione come percentuale compresa tra 0 e 1; più questo valore si avvicina ad 1, migliore è la prestazione del modello. Il RMSE rappresenta l'errore medio della predizione, è un metodo standard per misurare l'imprecisione di un modello nella previsione di dati quantitativi; ha il vantaggio di essere facilmente interpretabile, infatti, i valori assunti dal RMSE non sono quantità a-dimensionali, ma assumono l'unità di misura del fenomeno considerato. I valori di Accuracy emersi dal Training Set e dal Test Set dovrebbero essere simili tra loro. Nel caso in cui l'Accuracy del Test Set si rilevasse nettamente inferiore a quella ottenuta dal Training Set, sarebbe necessario valutare la presenza di *overfitting*; questo fenomeno, caratteristico del machine learning, si verifica quando un modello si adatta esattamente ed eccessivamente ai suoi dati di training usati per l'addestramento e, conseguentemente, non si dimostra valido per la stima accurata dei dati di test. In altre parole, nel caso in cui si manifesti tale anomalia, il modello ha semplicemente memorizzato modelli e rumore specifici nei dati di training, ma non è sufficientemente flessibile per eseguire stime su nuovi dati. Per quanto concerne la metrica del Root Mean Square Error, è la deviazione standard dei residui, o errori di previsione, che sono una misura della distanza dai punti dati della linea di regressione. RMSE valuta quanto sono distribuiti questi residui, fornendo una valutazione

quantitativa della concentrazione dei dati attorno alla linea di miglior adattamento. Questo indicatore valuta, pertanto, la bontà di adattamento del modello; nel caso di buon livello di fit tra modelli e base dati, il RMSE si abbassa e presenta valori simili tra addestramento e test.

5.4.1 Regressione lineare multipla

Il primo modello implementato è la regressione lineare multipla, nota anche semplicemente come regressione multipla. Questa è una tecnica statistica che utilizza diverse variabili esplicative per prevedere il risultato di una variabile di risposta. L'obiettivo della regressione lineare multipla è modellare la relazione lineare tra le variabili indipendenti e la variabile dipendente. La formula della retta di regressione è così composta:

$$Y_i = \beta_0 + \beta_1 X_{1i} + \beta_2 X_{2i} + ... + \beta_k X_{ki} + \epsilon$$

dove Y è la variabile dipendente, X_i sono le variabili indipendenti e β_i i rispettivi coefficienti angolari, β_0 è l'intercetta della retta, ovvero il valore atteso di Y quando tutte le X sono uguali a 0, ed infine ϵ rappresenta l'errore statistico.

I coefficienti dell'equazione risultante dall'applicazione di questo modello al caso di studio in esame sono i seguenti:

Variabile	**Coefficiente**
Intercetta	758.9976
FTE Performance	-474.5760
Complessità progetto	4543.6633
Numerosità defect	5.5898
F&F	407.6207

Coerentemente con quanto si potesse ipotizzare inizialmente, e con quanto analizzato precedentemente, i coefficienti che esplicitano le relazioni tra la variabile target Costi di follow up e gli altri fattori presi in considerazione confermano la presenza dell'andamento di alcune relazioni. L'unico coefficiente negativo, indice di un andamento discorde, risulta essere quello della variabile FTE performance, in linea con la riflessione secondo cui più le performance del gruppo di lavoro sono elevate, minori saranno i costi da sostenere in una prima fase

manutentiva. Al contrario, il modello presenta coefficienti della retta positivi per i fattori Complessità del progetto, Numerosità defect, F&F, che si muovono conformemente alla variabile target; più ingente sarà la complessità del progetto, più numerosi saranno i defect riscontrati in fase di test, più il progetto sarà legato al modulo family and friends, più i costi in discussione aumenteranno.

Nella tabella che segue sono riportati i valori di Accuracy e RMSE relativi al training e test set ottenuti con l'utilizzo del modello di regressione lineare multipla.

