Modern Project Management

PAPER

Rev. 1.1.0 – 12 Marzo 2023

ModernPM

an AgileConstellation Star

Sommario

1. Overview 4

2. Il Modern PM 5

2.1 Mindset 6

2.2 Governance 7

2.3 Guided Continuous Improvement 8

2.4 Leadership 10

3. Ruoli 13

3.1 Il Project Manager 14

3.2 Il Product Owner 15

3.3 Lo Scrum Master 16

3.4 Il Team Leader 18

4. ModernPM Lifecycle Toolkit 19

4.1 Lifecycle Predittivo 21

4.2 Lifecycle Iterativo e Incrementale 24

4.3 Lifecycle Agile 25

4.4 Lifecycle Lean 29

4.5 Lifecycle Exploratory 32

4.6 Lifecycle Program 34

4.7 Lifecycle Hybrid 36

5 ModernPM Fast Prototyping 37

5.1 Vision 38

5.2 Definizione dello Scope 39

5.2.1 Work Breakdown Structure 39

5.2.2 Story Mapping 40

5.3 Complexity 41

5.3.1 Scope Definition Accuracy 43

5.3.2 Uncertainty 44

5.3.3 ModernPM Decision Tree 47

5.4 Lifecyles Bouquet 48

6 Richiami 49

6.1 PMI Disciplined Agile 49

6.2 PMI PMBOK 52

7 Conclusione 54

1. Overview

Questa guida ha l'obiettivo di fornire un **approccio operativo** alla gestione ed esecuzione dei progetti, definito di seguito come **Modern Project Management** (**ModernPM**, **MPM**).

Basandosi su soluzioni consolidate, in particolare *sul PMBOK®* e sul toolkit *PMI Disciplined Agile® (DA™),* viene descritta una strategia concreta per analizzare il contesto specifico di ogni progetto e identificare il mix di lifecycle più idonei alla sua gestione ed esecuzione.

I diversi temi verranno trattati ad un livello medio-alto, lasciano al lettore lo studio e la ricerca degli approfondimenti che eventualmente ritiene necessario. Questo approccio è stato scelto per evitare di trasformare questa guida in un "tomo", evitando di riportare elementi facilmente trovabili in letteratura, ma focalizzandosi nel mettere insieme uno strumento flessibile, e di immediato utilizzo, che metta il lettore nelle condizioni di essere subito operativo.

2. Il Modern PM

La parola chiave della natura iterativa del **ModernPM** è **Adapt**: dopo aver creato la visione di progetto, e identificato il contesto relativo, si procede alla fase di definizione dei deliverable (obiettivi) e alla scelta dei lifecycle più idonei per generare rapidamente valore per il cliente e raccoglierne i feedback.

In tal modo è possibile riallineare le attività in funzione del reale valore generato per gli stakeholder, riducendo il rischio di dover fare interventi massivi nelle fasi finali del progetto dove i costi potrebbero essere non più sostenibili.

Come visibile nel *Poster* riportato di seguito, il ModernPM si basa su **4 Pillars** che ne esplicitano gli elementi fondanti:

- **Mindset**, l'aspetto culturale da sviluppare;
- **Governance**, come gestire operativamente un progetto e quali competenze sono necessarie;
- **Guided Continuous Improvement**, la capacità di migliorarsi continuamente in modo pragmatico;
- **Leadership**, la necessità di stimolare la nascita di leader naturali nel proprio contesto.

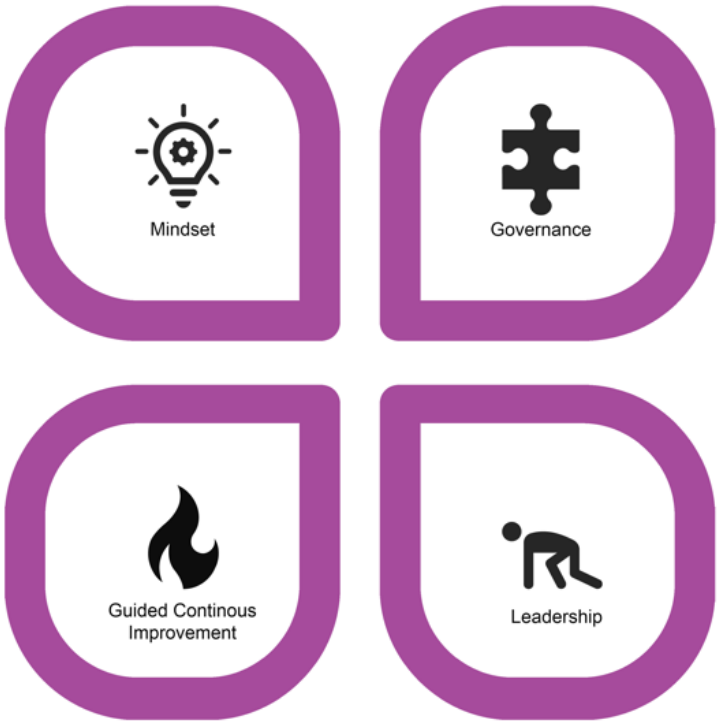

Figura 1 - ModernPM Poster

2.1 Mindset

Alla base del ModernPM troviamo la *Filosofia, i Principi e le Pratiche* dell'**AgileConstellation Manifesto**[1], di cui il ModernPM è una *Star*, definendo specifiche pratiche e principi attinenti al dominio di riferimento.

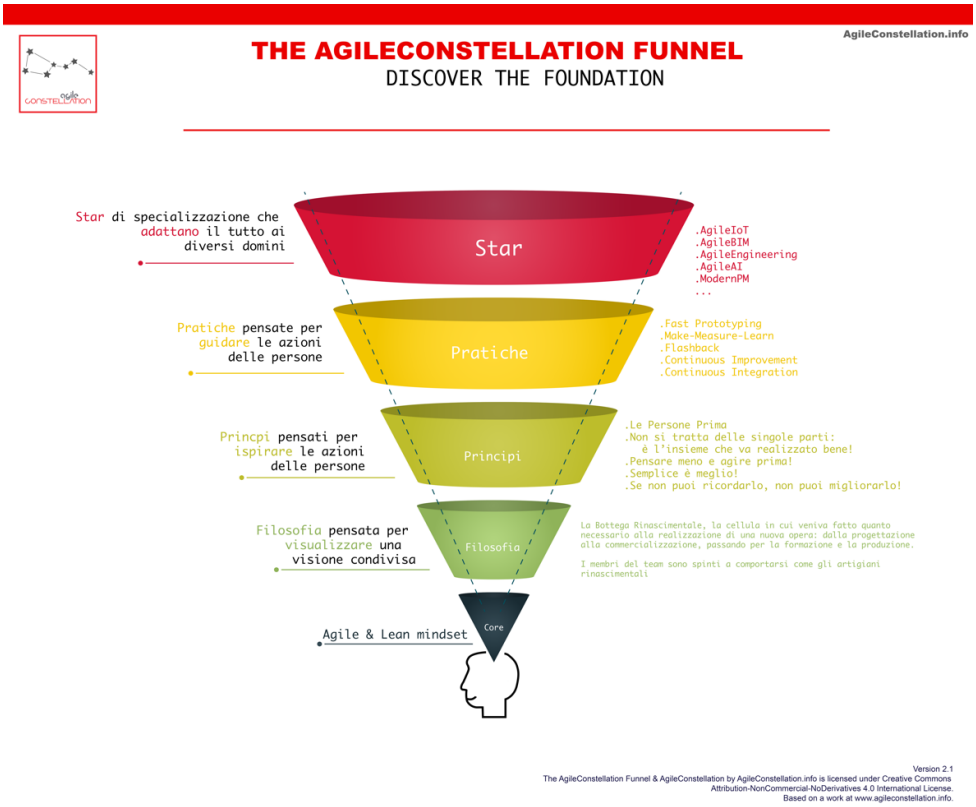

Figura 2 - AgileConstellation funnel

Si ha, quindi:

- **Filosofia**, ispirata alla **Bottega Rinascimentale**, ovvero la cellula che assolve a quanto necessario per la realizzazione di una nuova opera: dalla progettazione, alla realizzazione e alla commercializzazione.
- **Principi (core):**
  - *Non si tratta delle singole parti: è l'insieme che va realizzato bene!*
  - *Pensare meno e agire prima!*
  - *Semplice è meglio!*
  - *Se non puoi ricordarlo, non puoi migliorarlo!*
- **Pratiche (core):**
  - *Fast Prototyping*, validare la sostenibilità della soluzione

[1] www.agileconstellation.info

- *Make-Measure-Learn*, sperimentare rapidamente le diverse ipotesi e le diverse assunzioni
- *Flashback*, allineamento rapido in cui è l'osservatore ad andare al desk di lavoro
- *Continuous Improvement*, migliorare costantemente ogni aspetto

Continuous Integration, integrare costantemente le differenti anime della soluzione

2.2 Governance

Uno degli errori comuni è quello di confondere *Progetto* e *Prodotto*, utilizzando i due termini persino come sinonimi. Esistono invece profonde differenze tra di essi:

- Un **Progetto** ha un inizio e una fine ben definiti, rispondendo, inoltre, alla caratteristica di *univocità*, in quanto non è routinario ma composto da un insieme specifico di attività progettate per raggiungere un obiettivo unico, spesso attraverso persone che non lavorano stabilmente insieme. Alla fine di un progetto, il team (o i team) viene solitamente sciolto e i membri assegnati a nuovi progetti.
- Un **Prodotto** è un servizio, una piattaforma, un'applicazione, un sistema, che viene creato, mantenuto e supportato per fornire vantaggi a esigenze specifiche del cliente e dell'azienda. I prodotti tendono ad essere mantenuti da un team stabile (o anche da più team stabili) che lavora in continuità.

Questi specifici perimetri si riflettono in modo diretto su come i progetti ed i prodotti vengono *eseguiti* e *gestiti*, sugli *strumenti utilizzati* e sulle *figure* (*ruoli*) che li accompagnano.

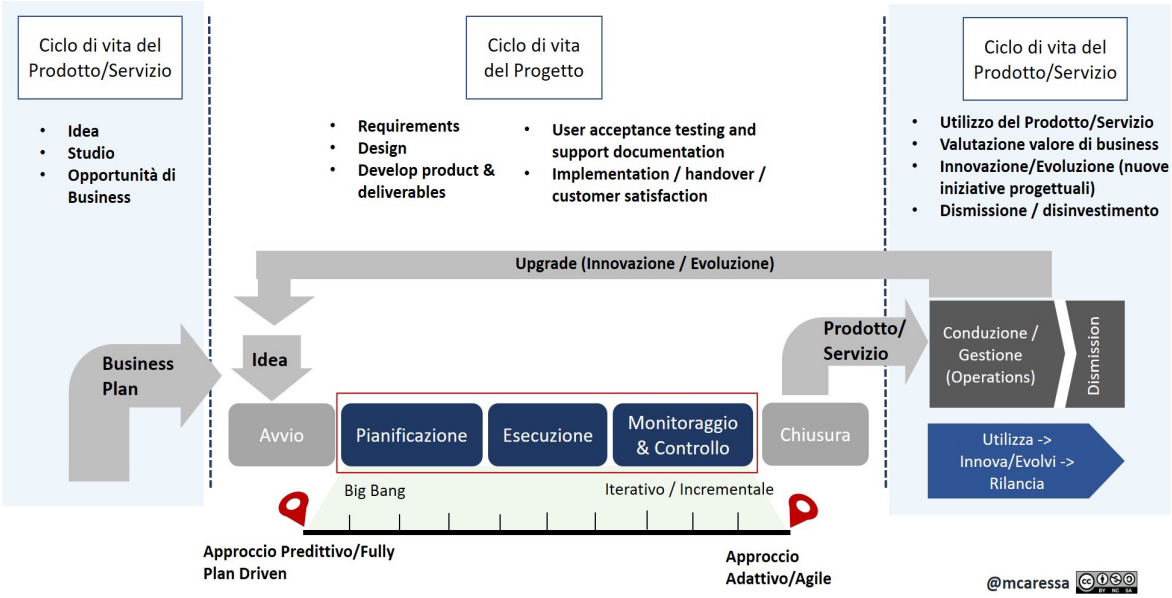

Figura 3 - Project & Product Lifecycle

2.3 Guided Continuous Improvement

La **Guided Continuous Improvement**[2] (**GCI**) va ad estendere il principio base (AgileConstellation) di Continuous Improvement, e si basa su due aspetti fondamentali:

- il **Kaizen loop**, ovvero l'approccio in cui il team sperimenta progressivamente piccoli cambiamenti nel proprio *Way of Working* (WoW). A valle di ogni esperimento, se quanto sperimentato ha dato i frutti sperati, il team lo adotta in modo strutturato, viceversa sceglierà una opzione diversa che andrà nuovamente a sperimentare.
- la **Continuous Improvement**, ovvero l'atto di applicare una serie di *Kaizen loop* per migliorare il proprio WoW nel tempo.

