Alma Mater Studiorum · Università di Bologna

SCUOLA DI INGEGNERIA E ARCHITETTURA
Corso di Laurea Magistrale in Ingegneria e Scienze Informatiche

Analisi Comparativa tra Metodologie Tradizionali e Agili con Enfasi su Scrum nel Contesto dello Sviluppo Software

Relatore:
Chiar.mo Prof. MARCO ANTONIO BOSCHETTI

Presentata da:
DIEGO SOZIO

Anno Accademico
2024-2025

Nobody believes in you.

You've lost again, and again, and again.

The lights are cut off, but you still are looking at your dream, reviewing it every day and say to yourself,

It's not over Until I win.

– *Les Brown*

Indice

Introduzione

L'evoluzione costante e la crescente complessità delle applicazioni software hanno posto l'accento sull'importanza di metodologie di sviluppo efficaci ed efficienti.

Nel vasto panorama dello sviluppo di soluzioni software, due approcci principali hanno dominato il dibattito:

1. le metodologie tradizionali
2. le metodologie Agili

Le metodologie tradizionali, come il modello waterfall, hanno a lungo rappresentato un pilastro nell'ambito dello sviluppo software, con il loro approccio sequenziale e ben strutturato.

Tuttavia, l'emergere delle metodologie Agili ha portato un vento di cambiamento, ponendo l'accento sulla flessibilità, la collaborazione e la risposta rapida ai cambiamenti.

Nell'ambito delle metodologie Agili, Scrum ha guadagnato una notevole popolarità grazie alla sua enfasi sulla flessibilità, la comunicazione continua, la pianificazione iterativa e il coinvolgimento attivo del cliente nel processo di sviluppo.

Questa tesi si propone di condurre un'analisi dettagliata delle metodologie tradizionali e Agili, con uno specifico approfondimento su Scrum, nel contesto dello sviluppo software.

Attraverso la valutazione delle caratteristiche, dei vantaggi e degli svantaggi di entrambi gli approcci, questo studio mira a fornire una prospettiva chiara per comprendere quale metodologia sia più adatta in determinati contesti e per affrontare specifiche sfide nell'ambito dello sviluppo software.

L'obiettivo principale è quello di delineare un quadro esaustivo che possa supportare le organizzazioni, gli sviluppatori e gli stakeholder nel prendere decisioni informate riguardo alla scelta delle metodologie di sviluppo, considerando il contesto specifico e le esigenze del progetto.

Capitolo 1

Project Management

1.1 Definizione di Progetto

Prima di poter definire quello che è il Project Management risulta necessario definire l'elemento principale di questa disciplina, il **Progetto**.

Un progetto è ciò che nasce in risposta all'esigenza di soddisfare un bisogno di un singolo o di più persone.

In ambito aziendale, l'esigenza da cui parte il progetto è relativa a un'organizzazione composta da persone che vogliono raggiungere un obiettivo strategico condiviso.
Volendo fornire una definizione formale di Progetto possiamo dire che un progetto è:

"Un'iniziativa temporanea intrapresa per creare un prodotto, un servizio o un altro risultato con caratteristiche di unicità" [3]

Da questa definizione possiamo evincere che un progetto si compone di due dimensioni:

1. **Temporaneità**: il progetto è caratterizzato da un inizio e da una fine. La fine si raggiunge quando sono stati ottenuti gli obiettivi del progetto o quando il progetto è terminato perché non si riesce a raggiungere tali obiettivi (anche perché non più possibile) o quando non sussiste più l'esigenza del progetto.

   Va sottolineato che il termine temporaneo non indica necessariamente una breve durata e non si applica generalmente al prodotto, servizio o altro risultato creato dal progetto.

2. **Unicità**: il prodotto, il servizio o il risultato finale del progetto che ha caratteristiche di unicità perchè non è mai stato ottenuto in precedenza oppure, nel caso

di progetti ricorrenti, ha comunque delle caratteristiche che lo differenziano dai progetti precedenti.

Da queste caratteristiche si evince che l'attività di gestione di un progetto è fondamentale per la buona riuscita, nasce quindi il concetto di **Project Management** che secondo il Project Management Institute (PMI). si definisce come

"Il project management corrisponde all'applicazione di conoscenze, capacità, strumenti e tecniche alle attività di progetto per soddisfare i requisiti." [3]

Per perseguire l'obiettivo il Project Management si serve dei seguenti passaggi:

1. Avere una previsione delle attività e dei processi necessari per concludere con successo il progetto.
2. Seguire le linee guida stabilite per i budget, i tempi e gli obiettivi da raggiungere.
3. Risolvere eventuali problemi in maniera efficace ed efficiente.
4. Identificare e eliminare attività che sono inutili ai fini del progetto.
5. Aumentare l'efficienza e la collaborazione tra i team e al loro interno.
6. Gestire i rischi.

Solitamente, un progetto interagisce con altri progetti che condividono gli stessi obiettivi. In questo contesto, si parla di un Programma, che è definito come:

"Un insieme di progetti correlati, gestiti in modo coordinato al fine di ottenere benefici e un controllo non possibili nella gestione individuale dei singoli progetti." [3]

Un progetto può essere definito considerando tre fattori chiave, quali:

1. **Contesto:** Il dominio applicativo del progetto stesso.
2. **Lifecycle:** L'arco temporale in cui opera il progetto caratterizzato da una finestra compresa tra le date di inizio e fine.
3. **Lifecycle cost:** Il budget che permette il compimento del progetto sostenendo i costi durante tutta la sua durata.

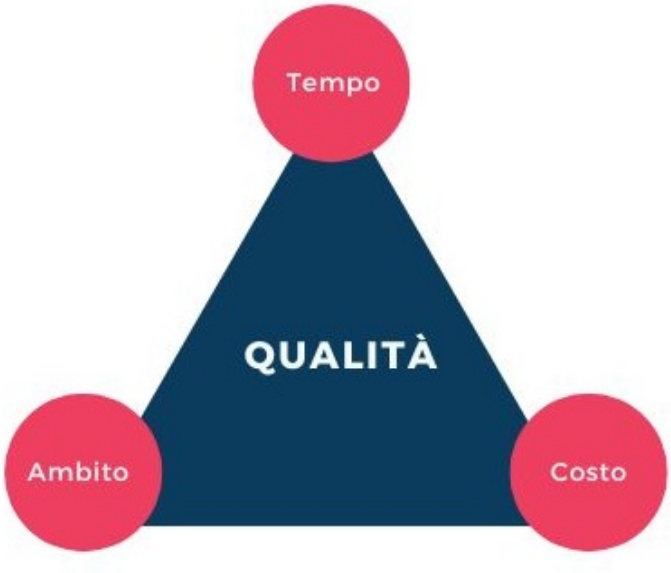

Figura 1.1: Triangolo dello scope

1.1.1 Contesto di progetto

L'ambiente in cui si sviluppa il progetto definisce anche la sua natura, questo si compone principalmente di due attori principali:

1. **Attori Esterni:** Elementi esterni che però inevitabilemente impattano sul progetto stesso, come enti governativi e concorrenti

2. **Stakeholder:** Sono individui, gruppi o entità che hanno un interesse o sono influenzati dalle attività, dagli obiettivi o dalle decisioni di un'organizzazione, di un progetto o di un'iniziativa.

   Questi soggetti possono essere interni o esterni e includono azionisti, dipendenti, clienti, fornitori, comunità locali, governo e altre parti interessate.

   Gli stakeholder possono avere diverse prospettive, interessi e impatti sull'attività o sul progetto in questione ed è importante considerare le loro esigenze e preoccupazioni nella gestione e nella pianificazione per garantire un coinvolgimento efficace.

1.1.2 Lifecycle di progetto

Ogni progetto è unico per definizione e, di conseguenza, il suo lifecycle è altrettanto unico, sebbene tutti i progetti condividano una serie di passaggi necessari per governare il progetto e consegnare il risultato atteso.

Il ciclo di vita di un progetto si propone di fornire al Project Manager una guida per le diverse fasi del progetto e per distribuire il lavoro su un arco temporale secondo una sequenza ben definita di attività, quali:

1. **Fase di iniziazione:** La fase di iniziazione del progetto rappresenta il punto di partenza, in cui viene definito il progetto stesso insieme alle basi fondamentali su cui si fonda. Durante questa fase, si svolgono diverse attività cruciali, tra cui:

   (a) l'identificazione degli obiettivi del progetto

   (b) la valutazione della sua fattibilità

   (c) l'analisi dei rischi iniziali

   (d) l'identificazione delle parti chiave

   Inoltre, si raccolgono e si definiscono i requisiti funzionali, cioè le specifiche delle funzionalità e delle caratteristiche che il prodotto o il servizio finale dovrà soddisfare per essere considerato un successo. Questa fase svolge un ruolo fondamentale nel fornire una visione chiara e condivisa del progetto, stabilendo le fondamenta per le fasi successive e garantendo che il team abbia una comprensione comune degli obiettivi e delle aspettative del progetto.

2. **Fase di pianificazione:** Fase in cui si definiscono gli elementi del progetto allo scopo di creare una mappa delle attività da svolgere associando un concetto di "tempo" utile al compimento di queste fasi.

3. **Fase di esecuzione:** In questo step viene allocato il team di lavoro e iniziano formalmente le attività precedentemente pianificate; questa fase è la più lunga di tutto il lifecycle visto che incamera tutte le attività atte alla realizzazione del prodotto.

4. **Fase di monitoraggio e controllo:** Fase dedicata al monitoraggio del prodotto dopo la consegna dello stesso, in questo step lo scopo è quello di identificare il più velocemente possibile le problematiche nascoste e seguire in tempi brevi alla loro risoluzione.

   Questa fase può durare più o meno tempo in funzione del flusso di lavoro usato durante gli sviluppi nella fase di esecuzione, è fisiologica la presenza di problematiche nel prodotto finale ma la mole di queste problematiche è gestibile tramite l'implementazione di un flusso di qualità non solo a valle degli sviluppi ma anche durante la loro realizzazione.

5. **Chiusura del progetto:** La fase di chiusura del progetto rappresenta l'ultimo step del ciclo di vita, durante il quale tutte le attività vengono completate e il prodotto viene consegnato al cliente, accompagnato da una documentazione esaustiva.

1.1.3 Lifecycle cost

Il lifecycle cost definisce i costi da sostenere lungo l'intero svolgimento del progetto. La stima dei costi, pratica che si espleta nella fase di pianificazione e il monitoraggio costante sono gli obiettivi principali del Cost Management.

Definiamo come Cost Management l'attività che si occupa di stimare, stanziare, monitorare e controllare i costi del progetto, permettendo di prevedere le spese future di un progetto, definire un budget e diminuire il rischio di superarlo.

Risulta fondamentale stimare questi costi perchè senza questa analisi non si può essere certi che il budget stanziato sia sufficiente per tutte le spese necessarie al completamento del progetto, risulta chiaramente fondamentale avere un budget allocato tale che sia fondamentale poter sopperire a tutto il ciclo di vita del progetto, il rischio è l'interruzione delle attività per mancanza di fondi.

Il Cost Management è una stima che si definisce generalmente per confronto quindi si usano come benchmark i progetti precedentemente svolti che costituiscono la conoscenza pregressa, considerando quindi i progetti lavorati nel passato possiamo stimare i costi per i progetti futuri, questo tipo di stima si definisce **Stima per confronto**.

Solitamente, l'andamento dei costi nel tempo all'interno di un progetto segue una distribuzione a forma di campana. Inizialmente, i costi sono bassi, poi aumentano fino a raggiungere il picco durante la fase centrale del progetto, poiché molte attività richiedono risorse in quel periodo. Successivamente, i costi tendono a diminuire verso la conclusione del progetto.

1.2 Definizione di Project Management

Il Project Management nell'ambito dello sviluppo software è un approccio metodologico fondamentale che comprende un insieme articolato di pratiche, strumenti e processi finalizzati alla gestione efficace e efficiente dei progetti software.

Questo campo multidisciplinare si concentra sulla pianificazione, organizzazione, coordinamento, esecuzione e controllo delle attività coinvolte nello sviluppo di software, con l'obiettivo primario di raggiungere gli obiettivi prefissati, rispettando vincoli quali tempo, costi, risorse disponibili e la qualità del prodotto.

Le pratiche del Project Management mirano a creare un ambiente di lavoro strutturato e organizzato, promuovendo la collaborazione e la comunicazione tra i membri del team e gli stakeholder coinvolti nel progetto.

La corretta implementazione di queste pratiche consente di gestire in modo efficiente le risorse umane, tecniche e finanziarie, minimizzando i rischi e ottimizzando l'utilizzo dei mezzi a disposizione.

Le metodologie utilizzate nel Project Management possono variare in base alle esigenze specifiche del progetto e alle preferenze dell'organizzazione coinvolta.

Tuttavia, sia che si tratti di approcci tradizionali o Agili, l'obiettivo rimane sempre quello di guidare il team attraverso tutte le fasi del ciclo di vita del progetto, dalla sua concezione alla sua conclusione, garantendo il rispetto dei tempi di consegna, il controllo dei costi, la qualità e l'aderenza ai requisiti stabiliti.

La **pianificazione** riveste un ruolo cruciale, consentendo di definire con precisione gli obiettivi, suddividere le attività in compiti gestibili, stabilire scadenze realistiche e assegnare responsabilità specifiche ai membri del team.

Il **coordinamento delle risorse**, comprese le competenze tecniche, l'infrastruttura e gli strumenti necessari, è fondamentale per mantenere l'intero processo di sviluppo allineato agli obiettivi e ai piani stabiliti.

Il **monitoraggio e controllo costante delle attività**, attraverso la valutazione delle prestazioni, l'identificazione e la gestione dei rischi, nonché l'implementazione di eventuali modifiche necessarie, è essenziale per garantire che il progetto prosegua sulla buona strada e per correggere eventuali deviazioni rispetto al piani iniziali.

Infine, la **qualità del prodotto software** è un pilastro imprescindibile del Project Management.