Metrica	**Risultato**
Training Accuracy	0.9233
Testing Accuracy	0.8826
Training RMSE	378.1679
Testing RSME	523.1659

I valori di Accuracy, vicini all'1, sembrano essere alti e abbastanza simili tra loro, segno di una buona precisione del modello e di una valida corrispondenza tra i dati ottenuti per testare la metodologia e quelli utilizzati per l'addestramento dell'algoritmo. Anche i valori del RMSE evidenziano risultati positivi: oltre ad essere abbastanza simili tra loro, si presentano anche relativamente bassi. Va ricordato, infatti, che il root mean square error si esprime nell'unità di misura in questione, nello specifico nell'unità di misura dei costi di follow up, e va rapportato ad un intervallo che copre da 0 ad approssimativamente 16500 euro. Ottenere, pertanto, un RMSE pari a 378 significa aver conseguito un errore particolarmente basso: effettuando la normalizzazione di tale fattore, ricavata dividendo il RMSE per l'intervallo in cui varia la variabile target, si consegue un errore del circa 2,30% nel caso del training, e del 3,17% nel caso del testing.

5.4.2 Regressione polinomiale multipla

La regressione polinomiale è una forma di analisi di regressione in cui la relazione tra la variabile indipendente X e la variabile dipendente Y è modellata come un polinomio di n - esimo grado in X, nel caso in esame di terzo grado. Questo tipo di regressione si adatta ad una relazione non lineare tra il valore di X e la corrispondente media condizionale di Y, indicata con E(Y | X). L'espressione che generalizza questo modello è la seguente:

$$Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + ... + \beta_n X^n + \epsilon$$

Si espongono nella tabella seguente i risultati ricavati dalla prova effettuata con il modello di regressione polinomiale multipla di terzo grado:

Metrica	**Risultato**
Training Accuracy	0.9686
Testing Accuracy	0.7609
Training RMSE	241.9602
Testing RSME	746.5804

In questo scenario l'Accuracy legata alla fase di training sembra essere ottimale, in quanto parrebbe spiegare i dati ed effettuare previsioni con un altissimo livello di precisione, di circa il 97%. Tuttavia, analizzando la risposta estratta dal testing del processo, si evince una netta ed ingente differenza tra i due valori: testing accuracy non supera la soglia del 76%; questo sostanziale dislivello, che si conferma esaminando il root mean square error, fa emergere la possibile presenza del fenomeno dell'overfitting. È, infatti, possibile che questo modello si sia adattato eccessivamente ai dati di training senza riuscire ad essere sufficientemente accurato e valido nelle successive fasi di testing. Anche il RMSE che sembra minimo nella fase di addestramento, subisce un cospicuo incremento quando si testa il modello, raggiungendo la percentuale del 4,5.

5.4.3 Albero decisionale

L'albero decisionale è uno strumento di modellazione predittiva generale che ha applicazioni che abbracciano diverse aree. In generale, gli alberi decisionali sono costruiti tramite un approccio algoritmico che identifica i modi per dividere un set di dati in base a condizioni diverse. È strutturato come un grafo, dove i nodi interni rappresentano le caratteristiche di un set di dati, i rami simboleggiano le regole decisionali e ogni nodo foglia equivale ad un risultato. Rappresenta uno dei metodi più utilizzati e pratici per l'apprendimento supervisionato, sia per la classificazione che per la regressione, a seconda che l'output previsto sia una variabile categorica o continua. L'obiettivo è creare un modello che preveda il valore di una variabile target apprendendo semplici regole decisionali dedotte dalle caratteristiche

dei dati; tali regole sono, generalmente, sotto forma di dichiarazioni se-allora-altro.

L'approccio alla soluzione con l'applicazione dell'albero decisionale, impostando una profondità massima pari a 5, ha condotto ai seguenti risultati:

Metrica	**Risultato**
Training Accuracy	0.9722
Testing Accuracy	0.9094
Training RMSE	227.7391
Testing RSME	459.5050

La casistica al vaglio, presenta ottimi risultati: l'Accuracy dell'addestramento concretizza una precisione che supera il 97%, mentre l'errore legato al training non oltrepassa la soglia del 1,39%. Sintomo della rivelazione di un modello ottimale, che ben fitta i dati, è la conferma di tali risultati anche nella fase di test; i valori di testing Accuracy e testing RMSE, rispettivamente pari al 91% e 2,78%, che si discostano solo lievemente dalle cifre sopraindicate, corroborano la tangibile validità della metodologia utilizzata, valutandone l'effettiva applicazione.