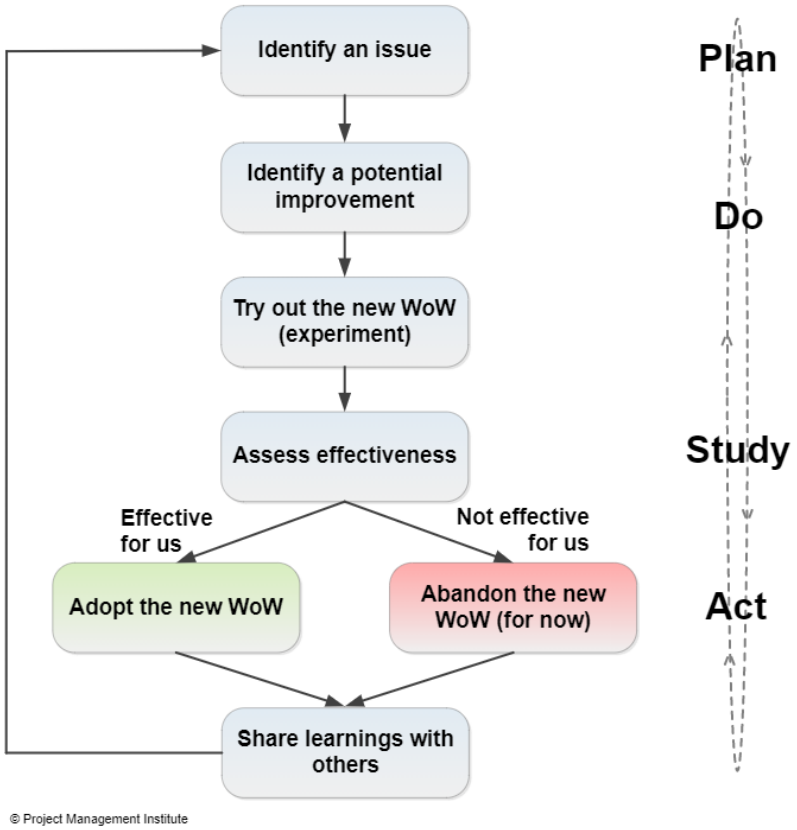

Figura 4 - Kaizen Loop e Continuous Improvement

La Guided Continuous Improvement è una tecnica (suggerita dal toolkit *Disciplined Agile*) che rafforza la Continuous Improvement, suggerendo al team un bouquet di opzioni per affrontare pragmaticamente il problema del "*method prison*"[3]:

[2] https://www.pmi.org/disciplined-agile/gci/guided-continuous-improvement
[3] https://essence.ivarjacobson.com/publications/white-papers/tear-down-method-prisons-set-free-practices

"The only option that development organizations and teams see themselves as having is to adopt a method and to reject all others—whereas, in fact, what is needed for organizations and teams to be free is to select the professional practices that they need, from wherever these may be defined, and use them in whatever permutations and combinations are appropriate to meet the exact set of circumstances and challenges they face".

Gli impatti del restare imbrigliati in un framework (o un metodo) troppo prescrittivo sono stati ulteriormente evidenziati da *Donald J. Reifer*[4] e rappresentati nella figura seguente:

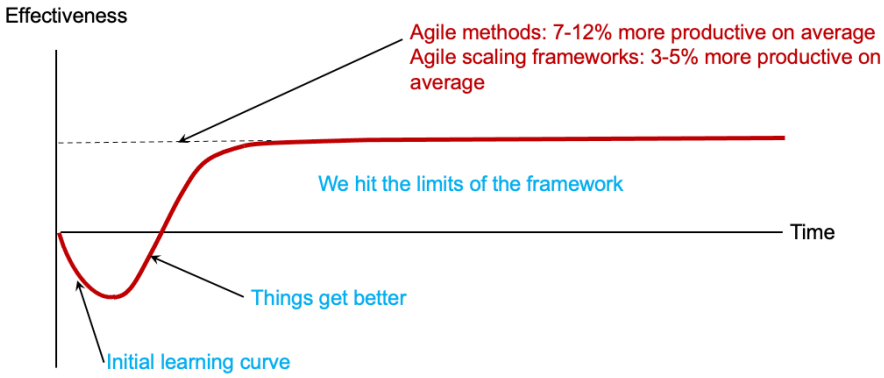

Figura 5 - Source: Reifer, D. Quantitative Analysis of Agile Methods Survey (2017): Twelve Major Findings

Il problema di fondo, con tali approcci, è che troppe volte ci si dimentica che sono *dei modelli*, legati spesso a contesti ben definiti. Se a questo si unisce la pretesa di utilizzarli indiscriminatamente per ogni progetto in modo dogmatico, e non in maniera critica e contestualizzata, ci si ritrova imprigionati in un meccanismo che poco aiuta a sviluppare la capacità di adattarsi in funzione degli obiettivi da raggiungere.

Per *evadere da questa prigione*, la GCI è l'arma vincente, fornendo un supporto diretto nell'identificazione dei potenziali problemi e, soprattutto, *una guida comprovata* per aiutare il team a identificare le tecniche che potrebbero funzionare nello specifico contesto.

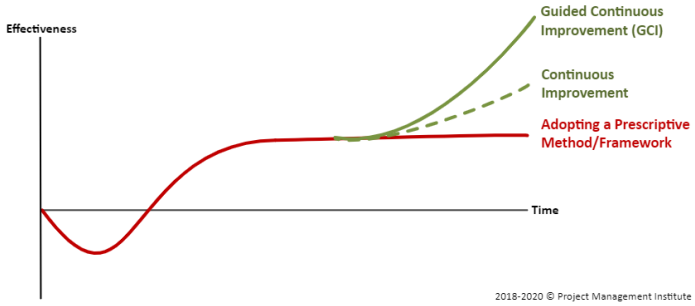

Figura 6 - CGI, CI & Methods/Framework

[4] https://www.infoq.com/articles/reifer-agile-study-2017/

Nonostante ogni opzione vada comunque trattata come un "*esperimento da fare*", il vantaggio è che la CGI consente di aumentare la percentuale di esperimenti riusciti e quindi di aumentare il tasso complessivo di miglioramento.

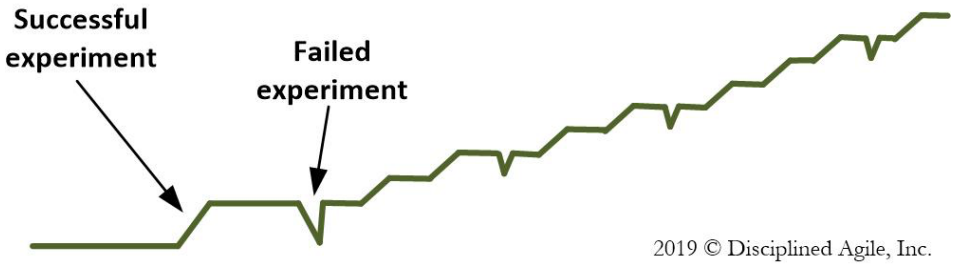

Figura 7 – La GCI aumenta le possibilità di esperimenti di successo

Esistono due indicazioni fondamentali per adottare un approccio basato sulla GCI:

- *Avvalersi di un esperto (PM, Scrum Master, Coach, ecc..) e, soprattutto, ascoltarlo.* Non è facile trovare un vero esperto che sappia aiutare il team e guidarlo nella crescita, ma, una volta trovato, è fondamentale avere il tempo e la pazienza di ascoltarlo e seguirlo.
- *Avvalersi di pratiche consolidate nel cotesto*, in modo da fornire una serie di opzioni ragionate per supportare pragmaticamente il team. I buoni coach hanno l'umiltà di riconoscere che non sanno tutto e, a loro volta, sfrutteranno un mix di strumenti e soluzioni per aiutare il team a prendere decisioni migliori sul come affrontare le sfide, migliorandosi continuamente in chiave sperimentale.

2.4 Leadership

Identificare l'appropriato *stile di leadership* è fondamentale per il successo del progetto.
Difficile però che un solo stile sia sufficiente, più realisticamente, sarà necessario adattarsi costantemente all'evoluzione delle attività e del contesto di sviluppo annesso.

Un aiuto in tale direzione viene dalla *Teoria Situazionale*[5], ovvero dagli studi e dall'esperienza di *Paul H. Hersey, Kenneth Blanchard e Dewey E Johnson, che descrivono la* ***Leaderhip Situazionale*** *nel libro Management of Organizational Behavior: Leading Human Resources*:

> *"non esiste un modo "giusto" e unico di essere leader, ma è necessario scegliere lo stile più appropriato alle diverse situazioni."*

La *Situational Leadership* si basata sulla relazione tra *leader*, *team e stakeholder*, fornendo un quadro per analizzare le diverse situazioni in base al livello di maturità del team e a quella degli stakeholder.

[5] https://it.brainmain.net/10493030-the-situational-theory-of-leadership

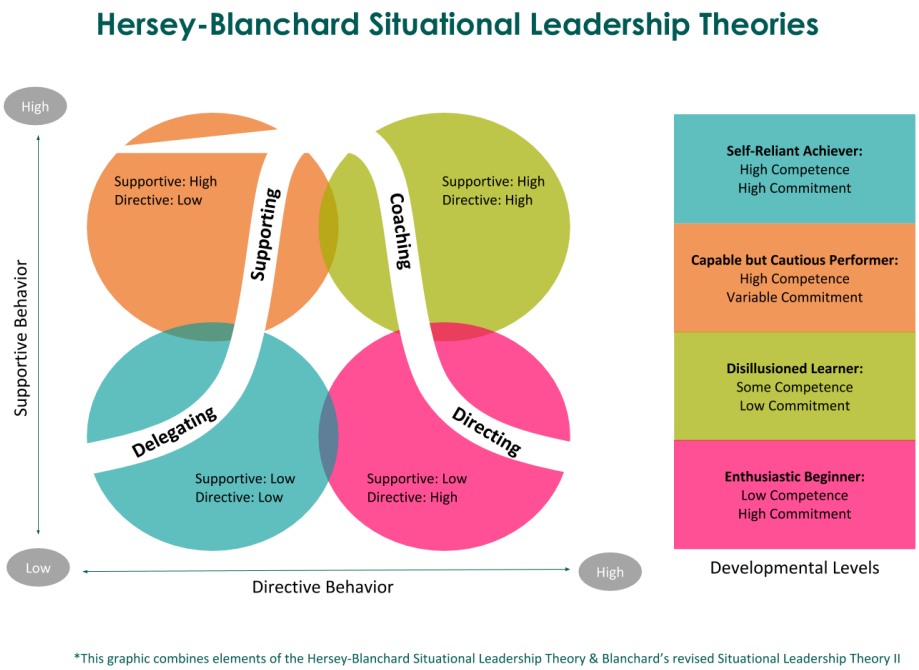

Figura 8 - Situational Leadership (source: https://guntergroup.com/situational-leadership/)

Identificare lo stile di leadership più adeguato al *livello di maturità* aiuta ad avere meno conflitti e una più ampia spinta motivazionale, generando a una maggiore coesione del gruppo di lavoro. Di fatto, al variare dei compiti e/o dei progetti, variano le competenze necessarie che devono essere acquisite dai membri del team e, pertanto, deve variare lo stile di leadership necessario per raggiungere gli obiettivi attraverso lo sviluppo e la crescita delle proprie persone.

Sempre secondo Blanchard (si veda *Leadership and the One Minute Manager* e il modello *SLII®*), tre sono gli elementi a supporto di un leader efficace:

- **Definizione degli Obiettivi**, il punto di partenza di un leader situazionale, creando allineamento su cosa deve essere fatto e quando e quindi motivando i partecipanti ad una iniziativa.
- **Livello di Maturità dei membri del team**, il leader co-valuta il livello di maturità (sviluppo) di ogni membro del team in base alla sua competenza e all'impegno su un obiettivo o un compito specifico. Il livello di maturità è sempre in relazione all'obiettivo o al compito, e non alla persona.
    - Ci sono *quattro livelli di sviluppo* caratterizzati da diverse combinazioni di competenza e impegno del collaboratore: *D1 (Bassa Competenza e Alto Impegno), D2 (Qualche Competenza e Basso Impegno), D3 (Moderata-Alta Competenza e Impegno Variabile) e D4 (Alta Competenza e Alto Impegno).*
- **Corrispondenza tra stile di leadership e sviluppo dei membri del team**, fornisce lo stile di leadership più appropriato per guidare e supportare il collaboratore in base al suo

livello di sviluppo. Con questa abilità i leader apprendono i diversi stili di leadership e come essere più flessibili.

- Così come ci sono quattro livelli di sviluppo, ci sono anche *quattro stili di leadership* caratterizzati da diverse combinazioni di comportamento direttivo e comportamento di supporto: *S1 (Alte Direttive e Basso Supporto), S2 (Alte Direttive e Alto Supporto), S3 (Basse Direttive e Alto Supporto), S4 (Basse Direttive e Basso supporto)*.

La fase della *corrispondenza* è il momento determinante, avendo come obiettivo quello di abbinare lo stile di leadership con il livello di maturità per individuare la corrispondenza più adeguata al contesto: più gli stili di leadership saranno allineati ai livelli di maturità, più rapidamente le persone miglioreranno ed evolveranno da un livello all'altro.

Nonostante sia difficile identificare tutti i possibili stili di leadership, la matrice seguente prova a rappresentarne i più comuni in relazione alla complessità delle attività e all'esperienza del gruppo di lavoro.

Figura 9 - Tipi di Leadership situazionali

3. Ruoli

Nell'ambito di un progetto possono essere presenti molti ruoli con responsabilità e obiettivi specifici, ma anche con sovrapposizioni che possono rendere difficile individuare con chiarezza il relativo perimento di operatività.

Tra di essi è possibile individuare i cosiddetti **leadership roles**:

Ruolo	Descrizione
Project Manager	Supervisiona il progetto e si concentra sugli stakeholder. Gestisce tutto il lavoro inter-funzionale, occupandosi che il progetto si sviluppi nei tempi e con il budget previsti.
Product Owner	Supporta il team da un punto di vista tattico, dando la priorità al Product Backlog. Rappresenta la voce del cliente e si occupa di trovare la giusta sinergia con il cliente, nonché il modo di rappresentare le sue esigenze in elementi lavorabili per il team.
Scrum Master (Agile Master)	Aiuta il team nella quotidianità operativa, in particolare nell'adozione efficace dell'Agile e nello sviluppo del relativo mindset. Si occupa di facilitare gli eventi, stimolare la comunicazione e aiutare i team a superare problematiche e ostacoli, il tutto in ottica di continuo miglioramento.
Team Leader	Coordina il gruppo di cui fa parte nello sviluppo di un progetto e nel raggiungimento degli obiettivi annessi. Non si tratta di un "capo", ma piuttosto di un "compagno" che offre il suo appoggio senza influenzare troppo le decisioni.

Tabella 1: Leadership Roles

Le competenze di tali ruoli possono essere mappate in relazione ai *Performance Domain* del *PMBOK* [6], ovvero sugli aspetti di focalizzazione di un leader di progetto:

- **Team**, affronta le attività e le funzioni associate alle persone responsabili dello sviluppo e dei risultati finali del progetto.
- **Development Approach and Lifecycle**, affronta le attività e le funzioni associate alla modalità di sviluppo, alla cadenza e alle fasi del ciclo di vita del progetto.
- **Measurement**, affronta le attività e le funzioni associate alla valutazione delle prestazioni del progetto e all'adozione di azioni appropriate per mantenere prestazioni ragionevoli.
- **Stakeholders**, affronta le attività e le funzioni associate alla gestione degli stakeholder.