Attraverso l'adozione di metodologie di sviluppo e di test approfonditi, il monitoraggio costante della conformità ai requisiti e il coinvolgimento continuo degli stakeholder, si mira a produrre un software affidabile e funzionale che soddisfi le esigenze e le aspettative degli utenti finali. Definiamo quindi in modo formale e non ambiguo quello che è il Project Management secondo il **Project Management Institute (PMI)**

"l'applicazione di conoscenze, competenze, strumenti e tecniche alle attività del progetto per soddisfare i requisiti del progetto" [3]

1.3 Elementi del Project Management

Andiamo ora in questa sezione a trattare dettagliatamente quelli che sono gli elementi chiave che compongono il Project Management focalizzati sullo sviluppo software.

1. **Scoping:** La fase di scoping rappresenta il fondamento su cui si basa l'intero progetto, poiché fornisce una roadmap chiara e dettagliata per il suo successo. Durante questa fase, vengono svolte una serie di attività cruciali che pongono le basi per il progresso.

Innanzitutto, si procede con l'identificazione degli obiettivi del progetto, che devono essere definiti in modo SMART:

(a) Specific: L'obiettivo deve essere chiaro e ben definito, evitando ambiguità e offrendo una chiara direzione.

(b) Measurable: L'obiettivo deve essere quantificabile in modo che sia possibile valutare il progresso e determinare se è stato raggiunto.

(c) Achievable: L'obiettivo deve essere realistico e raggiungibile, considerando le risorse disponibili e le capacità della persona o del team incaricato di raggiungerlo.

(d) Relevant: L'obiettivo deve essere rilevante e allineato agli obiettivi più ampi dell'organizzazione o della persona. Deve avere un impatto significativo sulle attività e sui risultati desiderati.

(e) Time-bound: L'obiettivo deve essere vincolato da un limite temporale definito, che aiuta a creare un senso di urgenza e a mantenere il focus sul raggiungimento dell'obiettivo entro un determinato periodo di tempo.

Successivamente, viene effettuata la raccolta e l'analisi dei requisiti del progetto, processo che coinvolge l'identificazione e la comprensione delle esigenze e delle aspettative degli stakeholder, nonché la determinazione delle funzionalità e delle caratteristiche che il prodotto deve soddisfare per essere considerato un successo.

2. **Pianificazione:** Viene redatto un piano di progetto dettagliato che delineerà tutte le attività, le risorse e le scadenze necessarie per portare a termine il progetto con successo. Questo piano conterrà una pianificazione temporale delle attività, specificando chi è responsabile di ciascuna attività e quando deve essere completata.

   Infine, vengono definite le milestone e le scadenze chiave del progetto. Le milestone rappresentano i principali traguardi o obiettivi intermedi che devono essere raggiunti lungo il percorso del progetto. Questi punti di riferimento consentono al team di monitorare l'avanzamento del progetto e di valutare se il progetto sta procedendo secondo le aspettative. Le scadenze, invece, indicano i termini entro i quali determinate attività o fasi del progetto devono essere completate. La definizione di milestone e scadenze permette al team di tenere traccia del progresso e di adattarsi prontamente a eventuali cambiamenti o imprevisti che possono verificarsi durante l'esecuzione del progetto.

3. **Organizzazione delle Risorse:** L'efficace gestione delle risorse è cruciale per il successo del progetto. Coinvolge una serie di attività chiave volte a garantire che il team abbia a disposizione le risorse necessarie per completare il progetto in modo efficiente e soddisfacente.

Innanzitutto, ciò comporta l'allocazione dei ruoli specifici ai membri del team. Questo processo richiede una valutazione attenta delle competenze, delle esperienze e delle capacità di ciascun membro del team, in modo da assegnare loro responsabilità e compiti che siano in linea con le loro capacità e che contribuiscano al successo complessivo del progetto. Un'allocazione dei ruoli ben pianificata consente di sfruttare al meglio le competenze individuali e di massimizzare l'efficacia del team nel suo complesso.

La preparazione dell'infrastruttura necessaria rappresenta un'altra parte critica della gestione delle risorse. Questo può includere l'acquisizione o la configurazione di strumenti, tecnologie e ambienti di lavoro necessari per supportare le attività del progetto. Garantire che l'infrastruttura sia pronta e funzionante in anticipo permette al team di concentrarsi sulle attività principali del progetto senza interruzioni o ritardi causati da problemi tecnici.

4. **Coordinamento e Gestione delle Attività:** Il coordinamento delle attività è un aspetto fondamentale della gestione di un progetto e richiede una supervisione attiva e costante delle operazioni quotidiane. Coinvolge una serie di azioni volte a garantire che tutte le attività del progetto siano svolte in modo efficiente e conforme agli obiettivi stabiliti.

   Innanzitutto, ciò comporta l'assegnazione e il monitoraggio delle attività. Questo processo implica l'attribuzione di compiti specifici ai membri del team, tenendo conto delle loro competenze, delle risorse disponibili e delle scadenze stabilite. Una volta assegnate, le attività devono essere monitorate attentamente per assicurarsi che vengano completate in tempo e secondo le specifiche.

   La gestione delle dipendenze tra i compiti è un altro aspetto critico del coordinamento delle attività. Molte volte, le attività di un progetto dipendono l'una dall'altra e la loro sequenza di completamento è fondamentale per il successo complessivo del progetto. Gestire queste dipendenze richiede una pianificazione attenta e una comunicazione chiara per garantire che le attività vengano eseguite nell'ordine corretto e che non ci siano ritardi o interruzioni nel flusso di lavoro.

   La risoluzione tempestiva dei problemi è un'altra responsabilità chiave del coordinamento delle attività. Durante lo svolgimento del progetto, possono sorgere imprevisti, ostacoli o problemi che richiedono una rapida risoluzione per evitare ritardi o deviazioni dal piano. Un coordinatore delle attività efficace deve essere in grado di identificare tempestivamente i problemi, trovare soluzioni appropriate e garantire che il progetto rimanga sulla giusta traccia.

   Il coordinamento delle attività richiede una leadership forte e una comunicazione efficace per garantire che il lavoro di tutti sia sincronizzato e mirato a raggiungere gli obiettivi del progetto in modo collaborativo e efficiente.

5. **Monitoraggio delle attività:** il monitoraggio continuo è fondamentale per garantire che il progetto rimanga all'interno dei limiti prestabiliti e che l'esecuzione sia conforme alla pianificazione stabilita.

   Il monitoraggio costante consente al team di tenere sotto controllo i vari aspetti del progetto, tra cui i tempi, i costi, le risorse e la qualità del lavoro svolto. Questo permette di identificare tempestivamente eventuali variazioni rispetto alla pianificazione iniziale e di adottare le misure necessarie per mantenere il progetto sulla giusta traccia.

   Inoltre, il monitoraggio continuo consente di verificare che l'esecuzione del progetto sia conforme a quanto pianificato. Ciò significa assicurarsi che le attività vengano svolte secondo le specifiche stabilite, che i risultati intermedi siano allineati agli obiettivi del progetto e che vengano rispettati i criteri di qualità definiti.

   Mantenere un monitoraggio continuo permette al team di rilevare eventuali deviazioni o problemi in modo tempestivo, consentendo di adottare le azioni correttive necessarie per riportare il progetto sulla giusta traccia. Inoltre, fornisce una maggiore trasparenza e visibilità sull'avanzamento del progetto, migliorando la comunicazione e la collaborazione all'interno del team e con gli stakeholder interessati.

6. **Controllo delle attività:** Il controllo costante dell'avanzamento del progetto rappresenta una pratica fondamentale per garantire il successo complessivo e per intervenire prontamente in caso di necessità con azioni correttive.

   Il controllo consente di valutare regolarmente lo stato del progetto rispetto alla pianificazione stabilita, identificando eventuali discrepanze tra quanto previsto e quanto effettivamente realizzato. Quando si rilevano queste non conformità, il controllo fornisce il quadro necessario per implementare azioni correttive o preventive.

   Le azioni correttive sono fondamentali quando si verificano discrepanze tra lo stato attuale del progetto e quanto pianificato. Tali azioni possono includere modifiche alla pianificazione, alla distribuzione delle risorse o alla gestione delle attività, al fine di riportare il progetto sulla giusta traccia e garantire il raggiungimento degli obiettivi.

   D'altro canto, le azioni preventive sono volte a prevenire il verificarsi di problemi o deviazioni nel progetto. Queste azioni possono comprendere l'implementazione di controlli aggiuntivi, l'aggiornamento delle procedure o l'allocazione di risorse extra per mitigare i rischi identificati.

   In entrambi i casi, il controllo costante dell'avanzamento del progetto offre una visione chiara dello stato attuale, consentendo al team di agire tempestivamente per mantenere il progetto sulla giusta traccia e massimizzare le probabilità di successo.

7. **Comunicazione e Collaborazione:** la comunicazione efficace rappresenta un pilastro fondamentale per il successo del progetto. Coinvolge una serie di pratiche e attività che contribuiscono a mantenere un flusso costante e chiaro di informazioni tra tutti i membri coinvolti nel progetto, quali:

   (a) Trasmissione di informazioni chiare e tempestive: La comunicazione efficace richiede che le informazioni pertinenti vengano trasmesse in modo chiaro, accurato e tempestivo a tutti i membri del team e agli stakeholder interessati. Questo include la condivisione di obiettivi, piani, progressi, sfide e risoluzioni.

   (b) Gestione delle aspettative degli stakeholder: È importante comprendere e gestire le aspettative degli stakeholder, che possono essere interne o esterne all'organizzazione. Questo coinvolge l'identificazione delle loro esigenze, preoccupazioni e priorità, e la creazione di canali di comunicazione adatti per soddisfare tali esigenze.

   (c) Risoluzione di conflitti: La comunicazione efficace aiuta a prevenire e a risolvere i conflitti che possono sorgere durante lo svolgimento del progetto. Questo richiede una gestione empatica e professionale dei conflitti, cercando soluzioni collaborative e inclusione di tutte le parti interessate per raggiungere un consenso.

   (d) Promozione di una cultura collaborativa: Una comunicazione aperta e trasparente favorisce una cultura di collaborazione all'interno del team e tra gli stakeholder. Questo crea un ambiente in cui le idee possono essere condivise liberamente, le competenze possono essere integrate e i membri del team possono lavorare insieme per affrontare le sfide e raggiungere gli obiettivi del progetto.

8. **Gestione dei Rischi:** la gestione dei rischi è un processo fondamentale per garantire il successo del progetto e per affrontare in modo proattivo le potenziali minacce o incertezze che potrebbero influenzarlo negativamente. Ecco una panoramica estesa del processo di gestione dei rischi:

   (a) Identificazione dei rischi: La prima fase coinvolge l'identificazione di tutte le potenziali minacce che potrebbero avere un impatto sul progetto. Questo può includere rischi legati al budget, alle risorse umane, alla tecnologia, ai requisiti del cliente, alle scadenze, ai fornitori, ai cambiamenti normativi e ai rischi ambientali. L'obiettivo è individuare il più ampio spettro possibile di rischi che potrebbero verificarsi durante lo svolgimento del progetto.

   (b) Valutazione dei rischi: Una volta identificati, i rischi vengono valutati in base alla loro probabilità di accadere e all'impatto che potrebbero avere sul progetto se si verificassero. Questo processo aiuta a determinare quali rischi richiedono

un'attenzione immediata e quali possono essere gestiti in modo più informale. L'impatto atteso, calcolato come prodotto della probabilità per l'impatto, fornisce una metrica chiave per comprendere la gravità dei rischi. La valutazione dei rischi, considerando sia la probabilità che l'impatto atteso, fornisce una base per stabilire le priorità e concentrare le risorse sulla gestione dei rischi più critici.

(c) Mitigazione dei rischi: Dopo aver valutato i rischi, vengono pianificate e implementate strategie di mitigazione per ridurre la probabilità di accadimento dei rischi identificati o per ridurne l'impatto sul progetto nel caso in cui si verifichino. Questo può includere l'adozione di azioni preventive per prevenire il verificarsi dei rischi, il trasferimento dei rischi ad altri soggetti, la riduzione dell'esposizione ai rischi tramite cambiamenti nella pianificazione del progetto o l'utilizzo di assicurazioni.

(d) Accettazione dei rischi: Dopo aver valutato i rischi, si arriva al punto in cui alcune minacce potrebbero essere considerate accettabili considerando gli impatti attesi e le risorse disponibili. In presenza di rischi con impatti attesi limitati, può essere deciso di non intraprendere azioni correttive formali, ma piuttosto di accettare le possibili conseguenze. Questa strategia può coinvolgere il trasferimento del rischio ad altre parti tramite l'acquisto di polizze assicurative, la stipula di fideiussioni bancarie o l'outsourcing ad un fornitore che si assuma la responsabilità dei potenziali ritardi o fallimenti. L'accettazione dei rischi non implica inattività, ma piuttosto una gestione consapevole e mirata dei rischi considerati accettabili.

(e) Piani di contingenza: In aggiunta alle strategie di mitigazione, vengono sviluppati piani di contingenza per affrontare i rischi che non possono essere completamente evitati o ridotti. Questi piani definiscono le azioni specifiche che devono essere intraprese nel caso in cui si verifichi un determinato rischio, al fine di minimizzare l'impatto sul progetto e garantire la sua continuazione senza gravi interruzioni. I piani di contingenza forniscono una guida chiara su come reagire in situazioni di emergenza o di imprevisti, consentendo al team di essere preparato per qualsiasi eventualità.

9. **Garanzia della Qualità del Prodotto:** la garanzia della qualità del prodotto software è essenziale per garantire la soddisfazione degli utenti finali e il successo complessivo del progetto. Ecco una panoramica espansa di questo processo:

   (a) Implementazione di processi di sviluppo: La garanzia della qualità inizia con l'implementazione di processi di sviluppo ben definiti e strutturati. Questi processi comprendono le pratiche di ingegneria del software, i metodi di sviluppo, le metodologie di gestione del progetto e le normative di settore. L'obiettivo

è garantire che il software sia sviluppato in modo sistematico, efficiente e conforme agli standard di qualità stabiliti.

(b) Test della qualità: Un elemento cruciale della garanzia della qualità è la conduzione di test mirati per valutare le prestazioni, la funzionalità, la sicurezza e l'affidabilità del software. Questi test possono includere test unitari, test di integrazione, test di sistema, test di accettazione e test di regressione. L'obiettivo è identificare eventuali difetti o anomalie nel software e garantire che venga correttamente rispettata la sua specifica funzionale e tecnica.