5.4.4 Foresta casuale

La foresta casuale è un algoritmo di apprendimento supervisionato, utilizzato sia per problemi di classificazione che di regressione. La "foresta" è costituita da un insieme di alberi decisionali, solitamente addestrati con il metodo del "bagging"; tale processo, noto anche come Bootstrap Aggregation, sceglie un campione casuale dal set di dati, con cui genera un modello che viene addestrato in modo indipendente e fornisce i propri risultati. L'output finale si ottiene a seguito della combinazione dei risultati di tutti i modelli e del calcolo della loro media, in modo tale da aggiungere ulteriore casualità all'algoritmo. Al crescere della numerosità degli alberi, infatti, invece di cercare la caratteristica più importante mentre si divide un nodo, cerca la caratteristica migliore in un sottoinsieme casuale di caratteristiche. Ciò si traduce in un'ampia diversità che generalmente implica un modello preferibile.

Nella tabella seguente si mostrano gli esiti dell'utilizzo della foresta casuale, ottenuti dopo numerosi tentativi, fino ad impostare un n-estimator, ovvero un numero di alberi decisionali utilizzati, pari a 400:

Metrica	**Risultato**
Training Accuracy	0.9641
Testing Accuracy	0.9008
Training RMSE	258.8541
Testing RSME	480.9739

L'algoritmo in questione evidenzia conclusioni non molto divergenti da quelle ottenute in 5.4.3. La foresta casuale sembra, infatti, ripercorrere e confermare quanto emerso dall'utilizzo dell'albero decisionale, coerentemente con la sua struttura basata su più alberi; tuttavia, non appare, come si poteva ipotizzare in un primo momento, distintamente migliore. La foresta casuale rinnova, infatti, un'accuratezza del 96,4% nella fase di apprendimento e del 90% nei test effettuati, ed un errore leggermente superiore a quello ottenuto con l'albero decisionale, pari, rispettivamente, al 1,56% e 2,91%. Anche questo modello, pertanto, si dimostra particolarmente valido per la previsione della variabile target analizzata.

5.5 Analisi dei risultati

Nei paragrafi precedenti sono stati esaminati ed approfonditi diversi modelli implementati affinché, attraverso i meccanismi di machine learning, potessero condurre ad una previsione dei costi di follow up il più precisa possibile. Nella tabella che segue si riepilogano i risultati ottenuti al fine di consentirne un confronto:

	Training Accuracy	Testing Accuracy	Training RMSE	Testing RMSE
Multiple Linear Regression	0.923315	0.882604	378.167936	523.165917
Polynomial Regression	0.968607	0.760929	241.960243	746.580457
Decision Tree Regression	0.972189	0.909436	227.739125	459.505026
Random Forest Regression	0.964070	0.900776	258.854137	480.973929

Figura 5.9. Risultati ottenuti con i 4 modelli

Ancora più immediata è la comparazione effettuata attraverso il grafico in figura 5.10:

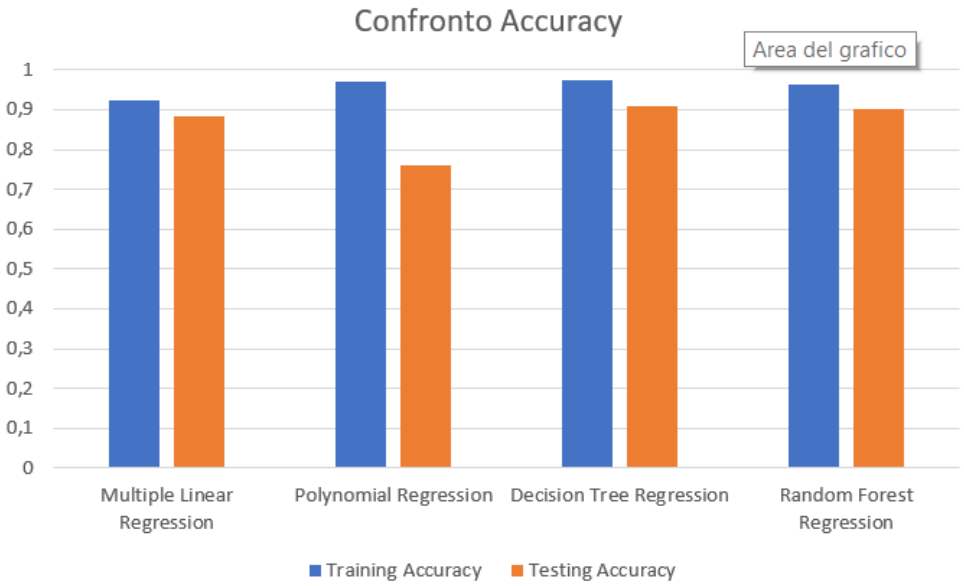

Figura 5.10. Confronto tra Training Accuracy e Testing Accuracy

Dal barchart sopra rappresentato è evidente che il primo modello da escludere sia la regressione polinomiale che, seppur con ottimi risultati raggiunti in fase di addestramento, non li conferma in fase di test, rendendo lecito avanzare l'ipotesi della presenza di overfitting. La non affidabilità del modello si conferma anche paragonando i valori ricavati dal RMSE, come chiarificato nel grafico 5.11

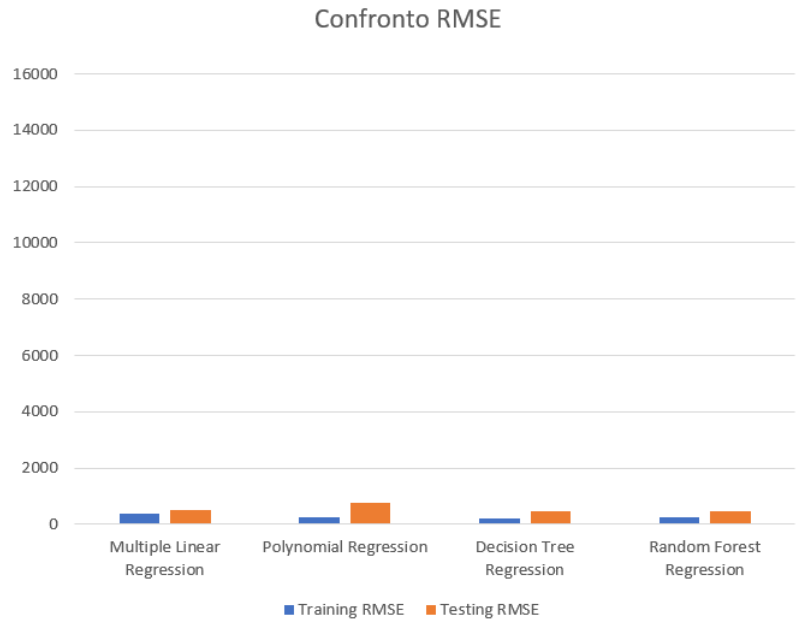

Figura 5.11. Confronto tra Training RMSE e Testing RMSE

Nonostante tutti gli errori siano minimi, infatti, la regressione polinomiale presenta il valore di Root Mean Square Error maggiore.

Avendo respinto la polynomial regression, l'attenzione si sposta sui tre modelli restanti: la regressione lineare, l'albero decisionale e la foresta casuale. Il primo dei tre, nonostante presenti la differenza minore tra Training Accuracy e Testing Accuracy, presenta un livello di performance inferiore se paragonato al decision tree e al random forest, rispetto ai quali mostra un valore di errore maggiore in entrambe le fasi. Si riporta di seguito la simulazione dell'applicazione della regressione lineare, paragonando i risultati attesi con quelli ottenuti dalla previsione effettuata dal modello.