[6] *"A Project Performance Domain is defined as a group of related activities that are critical for the effective delivery of project outcomes."*

- **Project Work**, affronta le attività e le funzioni associate alla definizione dei processi di progetto, alla gestione delle risorse fisiche e alla promozione di un ambiente che favorisce l'apprendimento.
- **Planning**, affronta le attività e le funzioni associate agli aspetti di pianificazione iniziale, di continua revisione ed evoluzione, al coordinamento necessario per fornire i risultati attesi dal progetto.
- **Uncertainty**, affronta le attività e le funzioni associate al rischio e all'incertezza.
- **Delivery**, affronta le attività e le funzioni associate alla consegna delle funzionalità attese e ai relativi aspetti di qualità.

I diversi ruoli possono coprire tutto, o solo parte, lo spettro complessivo dei performance domain, in relazione al loro contributo sul progetto:

	Project Manager	Product Owner	Scrum Master	Team Leader
Team	X	X	X	X
Development Approach and Lifecycle	X		X	X
Measurement	X	X	X	X
Stakeholders	X	X		
Project Work	X		X	
Planning	X	X	X	
Uncertainty	X	X		X
Delivery	X	X		X

Tabella 2 - Performance Domains and Roles Matrix

3.1 Il Project Manager

Il **Project Manager** (**PM**) è accountable del progetto, delle risorse e dell'ambito dei requisiti annessi, e come tale ha la governance complessiva su tutte le attività inerenti, confrontandosi costantemente con gli stakeholder relativi.

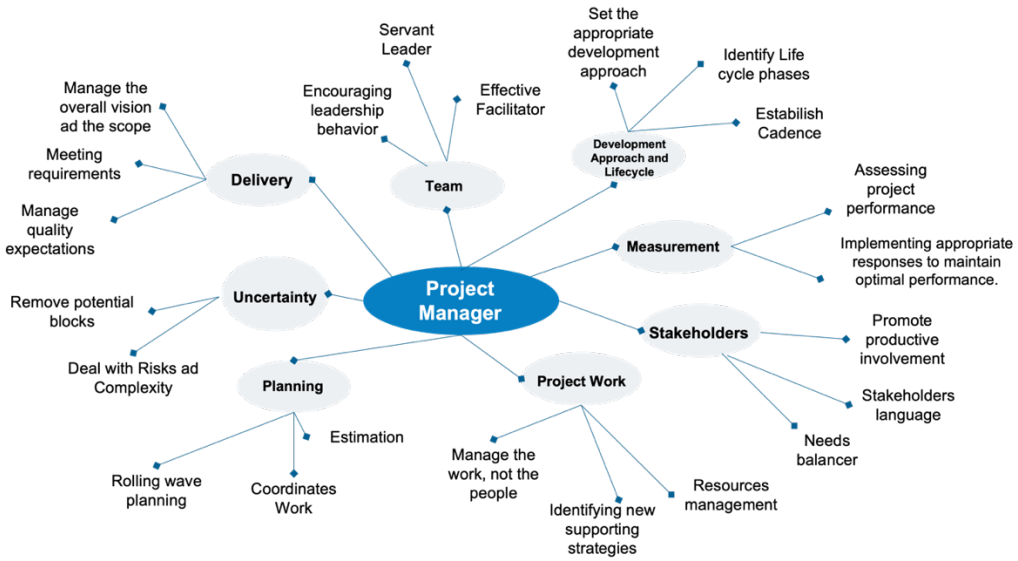

Figura 10 - Project Manager skills

Per svolgere al meglio le proprie attività, il project manager deve possedere un equilibrato mix di *hard* e *soft skills*, dimostrandosi sempre un ottimo comunicatore e un leader empatico. A questo va abbinata una meticolosità organizzativa, una risoluzione nel prendere decisioni, nonché flessibilità e spirito di adattamento.

Non da ultimo, è fondamentale saper affrontare con calma e adeguatezza i diversi problemi che si presentano durante l'intera vita del progetto, portando a boro le persone che possono aiutare a riportare lo stesso in linea con le aspettative.

3.2 Il Product Owner

Il **Product Owner** (**PO**), nell'ambito dei progetti che seguono un lifecycle Agile, è il responsabile del valore di un prodotto e come tale interagisce costantemente con il cliente per validarne lo scope. A differenza del Project Manager, il suo focus è sul prodotto, rispetto al quale declina le proprie attività ed azioni.

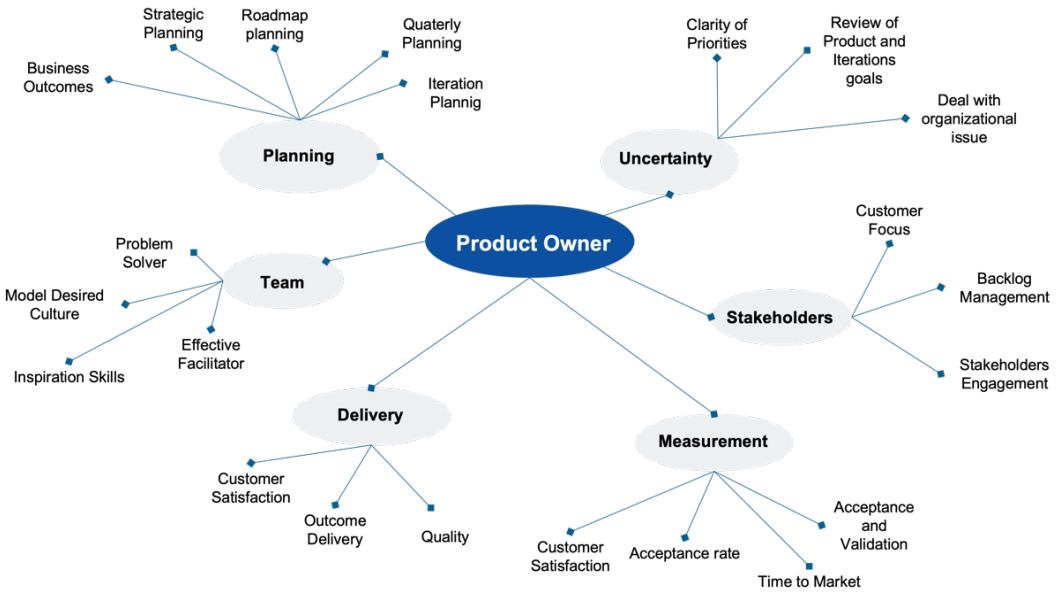

Figura 11 - Product Owner skills

Il PO cura e sviluppa la visione del prodotto e ha l'obiettivo primario di *armonizza le esigenze degli stakeholder in relazione al prodotto*. Per raggiungerle, il PO:

- ha la governance esclusiva del Product Backlog[7];
- si assicura che il team crei Valore, accettando o rigettando quanto realizzato in funzione della "Definition of Done";
- si assicura che il Product Backlog sia visibile a tutti;
- partecipa all'Iteration Planning;
- definisce e verifica gli *Acceptance Criteria.*

3.3 Lo Scrum Master

I compiti dello **Scrum Master** (**SM**), figura presente nell'ambito dei progetti che seguono un lifecycle Agile, sono molto variegati, dipendendo dal contesto e dalla situazione specifica.

Il suo focus è sulla facilitazione del lifecycle che il team sta utilizzando, supportando quest'ultimo grazie alla propria leadership, e alle competenze di coaching, e aiutandolo a migliorare costantemente nel raggiungimento degli obiettivi.

[7] Per una descrizione dei tool specifici annessi ai diversi lifecycle si faccia riferimento a quanto descritto di seguito.

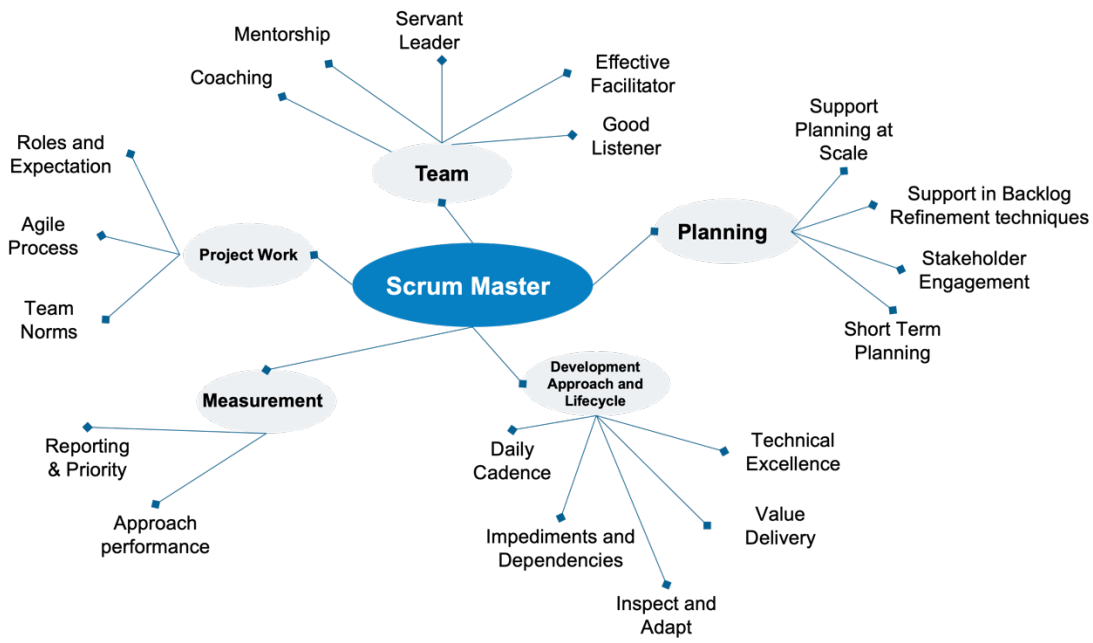

Figura 12 - Scrum Master skills

Lo Scrum Master svolge un ruolo quasi da «consulente, supportando in particolare:

- il **Product Owner**, nell'impostazione degli obiettivi di prodotto, nell'utilizzare il Product Backlog, nell'Iteration Planning e nei vari meeting.
- il **Team**, nell'adozione dello specifico *lifecycle toolkit* (si veda dopo), nel garantire la corretta implementazione dei deliverable e nelle ricerca di azioni per il miglioramento continuo.

Lo SM si concentra sul percorso di sviluppo del team:

- assicurandosi che il team stesso, e l'organizzazione, sviluppi un mindset Agile;
- aiutandolo, insieme all'organizzazione, nell'adizione delle pratiche agili;
- supportandolo nel miglioramento della propria produttività;
- ponendo l'accento sulla qualità di quanto realizzato;
- proteggendolo dalle interferenze esterne;
- rimuovendone gli impedimenti.

Da un punto di vista della crescita, generalmente uno Scrum Master segue il percorso indicato nella figura seguente:

Figura 13 - Percorso di crescita Scrum Master

3.4 Il Team Leader

Il **Team Leader** (**TL**) guida attivatamene lo specifico team di lavoro in relazione agli obiettivi che lo stesso ha in carico.

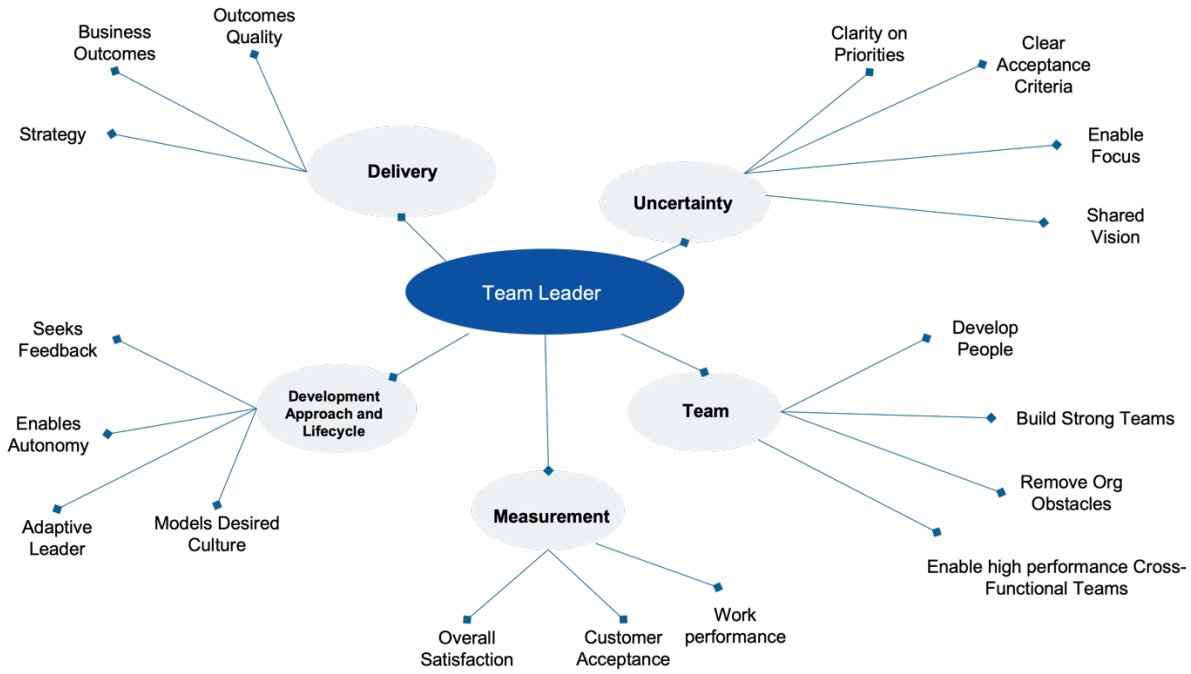

Figura 14 - Team Leader skills

La figura del team leader è spesso determinante per ottenere il massimo coinvolgimento e, di riflesso, un'adeguata produttività del team. Il TL supporta il team:

- nel lavoro quotidiano;
- nella risoluzione dei problemi inerenti alle attività specifiche;
- nell'identificazioni di possibili soluzioni;
- nella mitigazione dei conflitti;
- nella crescita in ambito professionale.