(c) Controllo della qualità: Il controllo della qualità implica la valutazione continua del software durante tutto il ciclo di vita del progetto. Ciò include l'ispezione del codice, la revisione dei documenti, la verifica della conformità agli standard e la misurazione delle prestazioni rispetto agli obiettivi di qualità stabiliti. Il controllo della qualità consente di identificare eventuali problemi o difetti in modo tempestivo e di adottare le misure correttive necessarie per garantire la conformità del software agli standard di qualità richiesti.

(d) Consegna di un software funzionale: L'obiettivo finale della garanzia della qualità è consegnare un software funzionale e di alta qualità agli utenti finali. Ciò significa assicurarsi che il software soddisfi le loro esigenze e le aspettative, che sia affidabile, sicuro e performante, e che sia conforme agli standard e alle normative pertinenti del settore. La consegna di un software di qualità massimizza la soddisfazione degli utenti finali e aumenta la reputazione e la credibilità dell'organizzazione che lo ha sviluppato.

Capitolo 2

Metodologie di Project Management

2.1 Metodologia Waterfall

2.1.1 Definizione della Waterfall Project Management

La metodologia Waterfall, descritta da Winston W. Royce nel suo articolo del 1970 intitolato "Managing the Development of Large Software Systems", è stata una delle prime metodologie strutturate utilizzate nello sviluppo del software. Il suo nome deriva dall'idea di un flusso di lavoro che procede in modo sequenziale, simile alla caduta dell'acqua in una cascata.

Il modello Waterfall è stato uno dei primi approcci adottati per gestire progetti software complessi, soprattutto in un'epoca in cui la disciplina dell'Ingegneria del Software stava ancora emergendo.

Storicamente, il modello Waterfall si è sviluppato grazie alle industrie di costruzione e produzione, infatti queste occupandosi della costruzione di edifici e infrastrutture partivano dal creare ogni singolo pezzo necessario per poi assemblarli, quindi una volta completato il montaggio, rifare anche solo una porzione del progetto richiedeva un costo elevato.

2.1.2 Caratteristiche del Project Management a Waterfall

I progetti a Waterfall sono definiti da una serie di caratteristiche:

1. **Il modello Waterfall è Project-based:** il Waterfall si concentra sul progetto stesso più che sulla soddisfazione del cliente finale alla consegna dell'artefatto.

2. **Tutti i processi sono sequenziali:** Si intende che non è possibile procedere all'attività successiva senza che sia stata conclusa l'attività precedente e dualmente non è possibile apportare modifiche alla fase precedente.

Questo approccio assume un carattere immutabile, quasi fosse scolpito nella pietra. Significa che viene percepito come definitivo e non soggetto a cambiamenti. Questo implica che ogni tentativo di modificare o adattare l'approccio potrebbe essere visto come un'impresa ardua, se non impossibile. Di conseguenza, le decisioni prese con questo metodo vengono trattate con un alto grado di serietà e stabilità, poiché la loro revisione o alterazione richiederebbe sforzi significativi e giustificazioni molto forti.

3. Le valutazioni sull'andamento del progetto sono fatte sul tempo speso, sull'esborso economico e su quanta manodopera è stata impiegata sulla base dei requisiti.

4. Ogni progetto gestito a Waterfall ha un persorso rigido definito per la realizzazione del progetto stesso.

5. **Controllo stato avanzamento:** In Waterfall esistono particolari chiavi di misura precise (Key Performance Indicator) come Earned Value Analysis (EVA) che permettono di misurare il progresso in un qualsiasi momento.

6. La fine di un progetto coincide con il passaggio al team di supporto.

7. Il successo del progetto è analizzato internamente in linea generale ma in determinati momenti viene previsto anche il coinvolgimento del cliente che può fornire giudizi anche negativi che possono avere un impatto sull'accettazione del risultato.

In conclusione, quindi, la filosofia Waterfall è quella di ottenere il miglior risultato possibile con la minor spesa in termini economici, temporali e di manodopera, tutto ciò giustifica la rigidità al cambiamento una volta definito il flusso.

2.1.3 Fasi di progetto

Il modello Waterfall si basa su un approccio sequenziale e lineare, composto da fasi distinte e ben definite, ciascuna delle quali dipende dal completamento della precedente.

Le fasi tipiche includono:

1. **Analisi dello Scope:** In questa fase iniziale, vengono identificati e raccolti tutti i requisiti del progetto.

   Gli obiettivi, le funzionalità e le restrizioni del software vengono definiti in modo dettagliato.

2. **Progettazione:** Dopo aver compreso i requisiti, si procede alla fase di progettazione, in cui vengono delineati l'architettura del sistema e il design tecnico del software.

In questa fase si definiscono i dettagli su come implementare le funzionalità stabilite durante la fase di analisi dei requisiti.

3. **Sviluppo:** Una volta completata la progettazione, inizia lo sviluppo effettivo del software.

   I programmatori scrivono il codice seguendo le specifiche e i prototipi stabiliti nella fase di progettazione.

4. **Testing:** Durante questa fase, il software è sottoposto a test approfonditi per garantire che soddisfi i requisiti stabiliti e corrisponda alle attese.

   Il monitoraggio e il controllo, focalizzati principalmente sulla fase di sviluppo e test, si estendono anche ad altre aree del progetto. Non solo si verifica la conformità del prodotto alle specifiche, ma anche l'efficacia dei processi di gestione del progetto.

5. **Installazione e Manutenzione:** L'ultima fase coinvolge l'installazione del software nel contesto operativo e il suo supporto continuo.

   Nei progetti waterfall, si pianifica il processo di manutenzione durante lo svolgimento del progetto, anche se questa manutenzione non è inclusa nel progetto stesso, a meno che non sia necessaria per ottenere l'accettazione. Eventuali problemi rilevati durante l'utilizzo vengono corretti e il software viene aggiornato di conseguenza.

Quindi, risulta necessario rappresentare il workflow e la migliore modalità possibile è l'uso del diagramma Gantt che è uno strumento atto a rappresentare la pianificazione temporale delle attività.

Il diagramma Gantt è costituito da un asse orizzontale che rappresenta il tempo e da un asse verticale che elenca le attività del progetto.

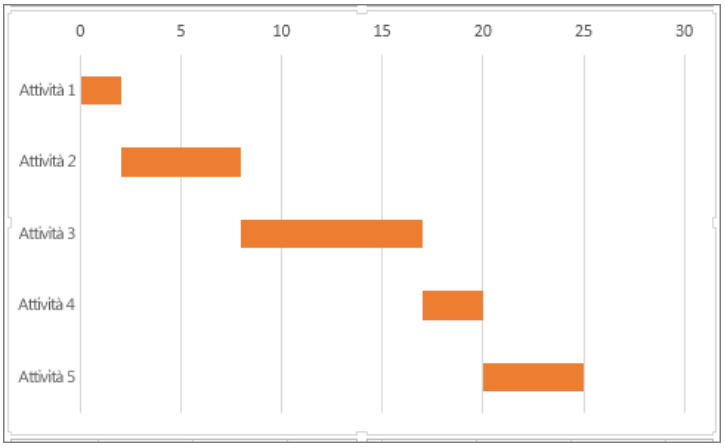

Figura 2.1: Diagramma di Gantt

Caratteristiche principali del diagramma di Gantt:

1. **Attività**: Ogni attività del progetto è rappresentata da una barra orizzontale sul diagramma.

   La posizione della barra indica l'inizio e la fine dell'attività nel tempo.

2. **Durata**: La lunghezza della barra rappresenta la durata prevista per completare l'attività.

   Una barra più lunga indica un'attività più lunga nel tempo.

3. **Relazioni tra attività**: È possibile rappresentare le dipendenze tra le attività, ad esempio, mostrando che l'inizio di un'attività dipende dal completamento di un'altra.

4. **Sequenza delle attività**: Le barre delle attività sono posizionate lungo l'asse temporale in base alla sequenza delle attività del progetto.

   Le attività precedenti devono essere completate prima che possano iniziare quelle successive.

5. **Milestone**: Punti significativi nel progetto, come tappe importanti o obiettivi raggiunti, possono essere rappresentati come punti sul diagramma di Gantt.

2.1.4 Vantaggi e Svantaggi

Il modello Waterfall, noto anche come modello a cascata, è uno dei metodi di sviluppo del software più tradizionali e ampiamente utilizzati. Questo approccio segue una sequenza lineare di fasi, ognuna delle quali deve essere completata prima di passare alla successiva. Di seguito sono elencati alcuni dei principali vantaggi di utilizzare il modello Waterfall.

1. **Struttura chiara e sequenziale** - Il modello Waterfall offre una struttura sequenziale ben definita, organizzata in fasi distinte e ordinate.

   Questa linearità fornisce un quadro chiaro e lineare alle attività di sviluppo, consentendo una gestione ordinata e comprensibile del progetto.

   La definizione chiara delle fasi aiuta i team a comprendere il processo nel suo complesso, facilitando la suddivisione delle responsabilità e consentendo una migliore pianificazione delle attività.

2. **Documentazione dettagliata** - Ogni fase nel modello Waterfall richiede una documentazione dettagliata, compresi requisiti, progettazione, codifica e test.

   Questa documentazione accurata contribuisce alla tracciabilità dei requisiti e delle decisioni prese durante lo sviluppo del software.

La creazione di documenti dettagliati aiuta a mitigare il rischio di ambiguità, consentendo a diverse parti interessate di comprendere il progetto e fornendo un riferimento solido per guidare lo sviluppo e le fasi successive.

3. **Facilità nella gestione dei progetti** - La natura sequenziale e ben strutturata del modello Waterfall facilita la gestione dei progetti.

   La chiarezza delle fasi consente una pianificazione dettagliata e il monitoraggio del progresso del progetto.

   Ciò rende più agevole stimare i tempi, i costi e le risorse necessarie per ciascuna fase, consentendo ai responsabili di progetto di prendere decisioni informate e di mantenere il controllo sullo sviluppo del software.

4. **Rigidezza al cambiamento** - Una delle criticità principali del modello Waterfall è la sua resistenza al cambiamento una volta avviato il processo.

   A causa della sua struttura lineare e sequenziale, è difficile apportare modifiche ai requisiti in una fase avanzata del ciclo di sviluppo.

   Le modifiche richiedono spesso di apportare delle variazioni a quanto fatto nelle fasi precedenti e potrebbero comportare un incremento di costo significativo oltre che a provocare dei ritardi con il rischio di superare i budget previsti e non rispettare le scadenze.

5. **Test tardivi** - Nel modello Waterfall il testing, tradizionalmente, avviene solo dopo che lo sviluppo del software è stato completato. Questo approccio può generare problematiche se vengono rilevati errori o problemi significativi nelle fasi finali del ciclo di sviluppo. Gli errori individuati tardivamente possono essere costosi da correggere e possono richiedere modifiche sostanziali che influenzano il piano di progetto e i tempi di consegna.

   Tuttavia, questo limite può essere mitigato prevedendo dei test alla conclusione di ciascuna attività. Implementando una strategia di testing incrementale, i team possono individuare e risolvere i problemi in una fase più precoce del processo di sviluppo.

   Questa pratica non solo riduce il rischio di problemi significativi nelle fasi finali, ma consente anche di apportare correzioni in modo più efficiente e meno costoso. Inoltre, il testing incrementale migliora la qualità complessiva del software e garantisce che ogni fase del progetto soddisfi i criteri di accettazione prima di avanzare. In questo modo, il modello Waterfall può diventare più flessibile e adattabile, mantenendo comunque la sua struttura sequenziale e ben definita.

6. **Rischio di non soddisfare requisiti non identificati** - Una delle sfide principali del modello Waterfall è la necessità di comprendere completamente i requisiti del progetto fin dall'inizio.

Se alcuni requisiti o dettagli critici non sono stati correttamente identificati nelle prime fasi, c'è il rischio di non soddisfare appieno le esigenze del cliente o degli utenti finali.

Questo può comportare una revisione significativa o persino la ristrutturazione del progetto, con conseguenti ritardi e costi aggiuntivi.

2.2 Metodologia Agile

2.2.1 Definizione della Agile Project Management

L'approccio Agile rappresenta una filosofia di sviluppo software che si concentra sulla flessibilità, sulla collaborazione e sull'adattabilità rispetto ai cambiamenti.

Questa metodologia è nata come risposta ai limiti delle metodologie tradizionali, come il modello Waterfall, offrendo una modalità di sviluppo più dinamica e adattabile alle mutevoli esigenze dei progetti.

Le origini della metodologia Agile secondo alcune ricerche risalgono agli anni 60 del 900, una prima versione di questo approccio è la metodologia Scrum definita nel 1986 che considerava la volatilità dei requisiti dovuta al possibile cambiamento delle richieste dei clienti una volta consegnato il prodotto ultimato [3].

La nascita ufficiale è stata però decretata nel febbraio del 2001 con la scrittura del manifesto Agile.

Manifesto Agile

Il Manifesto Agile è nato nel febbraio 2001 durante una riunione tenutasi presso il resort sciistico di Snowbird, Utah, negli Stati Uniti.

Questo incontro è stato organizzato da un gruppo di sviluppatori software, esperti e influenti nel settore, desiderosi di trovare alternative ai metodi di sviluppo software tradizionali, spesso rigidi e poco adattabili ai cambiamenti.

Il contesto in cui è emerso il Manifesto Agile era caratterizzato da una crescente insoddisfazione verso i metodi di sviluppo software tradizionali, come il waterfall, anche detti "pesanti".

Come detto, i metodi waterfall seguono un approccio sequenziale, in cui le fasi del progetto venivano svolte in modo lineare e rigido, con una pianificazione dettagliata all'inizio e poca flessibilità per adattarsi ai cambiamenti dei requisiti o del contesto.

Durante la riunione a Snowbird, un gruppo di 17 sviluppatori, leader e teorici del settore, si sono riuniti per discutere di metodi alternativi allo sviluppo software tradizionale, possiamo citarne alcuni:

1. Kent Beck
2. Mike Beedle
3. Arie van Bennekum
4. Alistair Cockburn
5. Ward Cunningham
6. Martin Fowler
7. James Grenning
8. Jim Highsmith
9. Andrew Hunt
10. Ron Jeffries
11. Jon Kern
12. Brian Marick
13. Robert C. Martin
14. Steve Mellor
15. Ken Schwaber
16. Jeff Sutherland
17. Dave Thomas [2]

Durante questa riunione, emerse un consenso comune sulla necessità di promuovere un approccio più flessibile, adattabile e collaborativo allo sviluppo del software.