	Costi Follow Up - Realtà	Costi Follow Up - Previsione
0	1691.2295	1696.107061
1	16524.3650	12492.366487
2	1691.2295	1690.517221
3	1691.2295	1690.517221

Figura 5.12. Simulazione Regressione Lineare

Nella figura 5.12 si mostrano, a titolo di esempio, i primi 4 risultati ottenuti eseguendo la regressione lineare. È evidente che, sebbene minimi, siano presenti errori, e nessuna previsione sia totalmente corretta. Per questo motivo si decide di scartare anche tale modello.

In ultima analisi, i modelli rimanenti e migliori risultano essere l'albero decisionale e la foresta casuale. Entrambi, come mostrato in precedenza, conseguono ottimi risultati: un elevato livello di accuratezza ed un errore trascurabile. Le verifiche realizzate simulando le previsioni confermano la validità dei modelli.

	Costi Follow Up - Realtà	Costi Follow Up - Previsione
0	1691.2295	1691.2295
1	16524.3650	14293.5700
2	1691.2295	1691.2295
3	1691.2295	1691.2295

Figura 5.13. Simulazione Albero Decisionale

	Costi Follow Up - Realtà	Costi Follow Up - Previsione
0	1691.2295	1691.22950
1	16524.3650	13311.11156
2	1691.2295	1691.22950
3	1691.2295	1691.22950

Figura 5.14. Simulazione Foresta Casuale

Nelle figure 5.13 e 5.14 sono riportati, rispettivamente, i risultati ricavati dalla previsione effettuata con il decision tree e con il random forest. Entrambi conseguono risposte estremamente accurate e precise, riuscendo ad ottenere in maniera ottimale la predizione dei costi di follow up. È importante sottolineare il verificarsi di una particolare esattezza nei casi in cui i valori dei costi si presentano relativamente bassi: questo è dovuto alla struttura del database che propone uno storico di osservazioni con costi di follow up relativamente ridotti se paragonati ai costi dei progetti; l'algoritmo è riuscito, infatti, ad allenarsi maggiormente sulle casistiche con valori bassi piuttosto che su quelle con costi più elevati, guadagnando, così, una capacità predittiva praticamente impeccabile sulla stima dei costi più contenuti.

Albero decisionale e foresta casuale sono, dunque, i modelli più opportuni; effettuando un confronto tra i due, sulla base delle metriche analizzate finora e delle previsioni realizzate, il decision tree sembra essere il modello che meglio si adatta all'elaborazione del database in questione e al raggiungimento dello scopo ultimo. È, infatti, l'algoritmo che propone il più elevato valore di Accuracy e la minor percentuale di errore sia in fase di addestramento che di testing.

L'approfondimento proposto conduce, pertanto, alla scelta del modello dell'albero decisionale come migliore metodologia per la previsione della variabile oggetto di studio; con la consapevolezza della necessità e della possibilità di acuire le analisi svolte, si può ritenere di essere giunti, sulla base del database a disposizione e delle analisi effettuate, all'individuazione di un modello di previsione ottimale utile alla realizzazione della stima desiderata.

Capitolo 6

Conclusioni

L'esecuzione di un progetto software rimane una tra le più complesse e diversificate realizzazioni di prodotti che vi siano. Lo sviluppo di un progetto IT è intrinsecamente soggetto a mutamenti legati alle caratteristiche del prodotto stesso che tendono a cambiare durante il naturale susseguirsi delle fasi di sviluppo; nella realtà è possibile, infatti, considerare una consuetudine la richiesta di apportare modifiche ai requisiti in corso d'opera, modificando, così, la complessità del programma e rendendone ancora più difficoltosa la gestione e pianificazione iniziale. Il settore tecnologico, inoltre, subisce continue innovazioni: sebbene ci si basi spesso sulle analogie con i progetti precedenti, in fase iniziale di stima progettuale è importante considerare le possibili trasformazioni dei tools, delle tecnologie e delle caratteristiche a cui i progetti possono esser stati sottoposti, valutando l'effettiva affidabilità della comparazione e dell'analogia del prodotto da sviluppare con quelli realizzati in passato; con il decorrere del tempo risulta, quindi, necessario, analizzare l'attendibilità del bagaglio di esperienze da cui attingere per qualsivoglia valutazione.