4. ModernPM Lifecycle Toolkit

Il modo in cui si decide di gestire ed eseguire un progetto porta ad un bouquet di elementi che può essere anche molto differente da progetto a progetto.

In linea generale è possibile identificare cinque famiglie di lifecycle (o approcci):

Approach	Requirements	Activities	Delivery	Goal
Predictive	Fixed	Performed once for the entire project	Single delivery	Management of cost
Iterative	Dynamic	Repeated until correct	Single delivery	Correctness of solution
Incremental	Dynamic	Performed once for a given increment	Frequent smaller deliveries	Speed
Agile	Dynamic	Repeated until correct	Frequent small deliveries	Customer value via frequent deliveries and feedback
Lean	Dynamic	Performed once	Frequent small deliveries	Customer value via one-at-time delivery

Tabella 3 – Lifecycles, adattato da Project Management Institute, Agile Practice Guide, Project Management Institute

Ciascuno di questi lifecycle utilizza uno specifico approccio alla gestione dei requisiti, alla pianificazione, alle stime ed all'esecuzione dei lavori, così come riassunto nell'immagine seguente:

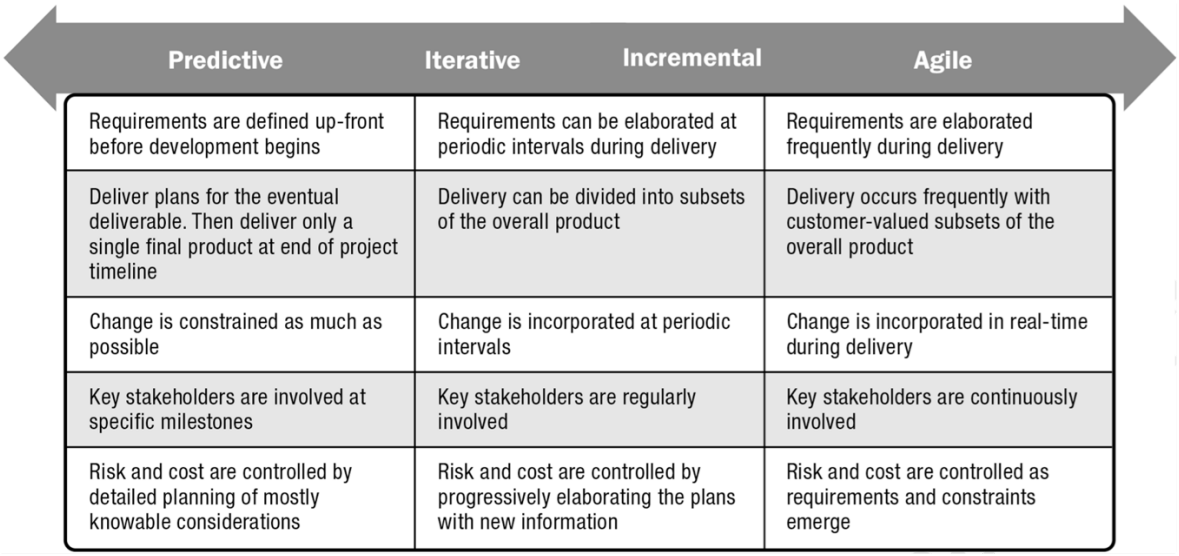

Figura 15 – Lifecycles ranges - A Guide to the Project Management Body of Knowledge (PMBOK® Guide)—Sixth Edition

Nel seguito, i diversi lifecycle vengono caratterizzati in relazione a cinque aspetti fondamentali: **Context, Metrics, Roles, Events** e **Tools** che, nell'insieme, definiscono il **ModernPM Lifecycle Toolkit** (**MLT**) per ognuno di essi.

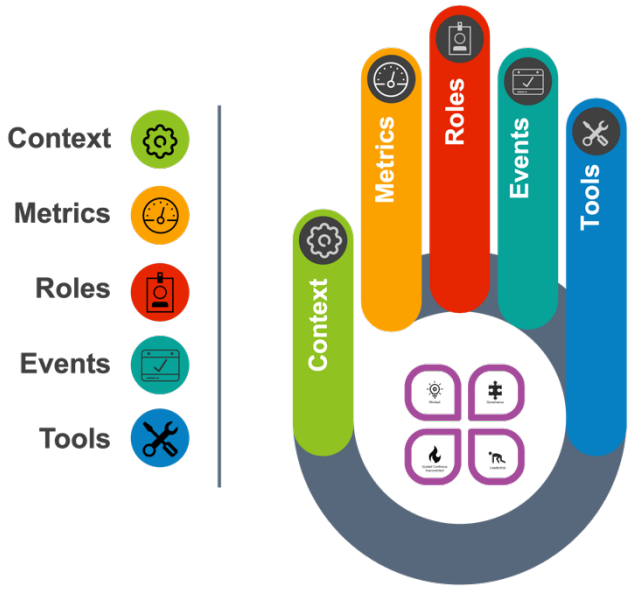

Figura 16 – ModernPM Lifecycle Toolkit (MLT)

Nello specifico si hanno quindi:

- **Context**: il contesto, in termini di progetto e complessità organizzativa, è il driver fondamentale nella scelta degli opportuni lifecycle.
- **Metrics,** le metriche giuste permettono al gruppo di lavoro di validare i miglioramenti ottenuti.
- **Roles**, in funzione dei lifecycle individuati, è fondamentale identificare e supportare i ruoli annessi.
- **Events**, gli specifici eventi aiutano il gruppo di lavoro ad allinearsi nei diversi aspetti operative e con gli stakeholder di progetto.
- **Tools**, per ogni lifecycle esistono specifici tool che aiutano nelle diverse fasi di gestione ed operative.

Ogni MLT esplicita quanto operativamente è necessario per implementare adeguatamente i lifecycle selezionato e come supportarne l'applicazione.

In relazione agli strumenti e gli output principali da ottenere per ogni fase di progetto, è possibile definire *"il bignami del ModernPM":*

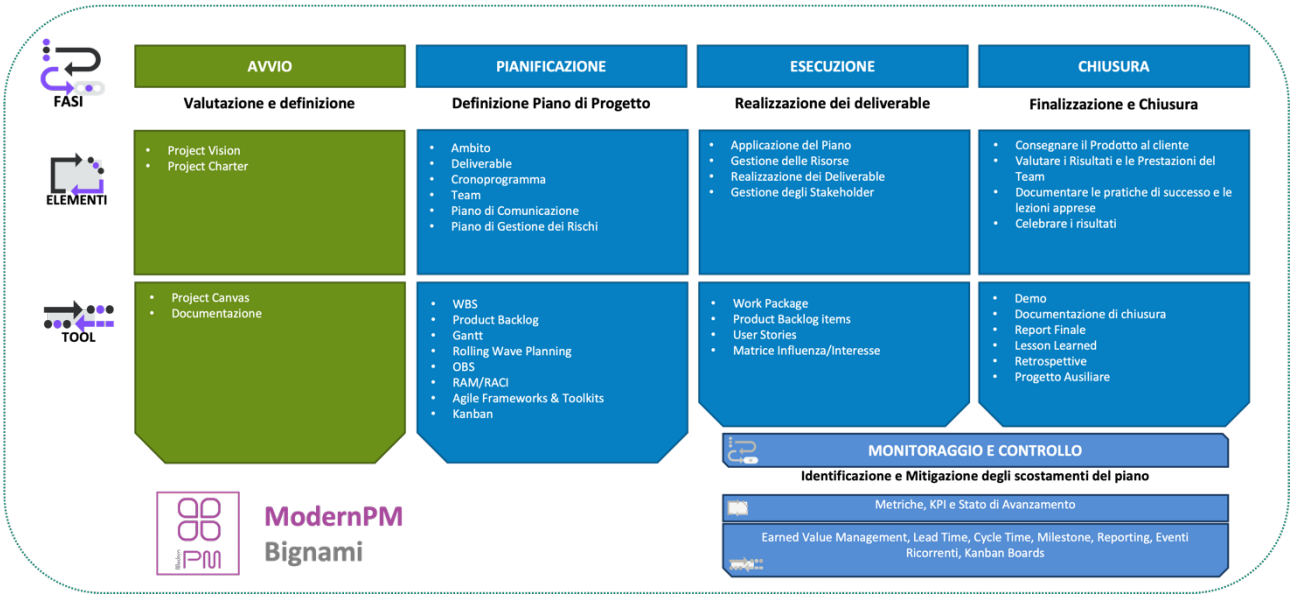

Figura 17 - Bignami del ModernPM

Andiamo ora a scoprire come tutto questo si specializza in relazione ai diversi lifecycle.

4.1 Lifecycle Predittivo

Nell'approccio **predittivo**, noto anche come *waterfall* o *serial*, l'ambito del progetto, i tempi ed i costi annessi sono determinati il prima possibile.

La capacità di sviluppare piani dettagliati, prima dell'esecuzione, consente al project manager di identificare le fasi di riferimento e la sequenza delle attività che andranno svolte in esse, cercando di ottimizzare i costi annessi e l'impatto dei rischi identificati.

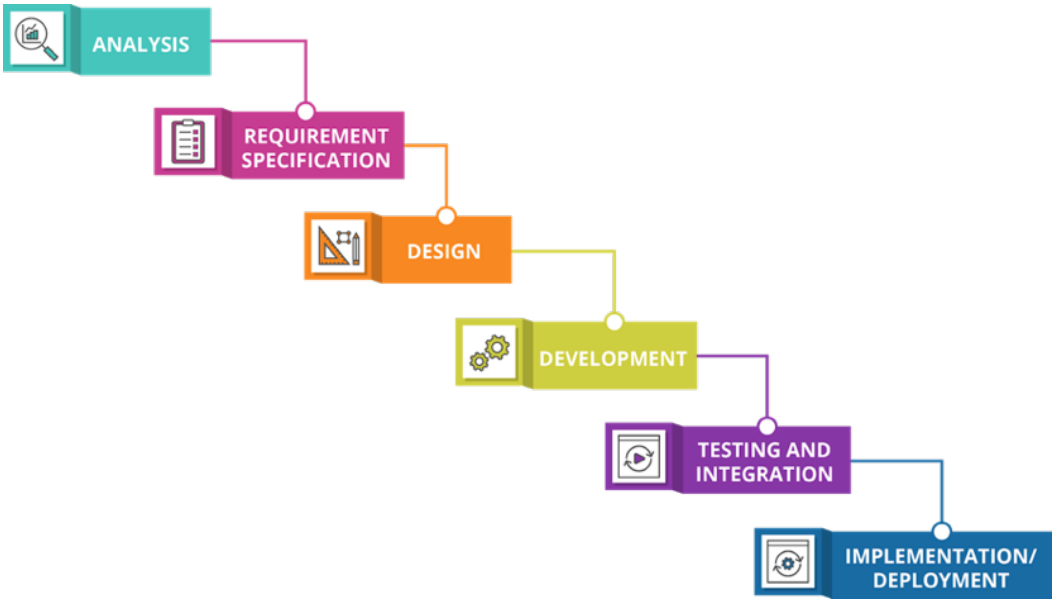

Figura 18 - Predittivo (Waterfall, Serial)

Tale approccio è caratterizzato da *fasi*, ognuna delle quali focalizzata su specifici deliverable e nella quale possono essere coinvolte diversi team. Il risultato finale di ogni deliverable non è, generalmente, reso disponibile individualmente, ma solo in forma aggregata alla fine del progetto.

Sebbene l'ambito del progetto sia ben definito all'inizio, non di rado è necessario effettuare azioni di ri-pianificazione e revisione del budget di un progetto, con un effort spesso non indifferente.

In generale, i cicli di vita predittivi sono più adatti a progetti che hanno risultati finali definibili a priori e ben dettagliati.

Il **Lifecycle Predittivo** è utile quando il grado di incertezza, su cosa fare (scope) e come farlo, è veramente minimo, abbinato al fatto che il valore del prodotto in oggetto può essere ottenuto solo alla fine della produzione.	
Context	• **Work Package** (WP) dettagliabili a livello alto. • **Persone** anche non totalmente dedicate al progetto ma coinvolte in relazione ai WP. • **Pochi cambiamenti** nelle specifiche e nelle condizioni a contorno.
Roles	• **Project Manager (PM)** • **Product Owner (PO)**, opzionale • **Scrum Master (SM)**, opzionale • **Architecture Owner (AO)**

	• **Developers** • **Specialisti**
Events	• **Meeting di allineamento**, si tengono in relazione a quanto stabilito dal piano operativo (Gantt) • **SAL**, evento periodico di allineamento sullo stato delle lavorazioni. • **Eventi on Demand**, si tengono quando è necessario discutere di tematiche specifiche
Tools	• **Work Breakdown Structure (WBS)**, è lo strumento di decomposizione nelle attività atomiche del prodotto, progetto. • **Organizational Breakdown Structure (OBS)**, la definizione gerarchica del team di progetto. • **RAM/RACI**, la matrice di assegnazione delle responsabilità. • **Gantt**, il più comune strumento di visualizzazione del piano operativo. • **Matrice Influenza/Interesse**, per la classificazione egli stakeholder. • **Documentazione formale e Report**.
Metrics	• **Earned Value Management (EVM)**, consente il monitoraggio attivo in relazione a: costi sostenuti, valore ottenuto e valore pianificato. • **Milestone**, valutazione degli obiettivi a breve, medio e lungo periodo.

4.2 Lifecycle Iterativo e Incrementale

Anche se spesso vengono confusi, esistono delle specifiche differenze tra un lifecycle **iterativo** ed uno **incrementale**.