Questi professionisti condivisero le loro esperienze, le sfide incontrate con i metodi tradizionali e le loro visioni su come migliorare il processo di sviluppo.

Da questo dibattito emerse il Manifesto Agile, un breve documento che enfatizzava valori e principi fondamentali per lo sviluppo software.

Il Manifesto Agile non proponeva un metodo specifico, ma delineava valori basilari e principi guida che hanno dato vita a metodologie come Scrum, Extreme Programming (XP), Kanban e altre.

Il manifesto Agile è basato su quattro valori e dodici principi fondamentali da seguire per la riuscita del progetto. [3]

I **quattro valori fondamentali** del Manifesto Agile sono:

1. **Individui e interazioni più che processi e strumenti** - L'attenzione è posta sul valore delle persone coinvolte nel progetto e sulla comunicazione efficace tra di esse.

2. **Software funzionante più che documentazione esaustiva** - Si attribuisce maggiore importanza alla produzione di software funzionante rispetto alla creazione di documenti dettagliati, pur riconoscendo l'importanza della documentazione necessaria.

3. **Collaborazione con il cliente più che negoziazione dei contratti** - Si promuove la collaborazione attiva con il cliente durante tutto il processo di sviluppo, favorendo l'adattamento alle esigenze in evoluzione.

4. **Rispondere al cambiamento più che seguire un piano** - L'Agile incoraggia la flessibilità e la capacità di adattamento alle modifiche dei requisiti, accettando che i piani possano cambiare nel corso dello sviluppo.

I quattro valori fondamentali del Manifesto Agile fungono da guida per i membri del team durante l'esecuzione di ogni ciclo di iterazione. Questi valori consentono di migliorare costantemente lo scopo, il design e il flusso di lavoro del progetto ad ogni iterazione.

Principi del Manifesto Agile

I dodici principi su cui si basa ogni progetto che utilizza queste metodologie sono i seguenti:

1. Soddisfare il cliente attraverso la consegna continua di software funzionante.

   Si enfatizza la consegna rapida e iterativa di funzionalità operative per soddisfare le esigenze del cliente.

2. Accogliere cambiamenti dei requisiti anche tardivi nello sviluppo.

   Si accetta la possibilità che i requisiti possano cambiare nel corso del progetto, adattandosi a tali cambiamenti per massimizzare il valore del software prodotto.

3. Completare cicli di sviluppo brevi e frequenti.

   Si preferiscono cicli di sviluppo brevi, noti come sprint in SCRUM, che vanno da una settimana a un mese, con una preferenza per iterazioni brevi. Questi sprint consentono di ottenere feedback rapidi e migliorare continuamente il prodotto.

4. I membri del team di progetto e il cliente devono lavorare a contatto quotidianamente per favorire la comunicazione tra di essi.

5. Il Coordinatore si deve occupare di motivare e supportare i membri del team di progetto fornendo loro l'ambiente necessario per lavorare in maniera adeguata e autonoma, facendo sentir loro soddisfatti del loro operato.

6. Le informazioni relative al progetto devono essere scambiate all'interno dei team e con il cliente di persona dove è possibile al fine di ottenere una comunicazione più chiare e esaustiva possibile.

7. La misura del progresso viene valutata tramite prodotti funzionanti e non tramite ore di lavoro o qualsiasi altro tipo di documentazione.

8. Lo sviluppo del progetto deve svolgersi in maniera sostenibile per contenere il debito tecnico ed evitare che la modifica e l'estensione della soluzione diventino progressivamente più difficili nel tempo, senza creare un ambiente tossico o pericoloso per la salute fisica e mentale del personale.

9. Il progetto si concentra sull'ottenere eccellenza tecnologica e un design ottimale.

10. La semplicità è essenziale, tutto ciò che è superfluo rispetto agli obiettivi deve essere ignorato.

11. I membri del team devono discutere costruttivamente tra di loro al fine di migliorare continuamente la propria performance.

12. Riflessione e adattamento continuo.

    Riflettere su quanto fatto con l'obiettivo di migliorare le prestazioni del team e dell'adattamento continuo incoraggia il team a valutare costantemente le proprie prestazioni, a identificare punti di forza e debolezza, e ad apportare le necessarie modifiche per migliorare.

    Questo principio promuove l'apprendimento continuo e l'ottimizzazione delle pratiche di lavoro in modo da affrontare in modo più efficiente i prossimi sprints.

Negli approcci Agile, a differenza del modello a Waterfall, le attività non sono definite temporalmente in anticipo, ma è possibile tracciare il loro stato. Di conseguenza, l'uso di un diagramma temporale dettagliato per monitorare tutte le attività manca, poiché le attività non sono programmate in modo rigido ma vengono seguite attraverso un monitoraggio dello stato.

Questo comporta che la definizione delle attività non può più solo passare tramite il diagramma di Gantt, ma al Gantt stesso è necessario affiancare le dashboards che rappresentano concettualmente "lavagne" raffiguranti i processi che si svolgono per la realizzazione del progetto raccolti tramite tematiche, comunemente le lavagne vengono definite, nel gergo comune, impropriamente Kanban indipendentemente dal fatto che sia applicato il metodo Kanban, Scrum, o altro.

Ogni lavagna indica il punto in cui si trova l'attività di nostro interesse i cui stati possono essere :

1. definita
2. in via di sviluppo
3. conclusa
4. consegnata

2.2.2 Metodologie Agile Popolari

1. **Scrum**

   Scrum è un framework Agile che si basa su un approccio iterativo e incrementale per lo sviluppo del software, promuovendo la trasparenza, l'ispezione e l'adattamento continuo.

   Principi Fondamentali di Scrum:

   - **Trasparenza:** Tutti i dettagli rilevanti relativi al lavoro devono essere visibili a tutti i membri del team coinvolto nel progetto.
   - **Ispezione:** Scrum prevede frequenti ispezioni degli artefatti prodotti e dei progressi fatti verso gli obiettivi prefissati.
   - **Adattamento:** Scrum si basa sull'idea di adattarsi ai cambiamenti. Il team può adattare strategie, piani e attività in modo efficace per affrontare cambiamenti nelle esigenze del progetto.
   - **Ruoli definiti:** Scrum prevede ruoli chiari e ben definiti all'interno del team, come il Product Owner, lo Scrum Master e il Team di Sviluppo.
   - **Artefatti:** Scrum utilizza artefatti come il Product Backlog, lo Sprint Backlog e il Product Increment per gestire il lavoro e il progresso.
   - **Time-boxing:** Scrum divide il lavoro in intervalli temporali definiti chiamati sprint, di solito della durata di 1-4 settimane.
   - **Iterazione e miglioramento continuo:** Dopo ogni sprint, il team riflette su ciò che è stato fatto e su come migliorare sia i risultati sia il processo di lavoro per gli sprint futuri.
   - **Focus sulla consegna di valore:** Scrum pone un'enfasi significativa sulla consegna continua di valore per il cliente.

Lifecycle tipico di Scrum:

Il lifecycle tipico di Scrum è un processo iterativo e incrementale utilizzato per gestire e sviluppare prodotti complessi. Scrum si basa su principi fondamentali come la trasparenza, l'ispezione e l'adattamento continuo per massimizzare il valore del prodotto. Ecco una panoramica del lifecycle di Scrum:

(a) **Pianificazione**: All'inizio del ciclo di sviluppo, il Product Owner (responsabile del prodotto) e il team di sviluppo collaborano per definire gli obiettivi del progetto e creare un Product Backlog, che è una lista priorizzata di tutte le funzionalità, miglioramenti e correzioni desiderate per il prodotto.

(b) **Sprint Planning**: Alla fine di ogni sprint, il team di sviluppo e il Product Owner pianificano il lavoro per il prossimo sprint durante la Sprint Planning. Vengono selezionate le attività dal Product Backlog per essere completate durante lo sprint successivo, tenendo conto delle risorse disponibili e delle priorità.

(c) **Sprint**: Lo sprint è il cuore del lifecycle di Scrum. Si tratta di un periodo di tempo fisso, di solito da una a quattro settimane, durante il quale il team di sviluppo lavora per consegnare un incremento di funzionalità utilizzabili del prodotto. Durante lo sprint, il team tiene riunioni giornaliere di 15 minuti chiamate Daily Stand-ups per monitorare il progresso e affrontare eventuali ostacoli.

(d) **Sprint Review**: Alla fine di ogni sprint, il team di sviluppo condivide i risultati del lavoro svolto con gli stakeholder durante la Sprint Review. Durante questa riunione, vengono mostrati i risultati ottenuti durante lo sprint e vengono raccolti feedback per il miglioramento continuo del prodotto.

(e) **Retrospettiva dello Sprint**: Dopo la Sprint Review, il team di sviluppo tiene una Sprint Retrospective per riflettere sullo sprint appena concluso. Durante questa riunione, il team esamina ciò che è andato bene, ciò che potrebbe essere migliorato e identifica azioni per apportare miglioramenti nel prossimo sprint.

Il ciclo continua con la pianificazione del prossimo sprint, mantenendo un flusso continuo di sviluppo iterativo e incrementale fino al completamento del prodotto.

2. **Kanban**

   Kanban è un metodo di gestione visuale del lavoro, liberamente ispirato dal processo di ottimizzazione della produzione introdotto da Toyota.

   Si basa sull'idea di utilizzare una tavola Kanban suddivisa in colonne che rappresentano lo stato di avanzamento delle attività.

Ogni colonna della tavola Kanban indica uno stato specifico delle attività, consentendo a tutti i membri del team di visualizzare rapidamente cosa è in corso, quali attività sono in coda e quali sono state completate.

Principi fondamentali di Kanban:

- **Visualizzazione del lavoro**: Utilizzando una tavola Kanban, il lavoro è visualizzato in modo chiaro, mostrando tutte le attività e il loro stato attuale.
- **Limitazione del lavoro in corso (WIP Limit)**: Si limita il numero massimo di attività che possono essere in corso contemporaneamente, evitando sovraccarichi e migliorando il flusso di lavoro.
- **Gestione del flusso di lavoro**: Si mira a massimizzare il flusso di lavoro, consentendo al lavoro di avanzare costantemente attraverso le fasi senza accumuli eccessivi.
- **Feedback e miglioramento continuo**: Kanban incoraggia il team a identificare problemi, inefficienze e aree di miglioramento per apportare regolari miglioramenti al processo.

Struttura della tavola Kanban:

Una tavola Kanban è composta da colonne che rappresentano diversi stati delle attività.

Ad esempio, le colonne potrebbero essere "Da fare", "In corso", "In revisione" e "Completate".

Le carte (o post-it) rappresentano singole attività e si spostano attraverso le colonne in base al loro stato di avanzamento.

Kanban è flessibile e può essere adattato a vari contesti e tipi di lavoro, fornendo una maggiore visibilità, controllo e flessibilità nei tempi di consegna.

3. **XP (Extreme Programming)**

   Extreme Programming (XP) è un approccio Agile che si focalizza sulla produzione di software di alta qualità, integrando pratiche e valori per garantire la consegna di prodotti affidabili e adattabili ai cambiamenti dei requisiti.

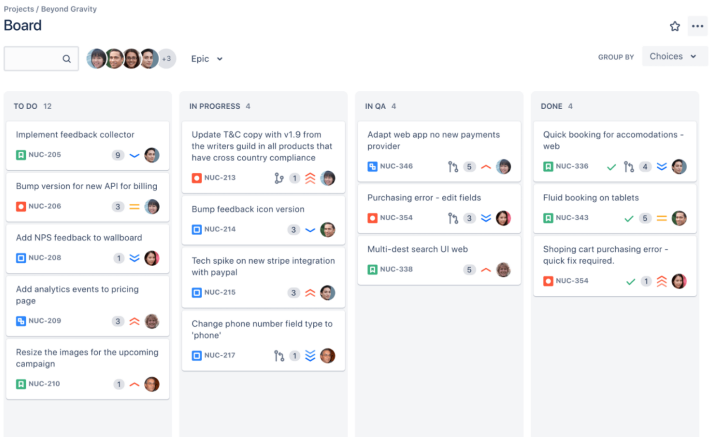

Figura 2.2: Tavola Kanban di Jira

Principi fondamentali di Extreme Programming:

XP si basa su valori e principi che guidano il suo approccio allo sviluppo software:

- **Comunicazione:** Favorire la comunicazione costante tra i membri del team per garantire una comprensione condivisa degli obiettivi e dei requisiti.
- **Feedback:** Ricevere feedback tempestivo, sia dai clienti che dal team, per adattare rapidamente il prodotto e il processo di sviluppo.
- **Semplicità:** Sviluppare soluzioni semplici, evitando l'eccessiva complessità, per massimizzare la comprensione e la manutenibilità del software.
- **Coraggio:** Essere disposti a prendere decisioni coraggiose e ad affrontare i problemi apertamente.
- **Rispetto:** Rispettare le opinioni, le competenze e le prospettive di ogni membro del team, promuovendo un ambiente di lavoro collaborativo e inclusivo.

Pratiche di Extreme Programming:

XP incorpora diverse pratiche per garantire la qualità del software e il miglioramento continuo:

- **Pair Programming**: Due programmatori lavorano insieme su un unico computer, migliorando la qualità del codice, condividendo conoscenze e riducendo gli errori.

- **Test-Driven Development (TDD)**: Scrivere test automatici prima di scrivere il codice, assicurando che il software funzioni correttamente e sia facilmente testabile.
- **Gioco di pianificazione (Planning Game)**: Coinvolgere attivamente il cliente nella pianificazione del lavoro, stabilendo priorità e definendo le funzionalità da sviluppare.
- **Refactoring**: Ristrutturare il codice senza modificarne il comportamento esterno per migliorare la qualità, la manutenibilità e la leggibilità.
- **Integrazione continua**: Integrare frequentemente il codice nel repository con test automatici, garantendo che il software rimanga funzionale e pronto per il rilascio in ogni momento.