Spesso la credibilità di una stima è fortemente legata alla trasparenza del metodo utilizzato, dei dati, delle assunzioni predisposte inizialmente, per questo motivo l'applicazione del machine learning per la realizzazione di stime economiche e temporali attraverso l'elaborazione di modelli predittivi rappresenta, sicuramente, una novità che sta attirando l'attenzione di molti ricercatori ed esperti del mestiere. Nella scelta della metodologia, inoltre, influiscono fortemente le decisioni del Project Manager in carico e, più ingenerale, le strutture funzionali dell'azienda.

Al netto delle considerazioni da effettuare, e delle evoluzioni che possano emergere in futuro, lo studio presentato ha voluto essere un

primo passo verso l'utilizzo di metodi intelligenti e moderni per investigare temi estremamente attuali e centrali nell'ambito del Project Management. In particolare, si è tentato di applicare questo processo ad una peculiare fase dello sviluppo del progetto software, la fase di *follow up*, oggi sottovalutata e inesplorata sia a livello dottrinale che aziendale. L'auspicio è che il presente lavoro di tesi, congiunto ad ulteriori approfondimenti, possa rappresentare un aiuto concreto per l'azienda e il team in cui è nata l'idea, migliorando ed automatizzando, in parte, il processo di elaborazione delle stime progettuali, e fornendo una soluzione al gap gestionale della fase in questione. Gli studi futuri che si potranno condurre potrebbero far emergere nuovi fattori chiave, indicatori o relazioni che consentiranno la riduzione degli errori legati alla previsione e il conseguente perfezionamento della tecnica predittiva.

Bibliografia

[1] Project Management Institute, What is Project Management?, https://www.pmi.org/about/learn-about-pmi/what-is-project-management

[2] Weaver, P. (2010). Understanding programs and projects—oh, there's a difference! Paper presented at PMI® Global Congress 2010—Asia Pacific, Melbourne, Victoria, Australia. Newtown Square, PA: Project Management Institute

[3] HumanWare, Project Manager Center, Vincoli di progetto: il triangolo del triplice vincolo, https://www.humanwareonline.com/project-management/center/vincoli-di-progetto-il-triangolo-del-triplice-vincolo/

[4] SCRUMstudy, A Guide to the Scrum Body of Knowledge (SBOK™ Guide), 2016

[5] S. Rohollah Moosavi Tayebi, S. Shervin Ostadzadeh and S. Mazaheri, "Designing enterprise operating system with PRINCE2 framework," 2010 International Conference on Computer and Communication Technologies in Agriculture Engineering, 2010, pp. 479-482, DOI 10.1109/CCTAE.2010.5544345

[6] Giovanni Massimo Benedetti, "Application Lifecycle Management, Il processo Hybrid fra Waterfall ed Agile"

[7] https://www.rasmussen.edu/degrees/technology/blog/what-is-information-technology/

[8] S. Liu e C. Liu, "Management Innovation of IT Project Managers", 2010 3rd International Conference on Information Management, Innovation Management and Industrial Engineering , 2010, pp. 62-65, DOI 10.1109/ICIII.2010.337

[9] LA Belady e MM Lehman, "A Model of Large Program Development", IBM Systems Journal , vol. 3, pp. 225-252, 1976.