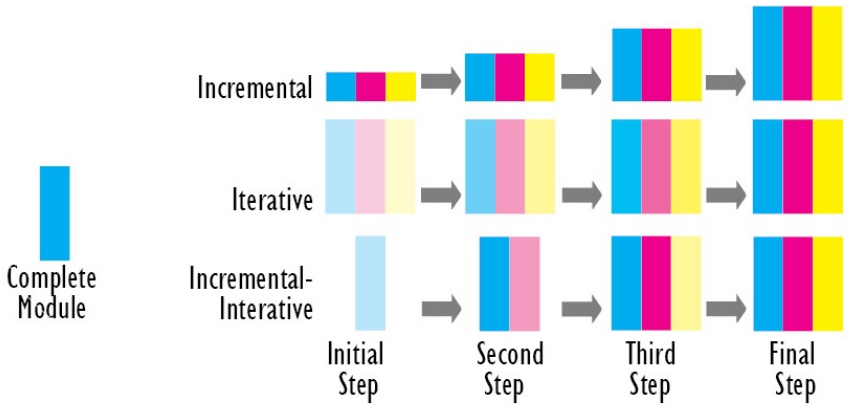

Figura 19 – Iterativo, Incrementale ed Iterativo + Incrementale

L'obiettivo del ciclo di vita **iterativo** è quello di raggiungere la *correttezza della soluzione*.
L'ambito del progetto viene generalmente determinato all'inizio del ciclo di vita del progetto, ma le stime dei tempi e dei costi vengono regolarmente modificate man mano che aumenta la comprensione del prodotto da parte del team di progetto.
La ripetizione (o iterazione) di una o più attività di progetto consente di approfondire la comprensione del prodotto da parte del team di progetto (ed eventualmente del cliente), consentendo di sviluppare progressivamente piani di sviluppo e di gestione dei rischi. Le iterazioni consentono, quindi, di sviluppare i deliverable attraverso una serie di cicli ripetuti, rappresentando l'elemento chiave utilizzato per guidare il valore nella direzione corretta. Sebbene vi sia il concetto di "incrementi", questi non sono deliverable distinti, ma piuttosto funzionalità o perfezionamenti di un singolo risultato finale per fase.
Un progetto con uno o più prototipi, ma con un solo un risultato finale, è un esempio di approccio iterativo

Dualmente all'approccio iterativo, quello **incrementale** si concentra sull'ottenere rapidamente una soluzione, sviluppandone le componenti in maniera incrementale. In un progetto incrementale, vengono creati continuamente nuovi deliverable utilizzabili che producono un aumento delle funzionalità del prodotto.
A differenza di un progetto predittivo, l'avere iterazioni al termine delle quali si ottengono elementi finiti, consente ai requisiti di essere dinamici per quelle parti del progetto che devono ancora essere pianificate ed eseguite.

Un esempio di sviluppo incrementale è la realizzazione delle diverse sezioni di un portale web che possono essere utilizzate, significativamente, in modo indipendente.

Nella pratica, gli approcci iterativi e incrementali vengono spesso utilizzati congiuntamente, anche in modo implicito, esistendo significativamente sovrapposizione: un lifecycle iterativo utilizza incrementi in una certa misura e un ciclo di vita incrementale utilizza iterazioni in una certa misura.

In generale, i lifecycle iterativi e/o incrementali vengono preferiti quando:

1. Il progetto è troppo complesso per essere pianificato fin dall'inizio e i feedback (o le lesson learned) sono lo strumento principale per migliorare le fasi successive.
2. L'ambito e gli obiettivi sono soggetti a modifiche man mano che le parti interessate apprendono più aspetti del progetto stesso.
3. Molte delle decisioni afferiscono ai risultati finali parziali o incrementali, a differenza dell'approccio predittivo dove i deliverable sono disponibili solo al termine del progetto.
4. Il progetto complessivo potrebbe non essere ancora considerato completo, ma gli incrementi forniti finora possono essere utilizzati per fornire anticipatamente al cliente.
5. Si decide di far ricorso concreto al *rolling wave planning*, in cui la fase successiva può avvenire mentre il team di progetto esegue la fase corrente.

Un approccio iterativo ed incrementale può essere caratterizzato da iterazioni di lunga durata e pianificazioni dettagliate, ricordando nello specifico una sequenza di cicli predittivi. Quando, invece, un approccio iterativo ed incrementale si basa su iterazioni brevi, spesso indipendenti tra loro e prive di una pianificazione estrema, si passa in modo naturale ad un lifecycle di tipo *Adattativo*, anche più comunemente noto come *Agile*.

4.3 Lifecycle Agile

Un ciclo di vita **Agile** (o Adattativo) è anch'esso contemporaneamente incrementale ed adattativo, ma è caratterizzato da iterazioni corte (tipicamente nell'ordine di qualche settimana) e dall'applicazione robusta del *rolling wave planning*. Ciò consente a questo approccio di affrontare rischi e rivedere i costi in relazione all'evoluzione del contesto.

L'approccio agile ha l'obiettivo di rendere meglio gestibili i progetti che prevedono un'elevata incertezza e un'elevata frequenza di cambiamento, richiedendo un alto grado di coinvolgimento continuo delle parti interessate. Nel cuore dell'azione vi è il concetto di *time-boxing*, ovvero il fatto che ogni evento/riunione/azione ha, normalmente, un tempo limite ben definito, accompagnato anche da una cadenza stabile.

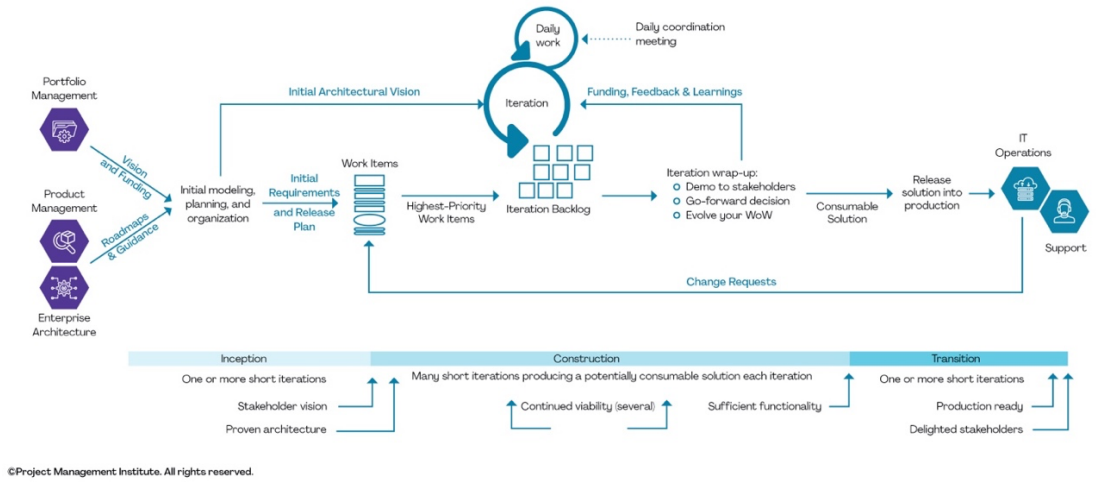

Figura 20 - Agile Lifecycle (PMI Disciplined Agile)

Ogni iterazione viene utilizzata per ridurre al minimo l'incertezza dei requisiti e avere piani contestuali aggiornati, andando a chiudersi con un incremento che viene valutato al suo termine. La relativa pianificazione avviene all'inizio dell'iterazione stessa, con il team che determina quanti dei requisiti di priorità più alta potranno essere completati. A valle dell'iterazione il team identifica idee per migliorare il processo ed eventuali nuove funzionalità, andando ad incidere sull'iterazione successiva.

Un progetto eseguito con un approccio Agile più essere visto come una serie di loop operativi di breve durata in cui il feedback viene raccolto continuamente, permettendo così di effettuare modifiche economicamente sostenibili, anche in fase avanzata di sviluppo del progetto.

Guardando al Lifecycle Toolkit di riferimento, si individuano i seguenti elementi caratterizzanti:

Un lifecycle Agile è, generalmente, il migliore per le situazioni in cui l'ambito e i requisiti non possono essere determinati in anticipo con dettaglio, o devono rimanere flessibili per adattarsi ai mutevoli fattori caratterizzanti l'organizzazione ed il mercato di riferimento. È indicato quando è possibile *identificare*, *prioritizzare* e *stimare* le diverse funzionalità da realizzare.	
Context	• I deliverable sono costituiti principalmente da miglioramenti o nuove funzionalità. • I deliverable possono essere identificati, prioritizzati e stimati in una fase di pianificazione relativa. • Il Team è dedicato principalmente ad un progetto.

Roles	• **Project Manager (PM)** • **Product Owner (PO)** • **Scrum Master (SM)** • **Architecture Owner (AO)** • **Technical Expert and Developers**
Events	• **Iteration**, definisce la finestra temporale di riferimento per sviluppare un nuovo incremento. È sempre accompagnato da un goal di riferimento. L'Iteration è nota anche come Sprint. • **Iteration Planning**, è l'evento di apertura dell'iterazione durante il quale vengono definiti i deliverable da realizzare nell'immediato. • **Daily Stand-up**, si tiene ogni giorno, possibilmente alla stessa ora, per consentire al team di fare il punto sulla situazione. • **Iteration Review**, ha lo scopo di mostrare l'Increment di prodotto agli stakeholder e raccoglierne i feedback. • **Iteration Retrospective**, permette al team di interrogarsi sulla validità del processo e su come migliorarsi a partire sin dalla prossima iterazione. • **Refinement**, si concentra sull'aggiornamento del Product Backlog in modo che sia costantemente allineato sulle aspettative degli stakeholder.
Tools	• **Story Mapping**, permette di illustrare le funzionalità del prodotto dal punto di vista dell'utente e decomporle a livello di user story da inserire nel Backlog. • **Product Backlog**, è l'elenco prioritizzato dei deliverable che gli stakeholder si attendono dal prodotto. Dovrebbe sempre essere associato ad un goal di prodotto. • **Iteration Backlog**, è l'insieme delle user story da realizzare nella prossima iterazione. Dovrebbe sempre essere associato ad un goal specifico. • **Iteration Board**, permette una rappresentazione grafica dello stato di avanzamento delle attività dell'iterazione. • **BurnDown/BunUp**, permette di visualizzare in maniera immediata ed efficace l'andamento delle attività durante l'esecuzione dell'iterazione.

	• **Definition of Done**, è una sorta di shortlist delle cose da fare affinché una funzionalità possa essere ritenuta completata.
Metrics	• **Velocity**, misura della capacità del team di sviluppare quanto previsto all'Iteration Backlog. • **Story Points**, metodo di stima relativo della complessità delle user story. • **Minimum Time To Recovery (MTTR)**, registra il primo momento in cui è possibile eseguire il ripristino di un determinato problema.

Quando il progetto attenzionato si occupa dell'evoluzione e della stabilizzazione di un prodotto già esistente, è possibile avvalersi di una specifica declinazione del lifecycle Agile in ottica Continuous Delivery, definita da PMI DA come: "**Continuous Delivery: Agile**".

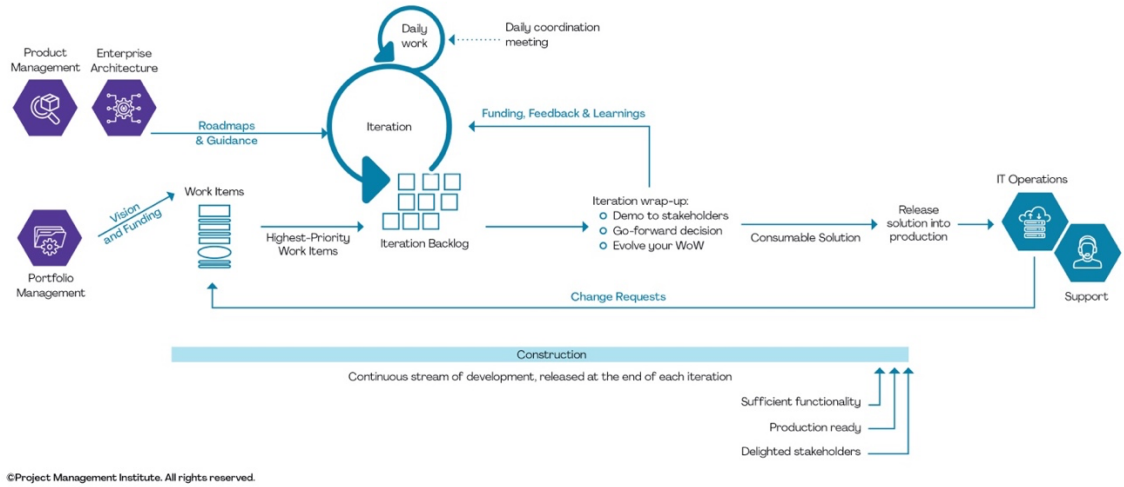

Figura 21 - Continuous Delivery: Agile

Pur mantenendo tutte le caratteristiche del lifecycle agile, questa specializzazione prevede normalmente iterazioni di una settimana (massimo 2) e richiede la capacità di delivery automatizzata tipica di una matura adozione della filosofia DevOps[8], ad ogni iterazione.

Come detto, un lifecycle "Continuous Delivery: Agile" è, generalmente, il migliore per le situazioni in cui l'obiettivo è quello di stabilizzare un prodotto da un punto di vista della qualità specifica, senza modificare le funzionalità offerte.

[8] https://www.pmi.org/disciplined-agile/process/disciplined-devops

4.4 Lifecycle Lean

Dualmente all'approccio Agile, esiste una seconda modalità operativa che ricade sotto il cappello adattivo, basata sulla gestione e ottimizzazione del flusso operativo, ovvero **Lean**.

Pur condividendo molti aspetti chiave dell'approccio Agile (come, ad esempio, feedback rapidi e definizione dei piani just-in-time), non è presente il concetto di iterazione in modo esplicito, bensì l'attenzione è sul *flusso di valore continuo*. Quando un nuovo deliverable viene richiesto, questo può essere subito messo in lavorazione, a patto che venga rispettato il limite delle attività concorrenti per il team, limite definito *Work-in-Progress limit* (*WIP limit*). Va da sé che anche nell'approccio Lean, ogni occasione è valida per discutere sull'andamento del processo e sull'evoluzione del progetto stesso.