Lifecycle tipico di Extreme Programming:

Extreme Programming (XP) è una metodologia di sviluppo software agile che si concentra sulla produzione di software di alta qualità e sulla massimizzazione del valore per il cliente. Il lifecycle di Extreme Programming è caratterizzato da cicli di sviluppo rapidi e iterativi, con un forte focus sulla collaborazione tra i membri del team e il coinvolgimento del cliente. Ecco le fasi principali del lifecycle di Extreme Programming:

(a) **Pianificazione**: In questa fase, il team di sviluppo e il cliente collaborano per identificare e prioritizzare i requisiti del progetto. Viene creata una lista di user stories, che rappresentano le funzionalità desiderate dal punto di vista dell'utente.

(b) **Analisi**: Durante questa fase, il team analizza le user stories per comprendere appieno i requisiti e le necessità del cliente. Viene effettuata una stima del lavoro necessario per completare ciascuna user story.

(c) **Progettazione**: Il team progetta e implementa le funzionalità del software in piccoli incrementi, utilizzando pratiche come il pair programming e la programmazione test-driven (TDD). L'obiettivo è produrre codice di alta qualità che sia facilmente mantenibile e estendibile.

(d) **Test**: Il testing è integrato in tutto il processo di sviluppo. Vengono scritti test automatizzati per verificare il corretto funzionamento delle funzionalità implementate e garantire che eventuali modifiche non introducano regressioni nel software.

(e) **Rilascio**: Il software viene rilasciato in piccoli incrementi, con una frequenza che può variare da settimane a pochi mesi. Questo consente al cliente di vedere rapidamente i risultati del lavoro del team e fornire feedback tempestivo.

(f) **Feedback**: Dopo il rilascio, il team raccoglie il feedback del cliente e lo utilizza per guidare lo sviluppo futuro del software. Questo ciclo di feedback continuo aiuta il team a adattare rapidamente il prodotto alle esigenze e alle preferenze del cliente.

Il ciclo continua con iterazioni successive, con il team che lavora in modo collaborativo e iterativo per consegnare valore al cliente in modo rapido e continuo.

Benefici di Extreme Programming:

XP promuove una maggiore qualità del software, una maggiore adattabilità ai cambiamenti dei requisiti e una maggiore soddisfazione del cliente attraverso la consegna continua di valore.

Vantaggi e Svantaggi

Tra i vantaggi principali troviamo:

1. **Adattabilità ai cambiamenti dei requisiti**

   L'Agile consente di adattarsi facilmente ai cambiamenti dei requisiti, garantendo un software che risponde alle reali esigenze degli utenti.

2. **Riduzione dei rischi**

   La consegna frequente di funzionalità operative permette di identificare e affrontare precocemente i problemi, riducendo i rischi nel corso dello sviluppo.

3. **Coinvolgimento del cliente**

   La collaborazione continua con il cliente durante il processo di sviluppo assicura la conformità del prodotto finale alle aspettative del cliente.

Invece tra gli svantaggi identifichiamo:

1. **Complessità nell'implementazione**

   L'adozione dell'Agile richiede una mentalità e una cultura organizzativa che possono risultare difficili da implementare in alcune realtà aziendali.

2. **Necessità di coinvolgimento costante**

   L'Agile richiede un coinvolgimento costante e una comunicazione attiva, il che potrebbe essere difficile in team distribuiti geograficamente o con diversi fusi orari.

3. **Mancanza di documentazione dettagliata**

   L'orientamento verso il software funzionante può portare a una ridotta documentazione, il che potrebbe generare problemi di tracciabilità e comprensione per team futuri o per i cambiamenti di personale.

4. **Adattabilità alle grandi organizzazioni:** L'Agile funziona meglio in progetti di piccole o medie dimensioni, ma può incontrare difficoltà nell'essere implementato efficacemente in grandi organizzazioni a causa della complessità e delle gerarchie presenti.

5. **Necessità di coinvolgimento totale:** Il successo dell'Agile dipende dal coinvolgimento attivo e dalla collaborazione di tutti i membri del team e degli stakeholder.

   Il mancato coinvolgimento di alcuni membri o dipartimenti può costituire un ostacolo significativo.

6. **Difficoltà nella misurazione dei progressi:** Alcuni critici sostengono che le metodologie Agile non offrano metriche di misurazione del progresso precise come altre metodologie più tradizionali, rendendo difficile comunicare il progresso del progetto a parti interessate esterne.

7. **Resistenza al cambiamento:** L'adozione dell'Agile può essere ostacolata dalla resistenza al cambiamento in organizzazioni abituate a pratiche di sviluppo tradizionali, con membri del team o dirigenti poco propensi a modificare i loro processi consolidati.

8. **Mantenimento del focus sulla qualità:** L'attenzione all'accelerazione del processo di sviluppo potrebbe portare a una minore enfasi sulla qualità del software se non vengono rispettate le pratiche chiave dell'Agile come il test-driven development e il pair programming.

9. **Overhead di comunicazione:** La necessità di comunicare costantemente e collaborare può aumentare il carico di lavoro e il tempo dedicato alle riunioni, portando a un eccessivo overhead di comunicazione e a una minore efficienza se non gestito correttamente.

10. **Scarsa documentazione:** L'Agile promuove il lavoro sul software funzionante piuttosto che sulla documentazione esaustiva.

    Questo potrebbe portare a una documentazione insufficiente, rendendo difficile per i nuovi membri del team o per le parti interessate comprendere il software. Inoltre, mantenere la documentazione aggiornata con le modifiche implementate può risultare complesso e dispendioso in termini sia temporali che di effort richiesto .

11. **Esigenze di formazione e coaching:** L'adozione dell'Agile richiede spesso una curva di apprendimento e un supporto adeguati.

    Il team potrebbe necessitare di formazione e coaching continuo per comprendere e adottare pienamente i principi e le pratiche Agile.

2.2.3 Implementazioni nelle realtà aziendali

La metodologia Agile ad oggi è estremamente utilizzata nell'industria visto gli oggettivi successi ottenuti grazie al suo utilizzo.

Tra i più noti nomi possiamo trovare:

1. **Spotify**: La piattaforma di streaming musicale utilizza metodologie Agili per lo sviluppo del software, organizzando i team in strutture agili come Squad e Tribe. Le **Squad** sono piccoli team autonomi, ognuno responsabile di una specifica area del prodotto. Ogni Squad ha tutte le competenze necessarie per progettare, sviluppare, testare e rilasciare funzionalità. Le **Tribe**, invece, sono gruppi di più Squad che lavorano su aree correlate del prodotto. Questo approccio permette a Spotify di mantenere l'agilità e la velocità di una startup, nonostante la grande dimensione dell'azienda.

2. **Google**: Diverse divisioni di Google adottano Scrum e Kanban per sviluppare e migliorare i loro prodotti. Utilizzando **Scrum**, i team lavorano in sprint di due settimane, durante i quali pianificano, eseguono e rivedono il lavoro. Questa metodologia aiuta Google a mantenere un ritmo costante di rilasci e miglioramenti continui. Con **Kanban**, invece, i team possono visualizzare il flusso di lavoro, identificare i colli di bottiglia e ottimizzare i processi. Questo è particolarmente utile per gestire i progetti che richiedono molta flessibilità e adattamento continuo.

3. **Amazon**: Amazon utilizza metodologie Agili per favorire la rapidità nell'implementare nuove funzionalità e rispondere alle esigenze dei clienti in modo più rapido. Un esempio di approccio agile in Amazon è l'uso del concetto di **"Two-Pizza Team"**, dove ogni team è abbastanza piccolo da essere alimentato da due pizze. Questi team sono autonomi e responsabili del ciclo completo di sviluppo del prodotto, dalla progettazione al rilascio. Inoltre, Amazon promuove una cultura di **"Continuous Deployment"**, dove il software viene rilasciato frequentemente e automaticamente, riducendo il tempo tra l'idea e la realizzazione.

4. **Microsoft**: Microsoft ha integrato approcci agili in molte delle sue divisioni, accelerando il ciclo di sviluppo e migliorando la qualità del software. Ad esempio, il team di **Visual Studio** utilizza Scrum per gestire lo sviluppo del software. Questo include pianificazione di sprint, daily stand-up meeting, review e retrospective,

che aiutano il team a migliorare continuamente i processi e i prodotti. L'adozione di pratiche come il **Continuous Integration** e il **Continuous Delivery** ha permesso a Microsoft di rilasciare aggiornamenti e nuove funzionalità in modo più frequente e affidabile.

5. **Adobe**: Adobe utilizza metodologie Agili come Scrum e Kanban per lo sviluppo dei propri prodotti software, consentendo rilasci più frequenti e mirati. Nel team di **Creative Cloud**, ad esempio, Scrum viene utilizzato per gestire i cicli di sviluppo. Ogni sprint si conclude con un **demo** in cui i team mostrano le funzionalità completate ai vari stakeholder, ottenendo feedback immediato e costruttivo. Kanban viene utilizzato per la gestione del supporto e della manutenzione dei prodotti, aiutando a visualizzare le attività in corso e a ottimizzare il flusso di lavoro.

6. **Zalando**: L'azienda di moda online Zalando adotta approcci Agili come Scrum per favorire un ambiente di lavoro più collaborativo e adattabile. I team di Zalando sono organizzati in **cross-functional teams**, ognuno responsabile di specifiche aree del business. Questo permette una maggiore autonomia e una responsabilità diretta sui risultati. Inoltre, Zalando utilizza **OKR** (Objectives and Key Results) in combinazione con metodologie agili per allineare gli obiettivi strategici dell'azienda con le attività dei team, garantendo che tutti lavorino verso gli stessi obiettivi.

Capitolo 3

Scrum una metodologia Agile

La guida di riferimento nel Project Management definisce il framework di un progetto come la struttura di base che fornisce un approccio generale e una metodologia per comprendere, pianificare, eseguire e gestire progetti.

Si tratta di un insieme di processi, pratiche, strumenti e concetti organizzativi che costituiscono la struttura di base su cui si basa la gestione e l'esecuzione di un progetto. Il framework offre linee guida e direzioni per organizzare e gestire le attività, definendo ruoli, responsabilità e procedure per il raggiungimento degli obiettivi del progetto. In sostanza, costituisce il fondamento su cui viene costruito e gestito il progetto. [3]

La principale differenza che si ha confrontando il Framework con le metodologie progettuali è che questa seconda definisce i principi, i valori e le best practices che fanno parte del progetto quindi concretamente che cosa si vuole ottenere con il progetto su cui si sta lavorando mentre il Framework invece definisce come ottenere il progetto descritto nella metodologia.

Un Framework che cerca di standardizzare prevede principalmente tre aree principali di interesse su cui bisogna lavorare:

1. **Project Life Cycle**
2. **Templates, Checklist e altri strumenti**
3. **Processi e attività**

3.1 Scrum

Per la metodologia Agile sono stati creati numerosi Framework con lo scopo di rendere più semplice la gestione dei progetti in Agile in un modo più semplice rispetto all'uso della metodologia tradizionale, ogni Framework contiene tutti i dettagli relativi alle fasi e alle attività del progetto che sono da seguire per la sua realizzazione e sono accomunate

nel supportare i *quattro valori* chiave e i *dodici principi* su cui si fonda la metodologia Agile.

I Framework della metodologia Agile più famosi come accennato nel capitolo precedente sono Kanban, Extreme Programming (XP), Feature-driven development (FDD), Dynamic Systems Development Method (DSDM), Crystal e Scrum.

3.1.1 Principi e Valori di Scrum

Scrum è una metodologia Agile che ha radici nella gestione dei progetti e nello sviluppo software.

La sua storia affonda le radici nel lavoro di Jeff Sutherland e Ken Schwaber, i quali, nel corso degli anni '90, svilupparono e formalizzarono questa metodologia, prendendo ispirazione da varie fonti, tra cui le teorie empiriche di controllo dei processi di produzione e i modelli di gestione del lavoro a squadre.

L'idea di Scrum emerse per la prima volta nei sistemi di sviluppo software come un approccio che promuoveva la flessibilità, l'adattabilità e la collaborazione interfunzionale.

Il termine "Scrum" deriva dal rugby, dove rappresenta un'azione nella quale i giocatori si uniscono in un cerchio per riprendere il gioco. Questo concetto di lavoro collaborativo e interdipendente è stato trasferito nell'ambito del project management per sottolineare l'importanza della collaborazione all'interno del team.

Nel corso degli anni, Scrum è cresciuto in popolarità non solo nel settore dello sviluppo software, ma anche in altre aree come la gestione dei progetti, la produzione e persino la gestione aziendale. La sua adozione è stata in gran parte guidata dai suoi principi chiave:

1. Trasparenza
2. Ispezione
3. Adattamento

che si riflettono nel modo in cui affronta i cambiamenti, gestisce i rischi e mantiene la focalizzazione sugli obiettivi.

Scrum si basa su iterazioni di breve durata chiamate **"Sprint"**, solitamente della durata di 1-4 settimane, durante le quali si pianifica, si sviluppa e si consegna un incremento di prodotto funzionante.

Questa struttura permette al team di adattarsi rapidamente ai cambiamenti dei requisiti o alle nuove informazioni, migliorando continuamente il prodotto in base al feedback.

Un aspetto fondamentale di Scrum è la sua struttura di ruoli definiti chiaramente:

1. Product Owner

2. Scrum Master

3. Team di Sviluppo

Questi ruoli hanno responsabilità specifiche che favoriscono la collaborazione e la chiarezza delle aspettative all'interno del team.

Oltre ai ruoli, Scrum si avvale di cerimonie regolari come

1. Sprint Planning

2. Daily Scrum

3. Sprint Review

4. Sprint Retrospective

tutte queste cerimonie permettono al team di sincronizzarsi, valutare i progressi, rivedere il lavoro svolto e pianificare miglioramenti per il futuro.

Gli artefatti principali di Scrum includono il **Product Backlog**, che elenca tutte le richieste di funzionalità, lo **Sprint Backlog**, che dettaglia le attività per la Sprint corrente, e **l'Incremento**, che rappresenta il lavoro completato alla fine di ogni Sprint.

L'adozione di Scrum ha portato a numerosi vantaggi, come una maggiore adattabilità ai cambiamenti, una migliore soddisfazione del cliente, un'accelerazione della consegna del prodotto e una maggiore efficacia nel lavoro di squadra.