[10] Software Maintenance Costs, Ó Jussi Koskinen, Information Technology Research Institute, ELTIS-project, University of Jyväskylä, Finland,

[11] Kearney, J. et al. Software complexity measurement. Commun. ACM 29, 11 (1986), 1044-1050
[12] Gibson, V.R. and Senn, J.A. System structure and software maintenance performance. Commun. ACM 32, 3 (1989), 347-358.
[13] Card, D.N., Page, G.T. and McGarry, F.E. Criteria for software modularization. In Proceedings of the Eighth International Conference on Software Engineering (1985), pp. 372-377
[14] Floris P, Vogt Harald H. Come risparmiare sui costi di manutenzione del software, Omnext white paper , SOURCE 2 VALUE , 2010
[15] Gibson, VR e Senn, JA Struttura del sistema e prestazioni di manutenzione del software. Comune. ACM 32, 3 (1989), 347-358.
[16] Dehaghani, Sayed Mehdi Hejazi e Nafiseh Hajrahimi. "Quali fattori influenzano i costi di manutenzione dei progetti software?" Acta informatica medica : AIM : rivista della Società per l'informatica medica della Bosnia ed Erzegovina : casopis Drustva za medicinsku informatiku BiH vol. 21,1 (2013): 63-6. DOI 10.5455/AIM.2012.21.63-66
[17] C. Ebert e H. Soubra, "Tecnologie di stima delle dimensioni funzionali per la manutenzione del software", in IEEE Software , vol. 31, n. 6, pp. 24-29, nov.-dic. 2014, DOI 10.1109/MS.2014.138
[18] E. Barry, "Evoluzione del software, volatilità e modelli di manutenzione del ciclo di vita: una sinossi dell'analisi longitudinale", Conferenza internazionale sulla manutenzione del software, 2002. Atti. , 2002, pp. 474-477, DOI 10.1109/ICSM.2002.1167806
[19] J. Amrutha e AS Remya Ajai, "Performance analysis of Backpropagation Algorithm of Artificial Neural Networks in Verilog", 2018 3a IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT) , 2018, pp. 1547-1550, DOI 10.1109/RTEICT42901.2018.901261

Sitografia

http://www.pmconsult.it/cenni-storici.html
http://checksound.gitbook.io/gestione/progettazione#pmbook
http://www.jest.it/project-risk-management/il-ciclo-di-vita-di-un-progetto/
https://opentextbc.ca/projectmanagement/chapter/chapter-3-the-project-life-cycle-phases-project-management/

https://www.invensislearning.com/blog/5-phases-project-management-lifecycle/
https://asana.com/it/resources/project-management-triangle
https://digitalinnovationhub.org/metodi-waterfall-e-agile-7c97dcfea54c
https://www.agileway.it/manifesto-per-lo-sviluppo-agile-di-software/
https://www.pmi.it/tag/information-technology
https://www.smartsheet.com/content-center/best-practices/project-management/project-management-guide/project-management-IT
https://www.techtarget.com/searchcio/definition/IT-project-management
http://asq.org/public/wqm/how-to-save-on-software-maintenance-costs.pdf
https://www.sciencedirect.com/science/article/pii/S1319157818313077?pes=vor#b0030
https://www.siliconrepublic.com/enterprise/white-box-machine-learning
https://www.machine-learning.it/regressione-lineare/
https://ischoolonline.berkeley.edu/blog/what-is-machine-learning/
https://www.oracle.com/it/data-science/machine-learning/what-is-machine-learning/
https://towardsdatascience.com/machine-learning-interpretability-techniques-662c723454f3
https://medium.com/hackernoon/deep-learning-feedforward-neural-networks-explained-c34ae3f084f1
https://www.sciencedirect.com/topics/chemical-engineering/feedforward-neural-networks
https://www.sciencedirect.com/topics/computer-science/backpropagation-algorithm
https://medium.com/@tam.tamanna18/comparing-black-box-vs-white-box-modeling-bd01575b7670
https://www.ibm.com/cloud/learn/machine-learning
https://www.ibm.com/cloud/blog/ai-vs-machine-learning-vs-deep-learning-vs-neural-networks
https://www.siliconrepublic.com/enterprise/white-box-machine-learning
https://builtin.com/data-science/random-forest-algorithm

https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm
https://www.section.io/engineering-education/introduction-to-random-forest-in-machine-learning/
https://corporatefinanceinstitute.com/resources/knowledge/other/multiple-linear-regression/