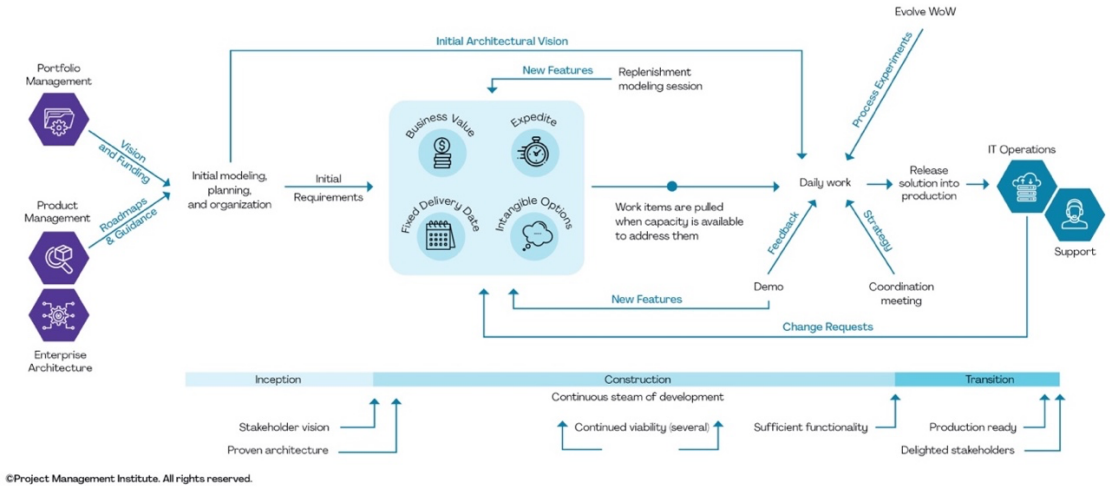

Figura 22 – Lean Lifecycle (PMI Disciplined Agile)

A differenza dell'approccio Agile, non vi è quindi una pianificazione di iterazione, ma il tutto avviene in modo rapido (e organico) rispetto agli eventi esterni. Inoltre, gli eventi/riunioni non hanno una cadenza predefinita, ma avvengo a "chiamata" quando necessari.

Guardando al Lifecycle Toolkit di riferimento, si individuano i seguenti elementi caratterizzanti:

Il **Lean Lifecycle** è orientato alla gestione di un flusso continuo di attività, rilasciando gli incrementi (o i singoli deliverable) quando sono pronti, senza avere una finestra specifica di riferimento, e sforzandosi di ridurre al minimo il tempo che intercorre tra due rilasci.
Un esempio tipico di utilizzo di un approccio Lean è nell'ambito dei progetti di ricerca, quando i deliverable da realizzare cambiano ad un ritmo molto elevato, giornaliero o addirittura nell'ordine delle ore.

Context	• Deliverable suddivisi in work item piccoli e della stessa. • Numero costante e ridotto di attività concorrenti. • Team dedicati principalmente ad un progetto. • Eventi on-demand, ovvero svolti quando ritenuto necessario.
Roles	• **Product Owner (PO)** • **Scrum Master (SM)** • **Architecture Owner (AO)** • **Developers**
Events	• **Demo**, si tiene quando il team ha sviluppato uno o più deliverable che ritiene utile mostrare agli stakeholder, ed eventualmente al cliente, per continuare le proprie attività, raccogliere feedback o effettuare un rilascio in produzione. • **Retrospective**, permette al team di interrogarsi sulla validità del processo e su come migliorarsi a partire dalla prossima iterazione. • **Daily Stand-up**, si tiene ogni giorno, possibilmente alla stessa ora, per consentire al team di fare il punto sulla situazione. • **3 Buckets planning**, la "tecnica dei tre secchi", permette di effettuare un raffinamento successivo dei deliverable, fino al grado di dettaglio che ne consente la pesa in carico.
Tools	• **Kanban Board**, è lo strumento di gestione visuale dell'avanzamento del lavoro. Si basa su una serie di stati di lavorazioni sequenziali che hanno capacità limitata (WIP limits). • **Classes of Services**, ogni nuovo deliverable viene categorizzato in relazione ad una specifica "Classe di Servizio". Nell'insieme, le classi di servizio permettono di gestire le priorità di lavorazione e i cambiamenti in corso di sviluppo. • **Control Chart**, si tratta di un grafico che presenta valori massimi/minimi (calcolati su base statistica) che permettono di intercettare eventuali situazioni di lavoro divergenti, soprattutto quando continuative.

	• **CFD (Cumulative Flow Diagram)**, permette di visualizzare ed analizzare l'andamento delle lavorazioni in relazione agli stati della Kanban Board e alle metriche di riferimento. • **Definition of Done**, è una sorta di shortlist delle cose da fare affinché una funzionalità possa essere ritenuta completata.
Metrics	• **WIP (Work in Progress)**, numero di attività in lavorazione concorrente. • **Lead Time**, tempo totale di evasione di una richiesta, da quando la stessa viene accettata (entra in una Classe di Servizio) a quando viene resa disponibile al cliente. • **Cycle Time**, tempo totale di lavorazione, ovvero da quando entra nello stato "lavorazione in corso" fino a quando viene contrassegnata come "completata". È un sotto-intervallo del lead time. • **Ideal Days**, calcolo del tempo/effort annesso allo sviluppo di un deliverable in senso assoluto, quindi non legato al tempo/gg. • **Minimum Time To Recovery (MTTR)**, registra il primo momento in cui è possibile eseguire il ripristino di un determinato problema.

Esattamente come accade per il lifecycle Agile, anche per quello Lean è possibile individuare una declinazione in chiave Continuous Delivery, che prevede un rilascio continuo in produzione di quanto realizzato: "**Continuous Delivery: Lean**".

L'obiettivo è quello di ridurre al minimo il tempo di delivery, anche con frequenza di più volte al giorno, sfruttando strumenti di automazione e processi standardizzati nel tempo.

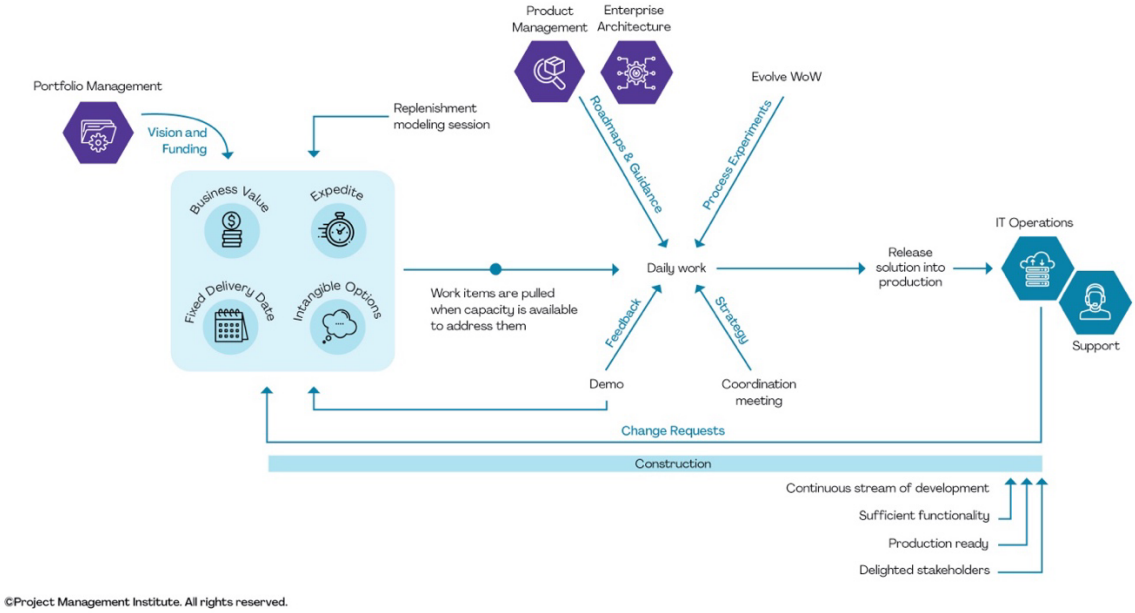

Figura 23 – Continuous Delivery: Lean Lifecycle (PMI Disciplined Agile)

Un lifecycle "Continuous Delivery: Lean" è, generalmente, il migliore per le situazioni in cui l'obiettivo è quello di consentire al cliente, o agli stakeholder in generale, una immediata risposta alle proprie richieste.

Anche in questo caso è richiesta una padronanza delle pratiche e tecniche di DevOps.

4.5 Lifecycle Exploratory

Il lifecycle **Exploratory** è ispirato a *Lean Startup*[9], pensato per validare la sostenibilità di una nuova idea e/o ipotesi grazie ad una serie di *Minimum Viable Product* (MVP) e una specifica implementazione del ciclo Build-Measure-Learn.

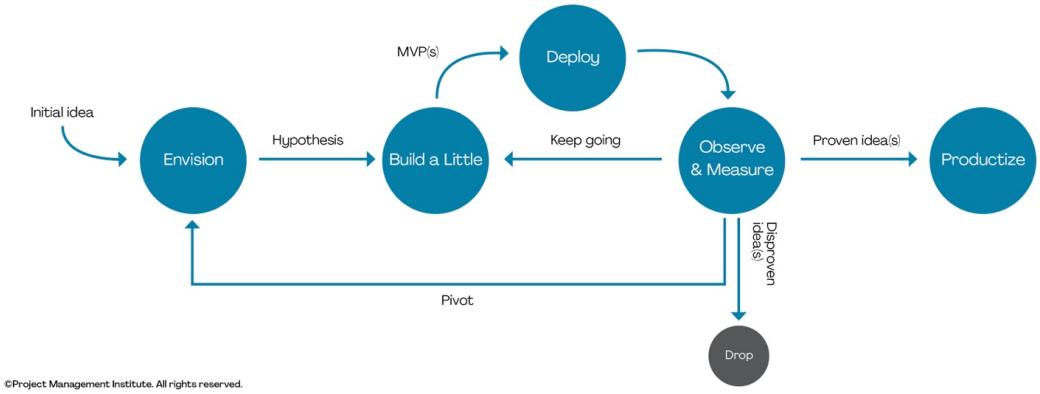

Figura 24 - Exploratory (Lean Startup) Lifecycle

[9] http://theleanstartup.com/principles

L’obiettivo è quello di ridurre il range di incertezza e ragionare sulla sostenibilità dell’iniziativa attraverso sei azioni di riferimento:

- **Envision**, si tratta di esplorare la nuova idea e formulare le ipotesi relative.
- **Build a Little**, sviluppare velocemente quanto necessario per validare le proprie ipotesi: *Minimum Viable Product (MVP).* Non è importante concentrarsi sulla qualità tecnica, bensì essere estremamente veloci e capaci di evidenziare le caratteristiche peculiari.
- **Deploy**, l’MVP deve essere velocemente testato in modo da comprovarne il valore.
- **Observe & mesure**, osservando le reazioni delle persone coinvolte nel test è possibile rendersi conto se quanto ipotizzato è sulla direzione giusta o è necessario cambiare.
- **Cancel**, se dopo aver testato gli aspetti portanti e sfruttato i pivot non si riesce a centrare l’obiettivo, è opportuno considerare l’dea/ipotesi come “non validata” e scartarla.
- **Productize**, se invece si raggiunge la validazione, si può passare alla fase di ingegnerizzazione tramite uno degli altri lifecycle presentati.

L’**Exploratory Lifecycle** è molto utile quando il progetto (o il deliverable) presenta una elevata incertezza, come, ad esempio, un nuovo mercato inesplorato o un nuovo prodotto L’obiettivo è quello di ottenere la massima flessibilità nell'adattare la soluzione mentre viene sviluppata, andando a testare le Ipotesi e la strategia con chiari criteri go/no-go, sperimentando l’idea in base a quanto si apprende.	
Context	Alta incertezza generale sul come e sul cosa.
Roles	Se si è all’interno di un Progetto, si ereditano i ruoli del (dei) lifecycle di riferimento. In caso si sia in una vera e propria azione di start-up, si preferisce essere alla pari nella discovery. Il **PM** può però essere presente in qualità di guida e facilitatore del team.
Events	Attività di continua **Review, allineamento e generazione di feedback**
Tools	Si utilizza un approccio **Lean/Kanban style**, ereditando tutti i tool descritti precede mente nello specifico lifecycle.

Metrics	Si utilizza un approccio **Lean/Kanban style**, ereditando tutte le metriche descritte precede mente nello specifico lifecycle.

4.6 Lifecycle Program

Il lifecycle **Program** (anche noto come Team di Team) descrive come organizzare lo sviluppo di un deliverable (o del progetto) quando si ha necessità di *"scaling orizzontale"*, ovvero quando più team in parallelo lavorano su di esso. La soluzione di avere più team in parallelo è normalmente preferita rispetto ad avere un unico team di grandi dimensioni.