Principi di Scrum

I principi principali di Scrum comprendono:

- **Trasparenza:** Questo principio sottolinea l'importanza della chiara comunicazione di informazioni rilevanti a tutti gli interessati.

  La trasparenza permette a tutti i membri del team di avere una visione comune del lavoro in corso, delle sfide e degli obiettivi, favorendo una maggiore comprensione e collaborazione.

- **Ispezione:** La pratica dell'ispezione comporta un'analisi regolare dei progressi compiuti durante lo sviluppo del prodotto.

  Questo consente di identificare variazioni, problemi o inefficienze in modo tempestivo, permettendo al team di adattare e migliorare continuamente il proprio lavoro.

- **Adattamento:** Scrum promuove la flessibilità nel modificare le strategie o il piano di lavoro per affrontare cambiamenti imprevisti o per rispondere a nuove informazioni.

  Questo principio consente al team di adattarsi prontamente alle mutevoli esigenze e ai requisiti in evoluzione, massimizzando il valore del prodotto sviluppato.

Valori di Scrum

I valori cardine di Scrum sono:

- **Coraggio:** Nell'ambito di Scrum, il coraggio sprona il team a prendere decisioni difficili ma necessarie per il successo del progetto.

  Ciò può includere la capacità di affrontare problemi complessi o di rivedere pratiche consolidate per migliorare continuamente.

- **Impegno:** Scrum richiede una dedizione completa e una responsabilità individuale e di gruppo per raggiungere gli obiettivi prefissati.

  Gli individui si impegnano a rispettare gli obiettivi stabiliti durante lo Sprint e a contribuire attivamente al successo del team.

- **Apertura:** Questo valore promuove una comunicazione sincera e trasparente all'interno del team. Un ambiente aperto incoraggia la condivisione delle informazioni, delle sfide e delle opinioni senza timore di critiche o giudizi negativi.

- **Rispetto:** Un elemento cruciale di Scrum è il rispetto reciproco.

  Questo valore sottolinea l'importanza di un ambiente in cui tutti i membri del team sono valorizzati, dove le opinioni di ciascuno sono ascoltate e considerate.

- **Focalizzazione:** Il valore della focalizzazione richiede al team di concentrarsi sugli obiettivi definiti durante lo Sprint, evitando distrazioni e mantenendo l'attenzione su ciò che è più importante per il successo del prodotto.

3.1.2 Ruoli in Scrum

Scrum definisce chiaramente diversi ruoli all'interno del processo:

Product Owner

Il Product Owner è responsabile della definizione delle funzionalità del prodotto, della gestione del backlog del prodotto e delle priorità.

È colui che comprende le esigenze del cliente e del mercato, traducendole in un backlog di lavoro che riflette le priorità aziendali.

Il Product Owner è quindi responsabile di:

- **Definizione e Comunicazione della Visione del Prodotto:** Il Product Owner è responsabile di articolare chiaramente la visione del prodotto.

  Questo coinvolge la comprensione dei bisogni del cliente, delle aspettative di mercato e degli obiettivi aziendali.

  Comunicare questa visione al team è essenziale per allinearli e mantenere un focus comune durante lo sviluppo.

- **Prioritizzazione e Gestione del Product Backlog:** Gestisce il Product Backlog, l'elenco prioritario delle funzionalità, delle modifiche e degli aggiornamenti necessari per il prodotto.

  È responsabile di garantire che il backlog sia sempre aggiornato, ben organizzato e che rifletta le esigenze del cliente e le strategie aziendali.

- **Collaborazione con il Team per Definire e Accettare gli Elementi del Backlog:** Lavora a stretto contatto con il Team di Sviluppo e altre parti interessate per definire e dettagliare le user stories o le richieste di funzionalità.

  Collabora con il team per chiarire i requisiti e garantire che siano compresi correttamente prima che vengano sviluppati.

- **Assicurazione della Soddisfazione del Cliente e del Valore del Prodotto:** Agisce come voce del cliente all'interno del team.

  È responsabile di garantire che il prodotto finale soddisfi le esigenze e le aspettative del cliente, nonché di massimizzare il valore del prodotto per l'azienda e per gli utenti finali.

- **Fornitura di Feedback Continuo:** Durante lo sviluppo del prodotto, il Product Owner fornisce feedback costante al Team di Sviluppo.

  Questo feedback aiuta il team a comprendere meglio le esigenze del cliente e a effettuare aggiustamenti, se necessario.

Il Product Owner è fondamentale per garantire che il prodotto sviluppato risponda alle esigenze del cliente, generi valore per l'azienda e mantenga una visione chiara e allineata con gli obiettivi complessivi del progetto.

La sua capacità di comunicare in modo efficace, prendere decisioni informate e collaborare con il team sono essenziali per il successo del prodotto.

Scrum Master

Lo Scrum Master è il custode del processo Scrum, facilitando il suo corretto utilizzo all'interno del team e dell'organizzazione. È responsabile di rimuovere gli ostacoli che impediscono al team di raggiungere i propri obiettivi.

Lo Scrum Master è quindi responsabile di:

- **Assicurare che il team comprenda e segua i principi di Scrum:** Il Scrum Master è responsabile di garantire che il team comprenda appieno i principi, i valori e le pratiche di Scrum.

  Questo può implicare la formazione iniziale dei membri del team su Scrum, nonché l'assistenza costante nel mantenere il team allineato e conforme ai principi fondamentali di Scrum durante lo sviluppo del prodotto.

- **Rimuovere gli ostacoli che impediscono al team di raggiungere i propri obiettivi:** Uno degli aspetti più importanti del ruolo di Scrum Master è quello di eliminare gli ostacoli che potrebbero ostacolare il progresso del team verso gli obiettivi stabiliti per lo sprint o il progetto.

  Ciò può includere la risoluzione di problemi operativi, la rimozione di impedimenti esterni che influenzano il team o la facilitazione della comunicazione tra le parti interessate.

- **Organizzare le cerimonie di Scrum:** Il Scrum Master è responsabile della pianificazione e della gestione delle cerimonie di Scrum, tra cui

  - Sprint Planning (pianificazione dello sprint)
  - Daily Scrum (riunione quotidiana)
  - Sprint Review (revisione dello sprint)
  - Sprint Retrospective (retrospettiva dello sprint)

  Deve garantire che queste cerimonie si svolgano in modo efficace, che siano focalizzate sugli obiettivi e che permettano al team di migliorare continuamente il proprio processo di sviluppo.

- **Favorire la collaborazione e la comunicazione tra i membri del team e gli altri stakeholders:** Il Scrum Master agisce come facilitatore della collaborazione e della comunicazione all'interno del team e con gli stakeholders esterni.

  Questo ruolo prevede di creare un ambiente in cui i membri del team possano lavorare insieme in modo efficace, incoraggiando la trasparenza e il dialogo aperto tra tutte le parti coinvolte nel progetto.

Team di Sviluppo

Il Team di Sviluppo è composto da individui che realizzano il lavoro richiesto per creare l'incremento di prodotto.

È un gruppo auto-organizzato e multidisciplinare che lavora insieme per raggiungere gli obiettivi dello Sprint.

Il Team di Sviluppo è quindi responsabile di:

- **Collaborare con il Product Owner per capire gli elementi del Product Backlog e fornire stime di lavoro:** I membri del team di sviluppo lavorano a stretto contatto con il Product Owner per comprendere le user stories o gli elementi del Product Backlog.

  Partecipano attivamente alle discussioni con il Product Owner per ottenere chiarezza sui requisiti e fornire stime sul tempo e le risorse necessarie per completare il lavoro.

- **Sviluppare, testare e consegnare incrementi di funzionalità potenzialmente rilasciabili:** Il team di sviluppo è responsabile dello sviluppo del software, inclusi il codice, i test e la consegna di incrementi di funzionalità del prodotto.

  Devono garantire che gli elementi del Product Backlog su cui stanno lavorando siano completati in modo da poter essere potenzialmente rilasciati al termine dello Sprint.

- **Auto-organizzarsi per determinare come completare il lavoro assegnato durante lo Sprint:** I membri del team di sviluppo hanno l'autonomia e la responsabilità di organizzare il proprio lavoro durante lo Sprint.

  Si auto-organizzano per determinare le attività da svolgere, le modalità di sviluppo e i metodi di test per completare con successo gli elementi assegnati durante lo Sprint Planning.

- **Assumersi la responsabilità collettiva per il completamento degli obiettivi dello Sprint:** Tutti i membri del team di sviluppo sono responsabili collettivamente del completamento degli obiettivi dello Sprint.

  Si impegnano a lavorare insieme per consegnare gli incrementi di funzionalità concordati entro la fine dello Sprint, garantendo che l'intero team si assuma la responsabilità per il successo del lavoro.

3.1.3 Cerimonie e Artefatti di Scrum

Le cerimonie in Scrum sono momenti chiave nel processo di sviluppo che consentono al team di pianificare, sincronizzare, valutare e migliorare costantemente il proprio lavoro.

3.1.4 Cerimonie in Scrum

Sprint Planning

La Sprint Planning è una cerimonia che si tiene all'inizio di ogni Sprint.

Durante la Sprint Planning, solitamente divisa in due parti, il Product Owner e il Team di Sviluppo collaborano per raggiungere una chiara comprensione degli obiettivi dello Sprint e per identificare quali user stories o elementi del Product Backlog verranno sviluppati durante lo Sprint stesso.

La prima parte della Sprint Planning coinvolge la discussione e la comprensione approfondita delle user stories e degli elementi del Product Backlog che possono essere sviluppati durante lo Sprint. Il Product Owner condivide la visione e le priorità del backlog, mentre il Team di Sviluppo si impegna a capire appieno i requisiti e le aspettative.

Vengono posti domande, chiarimenti e vengono fornite stime iniziali sul lavoro richiesto per ogni elemento.

Nella seconda parte della Sprint Planning, il Team di Sviluppo si auto-organizza per definire come affrontare e completare gli elementi selezionati durante la discussione iniziale.

Si discute sui task specifici necessari, sulle attività da svolgere e su come l'incremento di funzionalità pianificato verrà sviluppato e testato.

Questa parte si conclude con il Team di Sviluppo impegnato a consegnare un incremento di funzionalità potenzialmente rilasciabile entro la fine dello Sprint.

È importante notare che durante la Sprint Planning, il focus è sulla collaborazione e sulla comprensione condivisa degli obiettivi dello Sprint tra il Product Owner e il Team di Sviluppo.

Si tratta di una cerimonia interattiva che mira a garantire che tutti abbiano una chiara visione degli obiettivi da raggiungere e del lavoro da svolgere durante lo Sprint successivo.

Daily Scrum

Il Daily Scrum è un incontro giornaliero breve, della durata tipica di 15 minuti, in cui tutto il Team di Sviluppo si riunisce per sincronizzare il lavoro.

Durante questa riunione, ogni membro del team risponde a tre domande fondamentali:

- **Cosa hai fatto ieri?** I membri del team condividono brevemente le attività che hanno completato nell'ultima giornata lavorativa.

  Questo non dovrebbe essere un aggiornamento dettagliato, ma piuttosto un rapido riepilogo per informare gli altri membri sul proprio progresso.

- **Cosa farai oggi?** Ogni membro del team esprime le attività o i compiti che prevede di svolgere durante la giornata in corso.

Questo aiuta a informare gli altri membri sulle prossime attività e sull'eventuale impatto che potrebbero avere sul lavoro degli altri.

- **Ci sono ostacoli?** I membri del team condividono eventuali ostacoli o impedimenti che stanno incontrando durante lo svolgimento del lavoro.

  Questo può includere problemi tecnici, dipendenze da altri membri del team o qualsiasi altro elemento che possa rallentare il progresso.

Il Daily Scrum non è una riunione per risolvere i problemi, ma piuttosto per identificare gli ostacoli e individuare eventuali situazioni che richiedono attenzione ulteriore.

Il suo scopo principale è mantenere tutti i membri del team informati sul progresso del lavoro, garantire la trasparenza sulle attività svolte e pianificate e identificare rapidamente eventuali impedimenti che potrebbero ostacolare il completamento dello Sprint.

Sprint Review

La Sprint Review è una cerimonia che avviene alla fine di uno Sprint in Scrum.

Durante questa riunione, il Team di Sviluppo presenta il lavoro completato al Product Owner e agli altri stakeholder per ottenere feedback.

- **Valutare il lavoro svolto:** Il Team di Sviluppo mostra in modo dimostrativo il lavoro completato durante lo Sprint.

  Questo può includere le funzionalità sviluppate, i miglioramenti apportati e altri elementi realizzati durante lo Sprint.

- **Ottenere feedback:** Durante la presentazione del lavoro, il Team di Sviluppo cerca feedback dal Product Owner e dagli altri partecipanti sulla qualità del lavoro svolto.

  Si cerca di capire se il lavoro è allineato alle aspettative e agli obiettivi stabiliti per lo Sprint.

- **Verificare l'allineamento agli obiettivi:** L'obiettivo principale è assicurarsi che il lavoro svolto sia allineato agli obiettivi e alle aspettative stabiliti all'inizio dello Sprint.

  Si verifica che il lavoro sia conforme alle specifiche richieste e che sia in linea con la visione del prodotto.

La Sprint Review è un'opportunità per il Team di Sviluppo di presentare il risultato del proprio lavoro, ricevere feedback preziosi e assicurarsi che il prodotto stia progredendo nella giusta direzione.

L'interazione con il Product Owner e gli altri stakeholder durante questa cerimonia aiuta a garantire la trasparenza e l'allineamento del lavoro svolto agli obiettivi del progetto.

Sprint Retrospective

La Sprint Retrospective è una cerimonia che segue immediatamente la Sprint Review in Scrum.

Durante questa riunione, il Team di Sviluppo riflette sul processo di sviluppo che è stato seguito durante lo Sprint appena concluso.

Obiettivi della Sprint Retrospective:

- **Riflettere sull'esperienza dello Sprint:** Il Team di Sviluppo esamina l'esperienza vissuta durante lo Sprint, identificando cosa ha funzionato bene, cosa può essere migliorato e cosa ha funzionato male nel processo di sviluppo.
- **Identificare miglioramenti e suggerire modifiche:** Si discute apertamente sui punti di forza e sui problemi riscontrati durante lo Sprint.

  Vengono suggerite azioni concrete per migliorare il processo, risolvere eventuali problemi o superare le sfide affrontate.
- **Promuovere il miglioramento continuo:** L'obiettivo principale è promuovere un ambiente di miglioramento continuo.

  Il focus è sull'evoluzione del team e del processo, cercando costantemente di diventare più efficienti e di superare le sfide incontrate durante lo sviluppo.

Durante la Sprint Retrospective, si incoraggia la partecipazione attiva di tutti i membri del Team di Sviluppo per condividere le proprie opinioni e contribuire alle discussioni.

Si cerca di creare uno spazio sicuro dove i membri del team possono esprimere liberamente le proprie idee e collaborare per identificare soluzioni migliori.

La Sprint Retrospective è fondamentale per il processo di miglioramento continuo all'interno del team Scrum, consentendo di apprendere dagli errori, capitalizzare sui successi e adattare il processo per massimizzare l'efficienza e l'efficacia del lavoro svolto durante gli Sprint successivi.

3.1.5 Artefatti di Scrum

Gli artefatti in Scrum sono elementi fondamentali che forniscono trasparenza e strumenti per la pianificazione, l'esecuzione e la valutazione del lavoro.

Product Backlog

Il Product Backlog è un elemento chiave in Scrum, rappresentando un elenco dinamico e priorizzato di tutte le richieste di funzionalità, modifiche e miglioramenti che devono essere implementati nel prodotto.

Caratteristiche del Product Backlog:

- **Elenco dinamico e priorizzato:** Il Product Backlog è un elenco in costante evoluzione che contiene tutte le richieste e i requisiti che devono essere sviluppati.

  È gestito dal Product Owner, che definisce, aggiorna e priorizza gli elementi in base alla loro importanza, alle necessità del cliente e al valore aziendale.

- **Riflette l'evoluzione delle esigenze del prodotto:** Il Product Backlog è soggetto a continui cambiamenti per riflettere l'evoluzione delle esigenze del prodotto.

  Nuove richieste possono essere aggiunte, i requisiti possono essere modificati o rimossi, e la priorità degli elementi può essere aggiornata per adattarsi alle esigenze emergenti.

- **Responsabilità del Product Owner:** Il Product Owner è responsabile della gestione del Product Backlog.

  Questo include l'aggiornamento costante degli elementi, la definizione chiara delle user stories o dei requisiti e la priorizzazione degli elementi in base al valore che apportano al prodotto.

Il Product Backlog funge da guida per il lavoro futuro del Team di Sviluppo, fornendo una visione chiara delle richieste e delle necessità del prodotto.

È uno strumento fondamentale per assicurare che il lavoro svolto sia allineato agli obiettivi del prodotto e possa massimizzare il valore per il cliente e per l'azienda.

Sprint Backlog

Lo Sprint Backlog è un elemento operativo cruciale in Scrum, costituito da un insieme di attività selezionate dal Product Backlog per essere completate durante una specifica Sprint.

Caratteristiche dello Sprint Backlog:

- **Selezione di attività definite:** Lo Sprint Backlog è composto da elementi chiaramente definiti e suddivisi che il Team di Sviluppo si impegna a portare a termine entro la fine dello Sprint.

  Questi elementi possono essere user stories, task o altre attività definite.

- **Impegno del Team di Sviluppo:** Durante lo Sprint Planning, il Team di Sviluppo seleziona gli elementi dal Product Backlog e li trasforma in uno Sprint Backlog impegnandosi a completarli entro la fine dello Sprint.

  Questo impegno è fondamentale per il raggiungimento degli obiettivi stabiliti per lo Sprint.

- **Dinamico e adattabile:** Lo Sprint Backlog è dinamico e può essere adattato durante lo Sprint per affrontare cambiamenti, nuove informazioni o imprevisti.

  Se emergono nuove esigenze o se si verificano cambiamenti durante lo sviluppo, lo Sprint Backlog può essere modificato per riflettere la situazione attuale.

Lo Sprint Backlog rappresenta il lavoro concreto e dettagliato che il Team di Sviluppo si impegna a svolgere durante lo Sprint.

È uno strumento operativo che fornisce una guida quotidiana al lavoro e assicura che il team sia focalizzato sugli obiettivi e sugli elementi prioritari per il successo dello Sprint.

Incremento

L'Incremento rappresenta il risultato del lavoro completato alla fine di ogni Sprint.

È un componente fondamentale in Scrum, essendo la somma di tutte le attività portate a termine dal Team di Sviluppo durante lo Sprint.

Caratteristiche dell'Incremento:

- **Risultato tangibile dello Sprint:** L'Incremento è il risultato concreto del lavoro svolto dal Team di Sviluppo durante lo Sprint.

  È il prodotto finale dell'attività di sviluppo, rappresentando le funzionalità sviluppate o miglioramenti apportati al prodotto.

- **Evoluzione del prodotto:** L'Incremento contribuisce all'evoluzione tangibile del prodotto.

  Aggiunge nuove funzionalità o migliora quelle esistenti, aumentando il valore complessivo del prodotto.

- **Potenzialmente rilasciabile:** L'Incremento deve essere potenzialmente rilasciabile, il che significa che è pronto per essere consegnato o presentato agli stakeholder.

  Se necessario, può essere rilasciato in qualsiasi momento, aggiungendo valore al prodotto o al cliente.

L'Incremento rappresenta una milestone importante in Scrum, poiché ogni Sprint dovrebbe generare un Incremento aggiornato e funzionante del prodotto.

È una dimostrazione tangibile del progresso del lavoro e rappresenta un passo verso il completamento del prodotto finale.

3.2 Vantaggi e Svantaggi di Scrum

Scrum offre una serie di vantaggi che contribuiscono alla realizzazione di prodotti di successo e al miglioramento continuo dei processi di sviluppo.

3.2.1 Vantaggi di Scrum

- **Maggiore flessibilità nel rispondere ai cambiamenti:** Scrum è progettato per adattarsi facilmente a cambiamenti nei requisiti, nelle priorità o nelle condizioni di mercato.

  La struttura iterativa e incrementale consente al team di rispondere rapidamente e in modo flessibile alle modifiche, garantendo che il prodotto finale soddisfi meglio le esigenze del cliente.

- **Coinvolgimento continuo del cliente:** Scrum favorisce un coinvolgimento costante del cliente nel processo di sviluppo.

  Attraverso le cerimonie come la Sprint Review e il coinvolgimento del Product Owner, il cliente può fornire feedback regolare e guidare l'evoluzione del prodotto, assicurandosi che soddisfi le sue esigenze e aspettative.

- **Maggiore trasparenza nel processo di sviluppo:** Scrum promuove la trasparenza nel lavoro svolto.

  Attraverso i diversi artefatti e le cerimonie come la Sprint Review e la Sprint Retrospective, tutti gli interessati ottengono una visione chiara del progresso del lavoro e delle sfide affrontate, facilitando la comunicazione e la collaborazione all'interno del team e con gli stakeholder esterni.

3.2.2 Svantaggi di Scrum

- **Complessità iniziale:** L'adozione iniziale di Scrum può essere complessa, richiedendo un cambiamento culturale e organizzativo significativo all'interno dell'azienda.

  La transizione richiede tempo e impegno per formare il team e comprendere appieno i principi di Scrum.

- **Coinvolgimento completo del team:** Scrum richiede il coinvolgimento attivo e continuo di tutti i membri del team.

  L'assenza o il coinvolgimento non completo di anche un singolo membro potrebbe compromettere l'efficacia complessiva.

- **Eccesso di cerimonie:** L'abbondanza di cerimonie Scrum (Sprint Planning, Daily Scrum, Sprint Review, Sprint Retrospective) potrebbe diventare eccessiva e consumare tempo, riducendo il tempo effettivo per lo sviluppo.

- **Difficoltà nella stima precisa:** La stima delle attività potrebbe essere difficile.

  Il team potrebbe avere difficoltà a stimare con precisione la durata delle attività, portando a previsioni imprecise sul completamento delle funzionalità.

- **Adattabilità a grandi progetti o team distribuiti:** In progetti molto grandi o con team distribuiti su diverse posizioni geografiche, Scrum potrebbe incontrare difficoltà nell'essere gestito in modo efficace, poiché la comunicazione e il coordinamento possono diventare più complessi.

- **Mancanza di struttura predittiva:** Scrum è orientato più verso l'adattabilità e la flessibilità rispetto alla pianificazione dettagliata.

  Questo potrebbe rendere difficile la previsione precisa dei tempi e dei risultati.

Capitolo 4

Confronto tra Waterfall e Agile

Il confronto tra il modello Waterfall e le metodologie Agili è fondamentale per comprendere le diverse modalità di gestione dei progetti e dello sviluppo software.

Questi approcci presentano differenze sostanziali, influenzando notevolmente la pianificazione, l'esecuzione e il risultato finale dei progetti.

4.1 Quando utilizzare Waterfall

Il modello Waterfall è un approccio sequenziale nel project management dove le fasi del progetto procedono in modo lineare, passando da una fase all'altra in una sequenza rigorosa. Questo modello prevede fasi distinte, tra cui analisi, progettazione, sviluppo, test e distribuzione, permettendo inoltre di pianificare tutte le attività di progetto prima di procedere con l'esecuzione.

L'applicazione del Project Management a modello Waterfall è consigliata nei casi in cui sussistano le condizioni descritte nelle seguenti sezioni.

4.1.1 Requisiti stabili e ben definiti

Il Waterfall è indicato quando i requisiti del progetto sono stabili e ben definiti sin dall'inizio. È ottimale per progetti con requisiti rigidi, scadenze predeterminate e una struttura gerarchica chiara. Questo approccio permette una pianificazione dettagliata e una stima accurata delle risorse necessarie per completare il progetto. Esempi tipici includono:

- **Progetti governativi**: dove le specifiche sono spesso determinate da regolamenti e leggi.
- **Sistemi critici per la sicurezza**: come progetti di costruzione immobiliare o di impiantistica, dove i requisiti devono essere chiari e rigorosamente rispettati.

- **Infrastrutture fisiche**: come reti elettriche o oleodotti.
- **Produzione industriale**: dove i processi di produzione devono essere seguiti in una sequenza specifica.

4.1.2 Limiti del modello Waterfall

Risulta essere limitante come approccio in contesti in cui i requisiti non sono chiari o soggetti a cambiamenti durante lo sviluppo. La sua struttura rigida, basata su una pianificazione dettagliata iniziale, rende difficile apportare modifiche durante lo svolgimento del progetto, in quanto non prevede deviazioni dal percorso prestabilito. Alcuni svantaggi includono:

- **Feedback tardivo**: i feedback degli utenti finali arrivano solo dopo il completamento di una fase, il che può portare a ritardi nella correzione e risoluzione dei problemi.
- **Gestione dei rischi non individuati tempestivamente**: Sebbene negli approcci waterfall sia fondamentale svolgere un'approfondita analisi dei rischi nelle fasi iniziali del progetto, può risultare più complesso gestire l'emergere di rischi non individuati per tempo, ossia prima dell'inizio dell'esecuzione.

Il modello Waterfall è stato tradizionalmente utilizzato in settori in cui i requisiti possono essere definiti in modo esaustivo sin dall'inizio. Tuttavia, le limitazioni del Waterfall in contesti di incertezza e cambiamenti imprevisti dei requisiti hanno portato all'adozione di approcci più flessibili.

4.2 Quando utilizzare Agile

Le metodologie Agili, come Scrum, si rivelano efficaci in progetti dove i requisiti sono suscettibili di frequenti cambiamenti. L'approccio Agile favorisce una gestione iterativa e flessibile, concentrandosi sulla collaborazione, sulla rapida risposta ai cambiamenti e sulla consegna incrementale di valore.

L'applicazione del Project Management ai modelli Agile è consigliata nei casi in cui sussistano le condizioni descritte nelle seguenti sezioni.

4.2.1 Progetti con requisiti in evoluzione

Gli approcci Agile sono vantaggiosi nei progetti in cui i requisiti sono suscettibili di modifiche o evolvono nel tempo. Ad esempio, nello sviluppo di software per applicazioni web o mobile, l'approccio Agile può essere preferibile poiché consente di adattarsi rapidamente a nuove esigenze o requisiti in evoluzione. In questi contesti, Agile permette:

- **Flessibilità**: La capacità di rispondere rapidamente ai cambiamenti nei requisiti e alle nuove informazioni.
- **Iterazioni brevi**: Sviluppo suddiviso in brevi cicli che permettono di rilasciare funzionalità incrementali.
- **Prioritizzazione continua**: La possibilità di rivedere e riordinare le priorità delle funzionalità ad ogni iterazione.

4.2.2 Coinvolgimento del cliente elevato

La metodologia Agile è particolarmente utile quando il coinvolgimento del cliente è elevato. Richiede una continua iterazione tra sviluppatori e stakeholder per garantire il raggiungimento degli obiettivi e una stretta collaborazione per rispondere prontamente ai feedback e ai cambiamenti. I vantaggi includono:

- **Feedback frequente**: Il cliente può fornire feedback continuo su ogni iterazione, garantendo che il prodotto finale soddisfi le sue aspettative.
- **Collaborazione stretta**: Sviluppatori e stakeholder lavorano insieme per risolvere problemi e migliorare il prodotto.
- **Adattamento rapido**: La capacità di apportare modifiche in base ai feedback ricevuti, migliorando continuamente i risultati del progetto.

4.2.3 Necessità di risultati tempestivi e adattabilità

In situazioni in cui è fondamentale ottenere risultati tempestivi e necessario adattarsi rapidamente a nuove richieste o requisiti, l'approccio Agile si dimostra vantaggioso. Consentendo una consegna incrementale e rapida di funzionalità, Agile permette di ottenere feedback tempestivi e migliorare continuamente il prodotto. I benefici principali sono:

- **Rilascio rapido**: Funzionalità vengono rilasciate frequentemente, permettendo un miglioramento continuo.
- **Risoluzione di problemi**: Problemi e bug vengono identificati e risolti più velocemente grazie ai cicli brevi.
- **Adattabilità**: L'approccio Agile permette di rispondere rapidamente a cambiamenti nel mercato o nei requisiti del cliente.