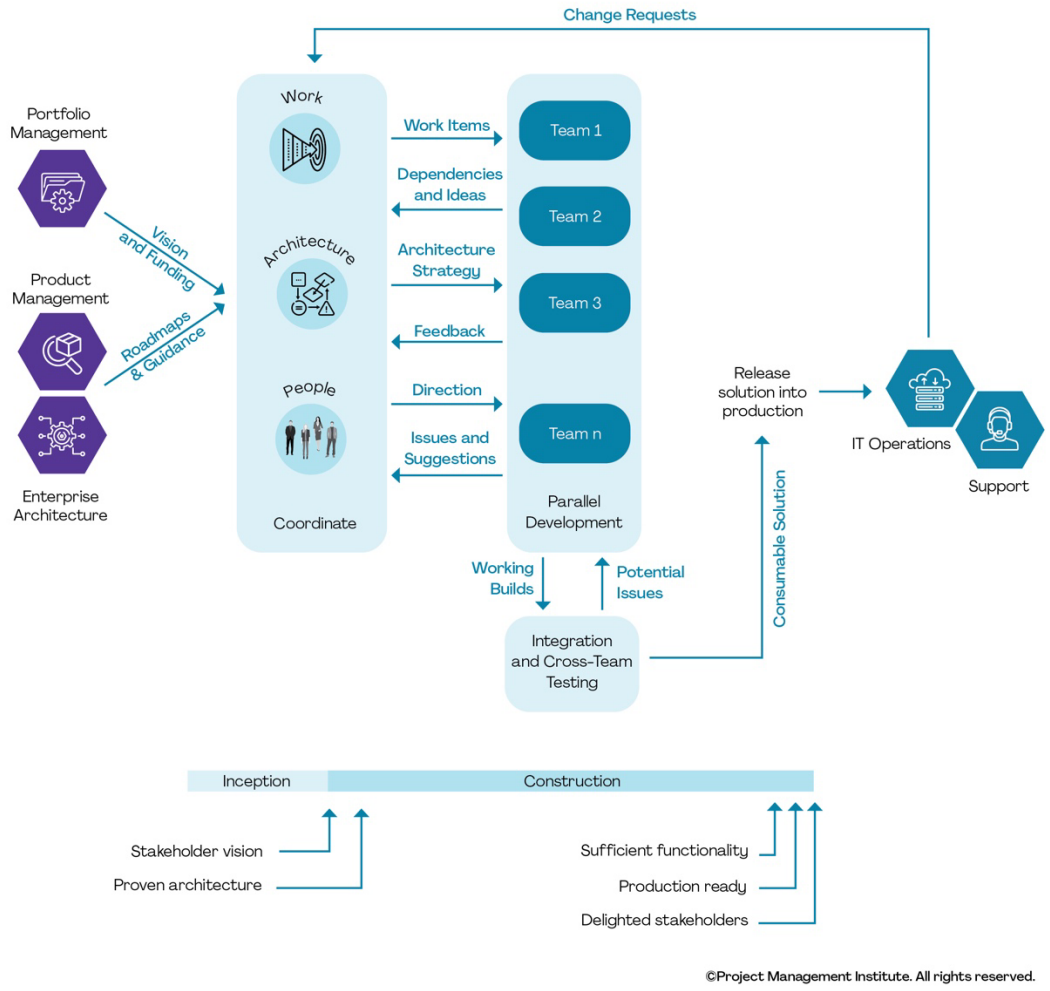

Figura 25 - Program Lifecycle (PMI Disciplined Agile)

Generalmente, il singolo team a sua volta può identificare il proprio lifecycle sfruttando una struttura di tipo bottom-up per allinearsi:

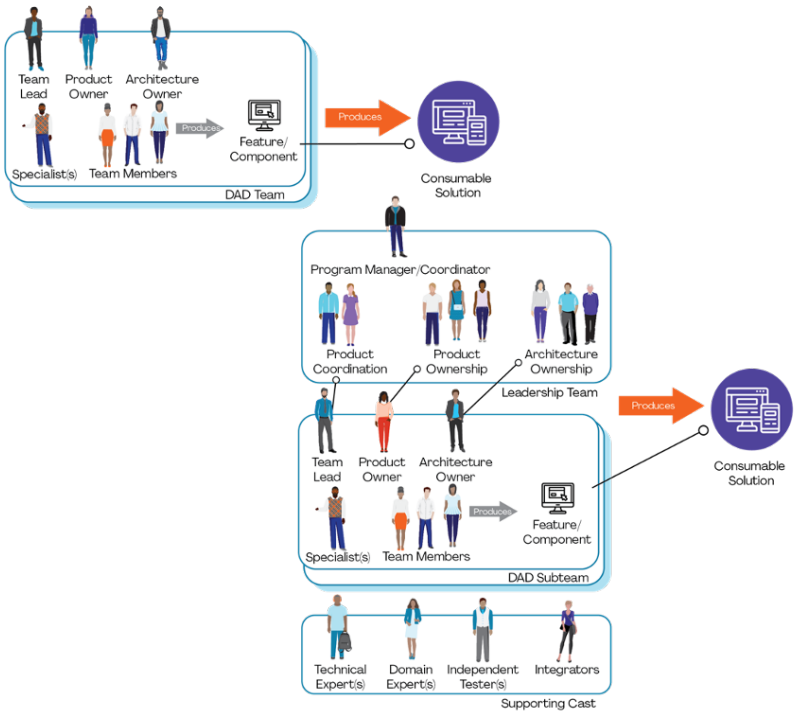

Figura 26 - Allineamento Bottom-up (Disciplined Agile)

Il **Program Lifecycle** è l'ideale quando si hanno più team al lavoro sullo stesso progetto/prodotto, promuovendo un modello organizzativo prevalentemente bottom-up in cui team agile o lean si coordinano senza appesantire il flusso organizzativo.	
Context	Si ha la necessità di avere decine di persone su un progetto con la volontà di mantenere la necessaria agilità operativa.
Roles	In funzione del tipo di lifecycle interno al team, si ereditano i ruoli del lifecycle Agile o Lean ed, opzionalmente, Waterfall. Inoltre, si creano 3 leadership team di coordinamento: • Product Coordination, formato da tutti gli SM • Product Ownership, formato da tutti i PO • Architecture Ownership, formato da tutti gli AO In aggiunta si ha un **Program/Coordinator/Project Manager** di riferimento.
Events	Ogni team utilizza quelli propri del lifecycle scelto, a cui si aggiungono: • **Integration and Cross-team testing**: sono gli eventi, o meglio le azioni, di integrazione del lavoro complessivo.

	• **System Review**, in cui quanto prodotto dall'insieme dei team viene discusso con gli stakeholder. • **Release in Production**, ovvero il rilascio (o consegna).
Tools	Ogni team utilizza quelli propri del lifecycle scelto, con l'utilizzo degli strumenti del mondo **Lean/Kanban** per il coordinamento ed allineamento.
Metrics	Ogni team utilizza quelli propri del lifecycle scelto, con l'utilizzo delle metriche del mondo **Lean/Kanban** per la valutazione d'insieme.

4.7 Lifecycle Hybrid

Un lifecycle **Ibrido** è una combinazione diversi cicli di vita.

Spetta al team determinare quale sia il mix più appropriato, garantendo che abbia la flessibilità necessaria per raggiungere al meglio gli obiettivi attesi.

In un lifecycle ibrido, invece di associare un unico lifecycle all'intero progetto, viene creato un mix di diversi cicli di vita in relazione alle caratteristiche dei deliverable previsti, analizzando le loro specificità e la strategia di rilascio annessa. In ogni caso, i deliverable vengono di solito condivisi con il cliente (e/o gli altri stakeholder) per mostrare gli avanzamenti e ottenere feedback preziosi.

I deliverable possono, inoltre, essere sviluppati in modo *sequenziale* e/o *parallelo*, e un progetto può avere entrambe le casistiche. Con deliverable sviluppati in modo *sequenziale*, lo sviluppo del successivo inizia solo dopo che quello precedente è stato completato e accettato. Questo approccio è consigliato quando i deliverable sono l'uno funzionale all'altro, quando ci sono forti dipendenze e quando non si hanno risorse per attuare uno sviluppo parallelo.

Nel caso di deliverable sviluppati in modo *parallelo*, le lavorazioni possono sovrapporsi (più team) o essere svolte contemporaneamente (un solo team). Questo tipo di approccio è consigliato quando i deliverable sono indipendenti l'uno dall'altro (o la dipendenza relativa è veramente minima) e si hanno team e le risorse in numero adeguato.

5 ModernPM Fast Prototyping

Il *ModernPM Fast Prototyping* (pratica ereditata ed estesa dall'AgileConstellation funnel) permette di definire rapidamente gli aspetti metodologici relativi ad un progetto specifico.

Figura 27 - ModernPM Fast Prototyping

Rispetto a quanto ereditato dalla pratica base[10], vengono aggiunti *4 nuovi aspetti* (bubble) di riferimento:

- **Visione (Vision)**, ovvero l'identificazione degli aspetti caratterizzanti il progetto specifico;
- **Scope (Ambito)**, l'ambito del progetto, ovvero le attività ed i deliverable da realizzare;
- **Complexity (Complessità**), analisi della complessità e dell'incertezza del progetto;
- **Lifecycles Bouquet**, la costruzione del lifecycle di riferimento, che può essere unico per tutto il progetto o costruito da un modello ibrido in relazione ai deliverable.

[10] AgileConstellation Manifesto – agileconstellation.info

5.1 Vision

La **Vision** è il punto di partenza di ogni nuova iniziativa, e, nel caso di un progetto, la relativa definizione permette di rispondere alla fondamentale domanda: **questo progetto merita di essere avviato ed è sostenibile per l'azienda?**

Per lavorare su questo tema si utilizza il **Project Canvas**, che ha l'obiettivo di "mettere al tavolo" tutti gli attori primari che ruotano intorno ad un progetto, al fine di discutere delle relative caratteristiche di riferimento.

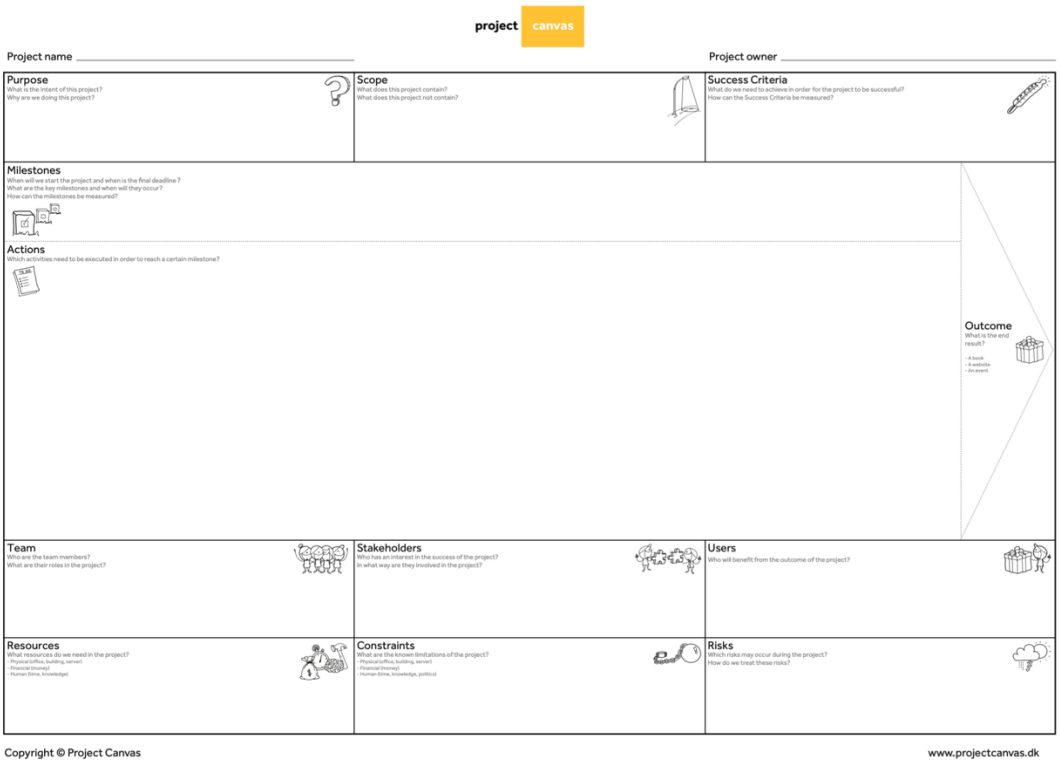

Figura 28 - Modello di Project Canvas (projectcanvas.dk)

Si tratta, in sintesi, di un *visual tool* che permette di focalizzare la discussione su un set di elementi fondamentali, migliorando la comunicazione del team di progetto stesso:

- **Purpose**: gli obiettivi e le motivazioni del progetto.
- **Scope**: cosa deve contenere e cosa non deve contenere il progetto, ad alto livello (nel paragrafo successivo si andrà a vedere come dettagliare questo aspetto).
- **Success Criteria:** come poter stabilire il "successo" del progetto.
- **Milestones**: i momenti di verifica del progetto.
- **Actions**: le azioni necessarie affinché venga raggiunto l'obiettivo.
- **Outcome**: i beneficio atteso.

- **Team**: i membri del team di progetto e il loro ruolo.
- **Stakeholders**: gli stakeholders di progetto.
- **Users**: i "consumatori" dei deliverable di progetto, ovvero del prodotto risultante.
- **Resources**: le risorse necessarie allo sviluppo del progetto: budget, persone, attrezzatura, software, ecc.
- **Constraints**: i vincoli, le limitazioni e le barriere del progetto.
- **Risks**: i rischi e le relative azioni di mitigazione.

5.2 Definizione dello Scope

Per l'analisi dei deliverable di progetto si possono utilizzare due strumenti di riferimento: la *Work Breakdown Structure* (*WBS*), con la definizione dell'annesso *dictionary* di riferimento, e lo *Story Mapping*.

5.2.1 Work Breakdown Structure

Nell'approccio predittivo (waterfall, serial) per la definizione delle attività, nonché la scomposizione di un progetto/prodotto nei suoi componenti, si utilizza tipicamente la *Work Breakdown Structure*, *WBS*.

La WBS consente di impostare il processo di pianificazione, accompagnando tutto il progetto lungo il processo di sviluppo e monitoraggio.