4.2.4 Settori di applicazione tipici

L'approccio Agile è stato ampiamente adottato per affrontare la complessità e la rapidità dei cambiamenti nei contesti moderni predisposti a mutamenti. La sua flessibilità e adattabilità lo rendono una scelta preferita in ambiti dove la risposta rapida ai cambiamenti e la consegna tempestiva di valore sono essenziali per il successo del progetto. Esempi di applicazioni includono:

- **Sviluppo software**: Ideale per progetti di sviluppo software dove i requisiti possono cambiare frequentemente.
- **Marketing**: Utilizzato in campagne di marketing per adattarsi rapidamente alle tendenze del mercato.
- **Educazione**: Implementato in ambienti educativi per adattare rapidamente i curriculum alle esigenze degli studenti.

4.2.5 Benefici dell'Agile

Adottare metodologie Agili porta diversi benefici tra cui:

- **Maggiore soddisfazione del cliente**: Grazie al feedback continuo e alle consegne incrementali, i clienti ricevono un prodotto che soddisfa meglio le loro esigenze.
- **Migliore gestione del rischio**: Problemi e rischi vengono identificati e affrontati più rapidamente grazie ai cicli iterativi.
- **Migliore qualità del prodotto**: Il focus sulla revisione continua e il feedback costante migliorano la qualità complessiva del prodotto finale.

L'approccio Agile è stato ampiamente adottato per affrontare la complessità e la rapidità dei cambiamenti nei contesti moderni. La sua flessibilità e adattabilità lo rendono una scelta preferita in ambiti dove la risposta rapida ai cambiamenti e la consegna tempestiva di valore sono essenziali per il successo del progetto.

4.3 Quando preferire il framework Scrum

Scrum, come framework Agile, offre una struttura ben definita con ruoli chiari, cerimonie regolari e artefatti specifici. È particolarmente indicato quando si desidera un approccio strutturato all'interno del paradigma Agile, con un forte focus sull'incremento di prodotto in intervalli di tempo definiti noti come Sprint. [4]

4.3.1 Struttura di Scrum

Scrum è caratterizzato da una serie di elementi distintivi che ne definiscono la struttura:

- **Ruoli definiti:** In Scrum, esistono ruoli chiari come il Product Owner, che rappresenta gli interessi degli stakeholder; lo Scrum Master, che facilita l'applicazione dei processi Scrum e il Team di Sviluppo, responsabile della creazione del prodotto.
- **Cerimonie regolari:** Scrum prevede cerimonie regolari, come lo Sprint Planning (pianificazione dello sprint), le Daily Stand-Up (riunioni giornaliere), lo Sprint Review (revisione dello sprint) e lo Sprint Retrospective (retrospettiva dello sprint).
- **Artefatti specifici:** Gli artefatti di Scrum includono il Product Backlog (elenco delle funzionalità richieste), lo Sprint Backlog (elenco delle funzionalità dello sprint) e l'Incremento (il risultato del lavoro svolto durante lo sprint).

4.3.2 Applicazioni di Scrum

Scrum è ampiamente adottato in diversi settori, ma è particolarmente efficace in contesti specifici:

- **Aziende di sviluppo software:** Scrum è molto popolare nelle aziende di sviluppo software per garantire una consegna regolare e iterativa del prodotto. Offre una struttura chiara con ruoli definiti come Product Owner, Scrum Master e Team di Sviluppo, consentendo una chiara divisione dei compiti e un'organizzazione efficiente del lavoro.
- **Consegna regolare e strutturata:** È vantaggioso quando si desidera una consegna regolare e strutturata del prodotto, con intervalli di tempo definiti. Questo approccio permette di suddividere il lavoro in iterazioni gestibili, consentendo un controllo più preciso sul processo di sviluppo.
- **Gestione di progetti complessi:** Scrum è efficace nella gestione di progetti complessi dove è necessario mantenere una chiara visione del progresso e adattarsi rapidamente ai cambiamenti.
- **Industria finanziaria:** Utilizzato per gestire progetti dove la conformità e la regolamentazione richiedono una pianificazione rigorosa e una consegna periodica di aggiornamenti.

4.3.3 Vantaggi di Scrum

Scrum offre numerosi vantaggi grazie alla sua struttura e alle sue pratiche:

- **Chiarezza nei ruoli e responsabilità:** I ruoli ben definiti aiutano a evitare confusione e sovrapposizione dei compiti, migliorando l'efficienza del team.
- **Ritmo sostenibile:** Iterazioni brevi e definite permettono ai team di mantenere un ritmo costante, evitando sovraccarichi e burnout. Un efficace controllo del debito tecnico è essenziale, poiché se trascurato, aumenta la complessità e i costi delle future implementazioni rallentando inevitabilmente i tempi di sviluppo.
- **Adattabilità e miglioramento continuo:** Le cerimonie come le retrospettive permettono ai team di riflettere e migliorare continuamente i loro processi e metodi di lavoro.
- **Focalizzazione sugli obiettivi:** Gli Sprint ben definiti aiutano a mantenere il focus sugli obiettivi a breve termine, facilitando la consegna di incrementi di valore.

4.3.4 Limitazioni di Scrum

Tuttavia, Scrum potrebbe presentare alcune limitazioni:

- **Struttura rigida:** In ambienti dove è necessaria una maggiore flessibilità e adattabilità senza dover seguire una struttura rigida, altre metodologie Agile più flessibili potrebbero essere più adatte rispetto a Scrum. Ad esempio, Kanban offre un flusso di lavoro più continuo senza le restrizioni degli Sprint.
- **Dipendenza dai ruoli:** La mancanza di un ruolo chiave come lo Scrum Master o il Product Owner può compromettere l'efficacia del framework.
- **Resistenza al cambiamento:** Team o organizzazioni abituati a metodi tradizionali potrebbero trovare difficile adattarsi alla disciplina richiesta da Scrum.

4.4 Quando preferire il framework Kanban

Kanban è un framework Agile che si concentra sulla visualizzazione del flusso di lavoro e sulla gestione del processo produttivo, senza imporre iterazioni a tempo fisso come gli Sprint di Scrum. È particolarmente indicato per contesti in cui il flusso di lavoro è continuo e la priorità è l'efficienza nel completamento delle attività. [1]

4.4.1 Struttura di Kanban

Kanban si distingue per una serie di elementi fondamentali che ne definiscono il funzionamento:

- **Visualizzazione del flusso di lavoro:** La visualizzazione tramite una Kanban board permette ai team di avere una chiara panoramica dello stato delle attività, con colonne che rappresentano le diverse fasi del processo.
- **Limiti del Work In Progress:** Kanban incoraggia l'imposizione di limiti sul numero di attività che possono essere gestite contemporaneamente.
- **Flusso continuo:** A differenza di Scrum, Kanban non utilizza iterazioni a tempo fisso. Il lavoro viene completato man mano che le attività avanzano nel flusso, con nuove attività che possono essere aggiunte in qualsiasi momento.

4.4.2 Applicazioni di Kanban

Kanban è utilizzato in diversi settori, ma è particolarmente efficace nei seguenti contesti:

- **Team di manutenzione e supporto:** Kanban è ideale per team che gestiscono flussi di lavoro continui e imprevedibili, come quelli di supporto tecnico o manutenzione, dove è necessario rispondere rapidamente a richieste variabili.
- **Ambienti con cambiamenti frequenti:** In situazioni dove le priorità cambiano frequentemente, Kanban permette una maggiore flessibilità rispetto agli Sprint di Scrum, adattandosi ai cambiamenti in tempo reale.
- **Gestione di flussi di lavoro complessi:** Kanban aiuta a gestire processi complessi e articolati, migliorando la visibilità sullo stato delle attività e ottimizzando il flusso di lavoro complessivo.

4.4.3 Vantaggi di Kanban

Kanban offre diversi vantaggi grazie alla sua flessibilità e all'attenzione sulla gestione del flusso di lavoro:

- **Flessibilità nella gestione del lavoro:** Kanban consente di aggiungere e rimuovere attività in qualsiasi momento, offrendo una gestione continua e dinamica del flusso di lavoro.
- **Miglioramento continuo:** La possibilità di monitorare il flusso di lavoro in tempo reale consente ai team di identificare colli di bottiglia e ottimizzare continuamente il processo.

- **Riduzione dei sovraccarichi:** I limiti del Work In Progress evitano che il team sia sovraccarico, migliorando la produttività e la qualità del lavoro.
- **Visualizzazione trasparente:** La Kanban board offre una chiara rappresentazione visiva dello stato delle attività, migliorando la trasparenza e la comunicazione all'interno del team.

4.4.4 Limitazioni di Kanban

Nonostante i suoi vantaggi, Kanban presenta alcune limitazioni:

- **Meno struttura rispetto a Scrum:** Anche se Kanban non prevede formalmente ruoli specifici come lo Scrum Master o il Product Owner, è comunque possibile introdurre figure analoghe o organizzazioni alternative. Tuttavia, la gestione dei processi e delle 'iterazioni' in Kanban tende ad essere meno strutturata e potrebbe risultare più caotica rispetto a Scrum.
- **Dipendenza dall'autogestione:** Kanban richiede che il team sia altamente autogestito, con una forte capacità di monitorare e migliorare continuamente il proprio processo.
- **Meno focalizzazione su rilasci a breve termine:** A differenza degli Sprint di Scrum, Kanban non impone obiettivi temporali su rilasci a breve termine, il che potrebbe ridurre la pressione positiva per una consegna rapida del valore.

Capitolo 5

Conclusioni

Il confronto tra il modello Waterfall e le metodologie Agile è cruciale per comprendere le diverse modalità di gestione dei progetti di sviluppo software. Ciascun approccio presenta vantaggi e svantaggi specifici, rendendo la scelta del modello di gestione un fattore determinante per il successo del progetto.

La scelta tra il modello Waterfall e Agile non è universale e deve essere basata sulle caratteristiche specifiche del progetto. Il modello Waterfall, caratterizzato da una struttura sequenziale e una pianificazione dettagliata svolta prima dell'avvio dell'esecuzione, risulta ideale per progetti con requisiti stabili e una bassa probabilità di cambiamenti.

D'altro canto, le metodologie Agile offrono un approccio flessibile e iterativo, adatto a progetti dove i requisiti possono evolvere nel tempo. La capacità di rispondere rapidamente ai cambiamenti e di coinvolgere continuamente gli stakeholder rende Agile particolarmente vantaggioso in ambienti dinamici come lo sviluppo di applicazioni web e mobile.

Le metodologie Agile stanno guadagnando sempre più popolarità, soprattutto in settori dove la velocità e l'adattabilità sono essenziali. Tuttavia, nonostante la crescente adozione di Agile, il modello Waterfall continua a essere rilevante in numerosi contesti industriali dove la prevedibilità è prioritaria.

Per i project manager e i team di sviluppo, è essenziale valutare attentamente le caratteristiche del progetto e le esigenze degli stakeholder prima di scegliere un approccio di gestione. In alcuni casi, si potrebbe valutare l'adozione di un approccio ibrido che combina elementi Waterfall e Agile, che potrebbe offrire una soluzione bilanciata, sfruttando i punti di forza di entrambi gli approcci.

In conclusione, la comprensione approfondita delle metodologie Waterfall e Agile e delle loro applicazioni è fondamentale per la gestione efficace dei progetti software.

L'adozione dell'approccio più adatto, basata su un'analisi accurata del contesto e dei requisiti del progetto, può migliorare significativamente la probabilità di successo del progetto e la soddisfazione degli stakeholder.

Capitolo 6

Ringraziamenti

Siamo finalmente giunti al termine di questo lungo percorso che oggi siamo a celebrare.

In tutti questi anni, molte volte mi sono chiesto se fosse davvero la scelta giusta, se fosse questa la mia strada e se questo giorno sarebbe mai arrivato.

Oggi che lo sto vivendo, voglio fare i miei più sinceri ringraziamenti a:

- A me stesso, per aver sempre provato ancora una volta dopo ogni fallimento. Se non fosse stato per la mia determinazione, oggi non sarei qui a concludere questa lunga avventura.
- Laura, la mia metà nonchè futura moglie tra 16 giorni, valvola di sfogo durante i tanti momenti in cui pensavo di ritirarmi. Oggi festeggio questa Laurea con te, ma anche grazie a te, che non hai mai assecondato i miei momenti di crisi.

  Ti aspetto all'altare
- Mia madre Lina, che da buona madre del sud mi ha sempre spronato ad andare avanti anche dopo i tanti fallimenti e i periodi difficili.

  Mamma siamo arrivati dove mai avremmo pensato.
- Mio padre Filippo, che mi ha dato il tempo e il sostegno per proseguire gli studi in tutti questi anni.

  Papà è finita, sono ufficialmente Ingegnere.
- Mia sorella Ilenia, spalla certa e forte, che si è sempre offerta di aiutarmi e supportarmi in ogni singolo esame, non ha mai fatto mancare la frase "Com'è andata?".
- Matteo, collega e amico, grande sostenitore e fonte di sprone nel raggiungimento di questo obiettivo. Oggi festeggiamo me, ma non vedo l'ora di festeggiare te.

- Al mio Relatore, grazie per questa tesi che mi ha permesso di confrontarmi con la materia in modo viscerale, scoprendone le sfaccettature che tanto mi hanno appassionato.
- I miei datori di lavoro, prima Fastcode e poi Antreem, che sono stati i finanziatori di questo percorso. Senza di voi, non avrei mai raggiunto questo importante traguardo, mi avete fornito la tranquillità economica necessaria.
- A tutti coloro con cui ho condiviso anche solo un esame, grazie.

Ogni tassello posato ha contribuito a costruire questo grandissimo risultato che è sicuramente personale, ma senza tutti voi non si sarebbe mai realizzato,

Grazie di cuore.

Bibliografia

[1] Henrik Kniberg. Kanban and scrum—making the most of both. *C4 Media*, 2010.

[2] Agile Manifesto. *http://agilemanifesto.org/*. 2024-06-05.

[3] Project Management Institute. *A Guide to the Project Management Body of Knowledge (PMBOK Guide)*. Project Management Institute, Newtown Square, PA, 2021.

[4] Kenneth S. Rubin. *Essential Scrum: A Practical Guide to the Most Popular Agile Process*. Addison-Wesley, Upper Saddle River, NJ, 2012.