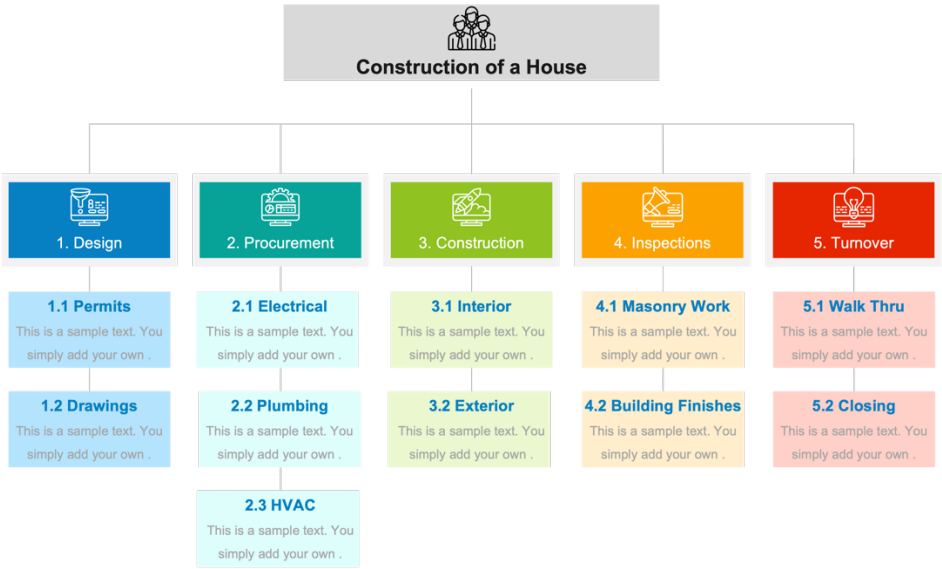

Figura 29 - Esempio di WBS

Partendo dalla *Vision*, la WBS permette di dettagliare lo *scope* di progetto (e/o prodotto) attraverso una sua specializzazione (scomposizione) ad albero che si conclude al livello di massimo dettaglio con i *Work Package* (WP).

I WP sono, dunque, i diversi elementi che compongono il progetto (prodotto) e sono accompagnati da tutte le informazioni di dettaglio che ne permettono la realizzazione, a cui è possibile collegare specifici membri del team o anche un singolo specialista.
Inoltre, la WBS permette la strutturazione del *reticolo delle attività*, consentendo la schedulazione e la definizione dei tempi di realizzazione, nonché una specifica correlazione del budget annesso.

Un approccio costruttivo alla WBS è quello di considerarla per quello che è: **non il piano, ma uno strumento di rappresentazione del piano**. In tal mondo non si corre il rischio di considerarlo come la "stella polare" ma comune un ausilio alla governance del progetto.

5.2.2 Story Mapping

Quando ci si sposta verso un approccio più adattivo (agile), si tende a focalizzarsi maggiormente su funzionalità rilasciabili velocemente ai consumatori, rappresentando, atomicamente, un nuovo "pezzettino" di valore. In questo ambito le **Storie** (o *User Story*) sono un po' diventate lo **standard de facto** per stabilire un agreement con il team di progetto, in sostituzione dei Work Package visti in precedenza.

Per poter però individuare il quadro d'insieme, similarmente alla WBS, viene spesso utilizzato lo **Story Mapping** che consente di creare una "*mappa*", *facile da visualizzare*, strutturata, condivisa e contestuale delle storie.

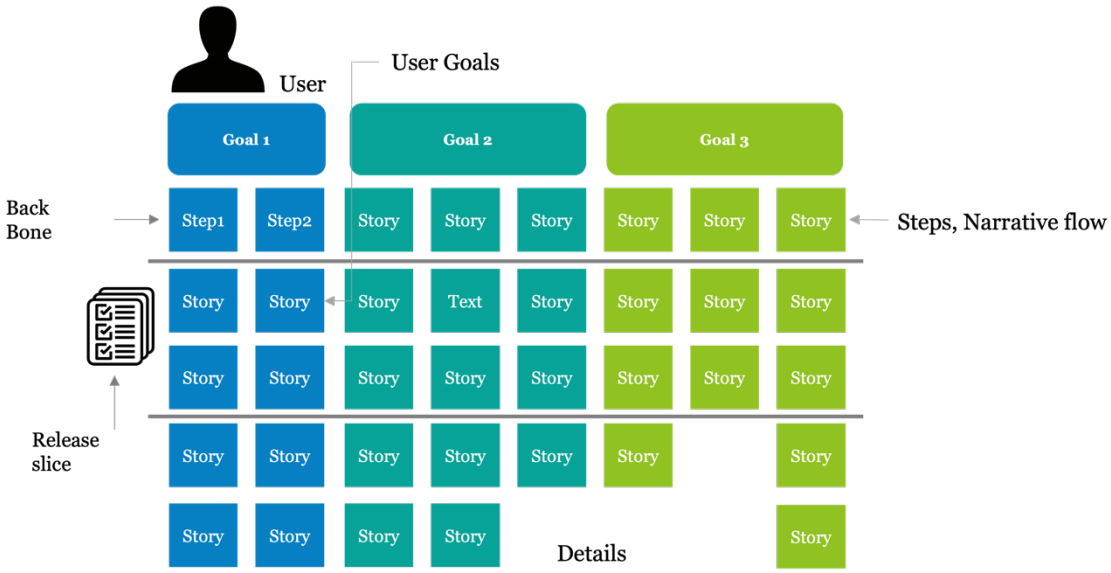

Figura 30 - Esempio di Story Mapping

Il processo di definizione della mappa può essere sintetizzato in tre azioni fondamentali:

1. **Identificare le funzionalità/attività di alto livello**: questa azione può essere eseguito esaminando l'interazione dell'utente (*user journey*) con la parte pertinente del sistema per cui si desidera creare la mappa. Le attività sono anche conosciute come **Goal** (anche *Obiettivi* o *Temi*). È fondamentale essere in grado di identificare come l'utente utilizza il nostro sistema, in modo tale da definire i requisiti di alto livello e condividerli, per una conferma, con le parti interessate. Una volta definiti gli obiettivi, gli stessi vanno disposti in ordine cronologico, da sinistra a destra, come se si stesse raccontando il viaggio dell'utente attraverso il sistema.
2. **Documentare i passaggi**: si tratta di specificare i passaggi (step) necessari per raggiungere l'obiettivo. Spesso gli step vengono etichettati come "epiche", in quanto tendono a comprendere gruppi correlati di storie utente (o attività) e, nell'insieme, costituiscono la *backbone* ("spina dorsale") della story map, fornendone gran parte della struttura.
3. **Identificare i dettagli**: a questo punto vengono identificare le effettive storie utente, nonché le attività di dettaglio richieste. Le storie andrebbero sempre sviluppate in collaborazione con le parti interessate, consentendo al team di implementare la "cosa giusta" nel "modo giusto".

Una volta realizzata la mappa, è possibile iniziare a identificare le diverse aree atomiche del prodotto, andando ad applicare le opportune strategie di prioritizzazione per lo sviluppo iterativo ed incrementale.

5.3 Complexity

Definito lo scope, si passa ad analizzare la complessità del progetto, o di uno specifico deliverable, al fine di identificare il lifecycle più efficace, ricorrendo allo **Stacey's Diagram**.

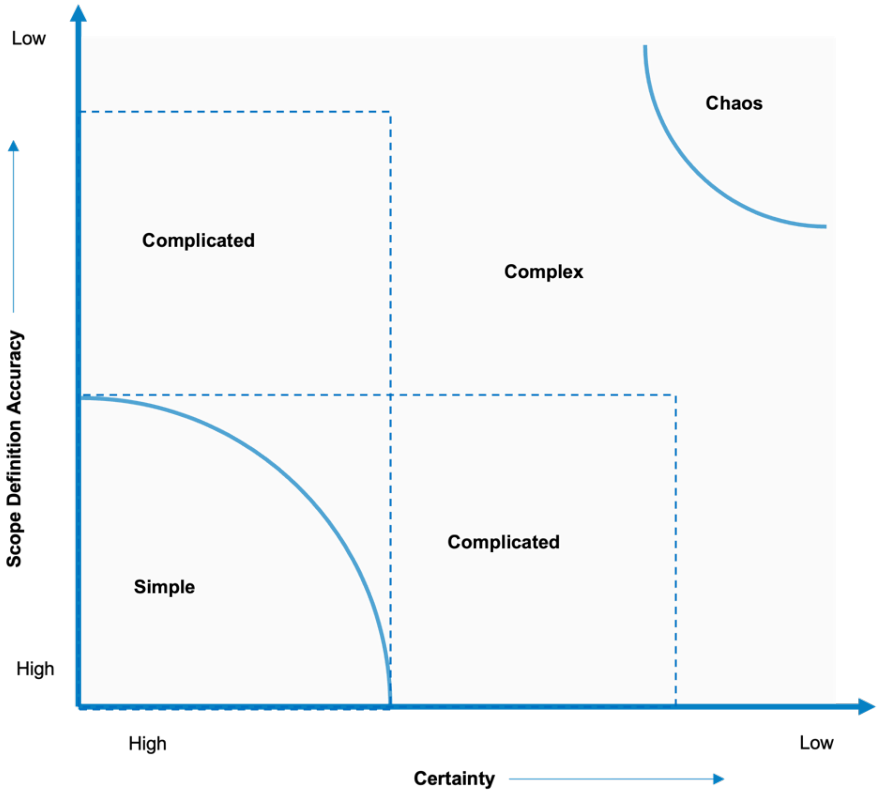

Figura 31 - Stacey's Diagram

In esso, sull'asse Y, si ha il livello di confidenza con lo scope (*Scope Definition Accuracy*), mentre sull'asse X, si ha l'incertezza (*Uncertainity*), ovvero il grado di incertezza sugli aspetti portanti, quali: tecnologia, skill, ecc..

In relazione alle possibili intersezioni ottenibili, è possibile delineare le seguenti aree di classificazione dei deliverable (o, eventualmente, del progetto nel suo complesso) e indentificare il (o i) lifecycle più adatto:

- **Simple (semplice)**, in basso a sinistra: si sa con buona precisione cosa fare e come farlo. In questo caso *l'approccio predittivo* è quello più idoneo.
- **Complicated (complicato)**, in questo caso si hanno 2 casistiche: in alto a sinistra si è in presenza di requisiti poco chiari e l'approccio consigliato è quello *iterativo*. Se invece ci si posiziona in basso a destra, quindi l'incertezza operativa aumenta, è preferibile riscorrere ad un lifecycle di tipo *incrementale*.
- **Complex** (**complesso**), area medio-alta. In questo caso si ha una forte incertezza sui requisiti, così su come realizzare le attività annesse, per cui è preferibile ricorre ad un lifecycle di Agile o Lean.

Come più volte sottolineato, un progetto può essere caratterizzato da più lifecycle durante la sua esecuzione, in funzione della sua tipologia, del tempo, delle risorse stimate e di altri fattori caratterizzanti.

Andando a sovrapporre lo spettro di applicazione dei lifecycle con il Diagramma di Stacey si ottiene, per cui, lo scenario seguente:

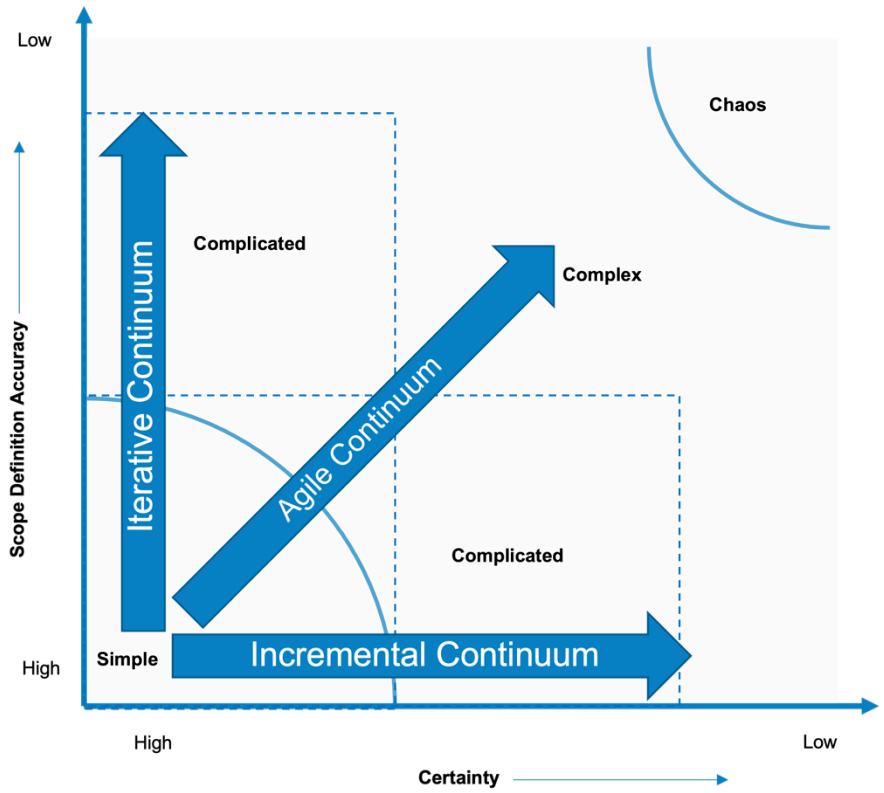

Figura 32 - Stacey's Diagram e lifecycles

Per fare la scelta più opportuna, bisogna quindi andare ad analizzare le due dimensioni utilizzando dei fattori di valutazione quanto più possibile oggettivi.

5.3.1 Scope Definition Accuracy

In relazione ai diversi deliverable identificati (Work Package o Epiche/Storie), o, nel caso limite, per l'intero progetto, si andranno considerati fattori determinati quali:

- **Numbers of Stakeholders**. Il numero degli stakeholder coinvolti sul progetto, sia interni che esterni all'organizzazione.
- **Stakeholders Engagement level**. Il livello di ingaggio degli stakeholder in termini di interesse/potere.
- **Assumed Frequency of Change**. La presunta stabilità dello scope, ovvero quanto si prevede che i requisiti siano stabili nel tempo.

- **Project Duration**. La durata del Progetto in termini temporali.

Tali fattori andranno persati per ottenere uno specifico rate complessivo del livello di confidenza con le specifiche dei deliverable:

Factors	**Level (1…5)**
Numbers of Stakeholders	
Stakeholders Engagement level	
Assumed Frequency of Change	
Project Duration	
Average rate	
*Rate: **from 1 to 2, Low level of Agreement***	
Rate: **3, Medium level of Agreement**	
Rate: **from 4 to 5, High level of Agreement**	

5.3.2 Uncertainty

Per l'analisi dell'incertezza caratterizzante i deliverable (anche qui è possibile riferirsi, se opportuno, all'intero progetto), si potrà fare riferimento ai fattori di *tactical* scaling individuati da Disciplined Agile[11]:

[11] https://www.pmi.org/disciplined-agile/agility-at-scale/tactical-agility-at-scale/scaling-factors

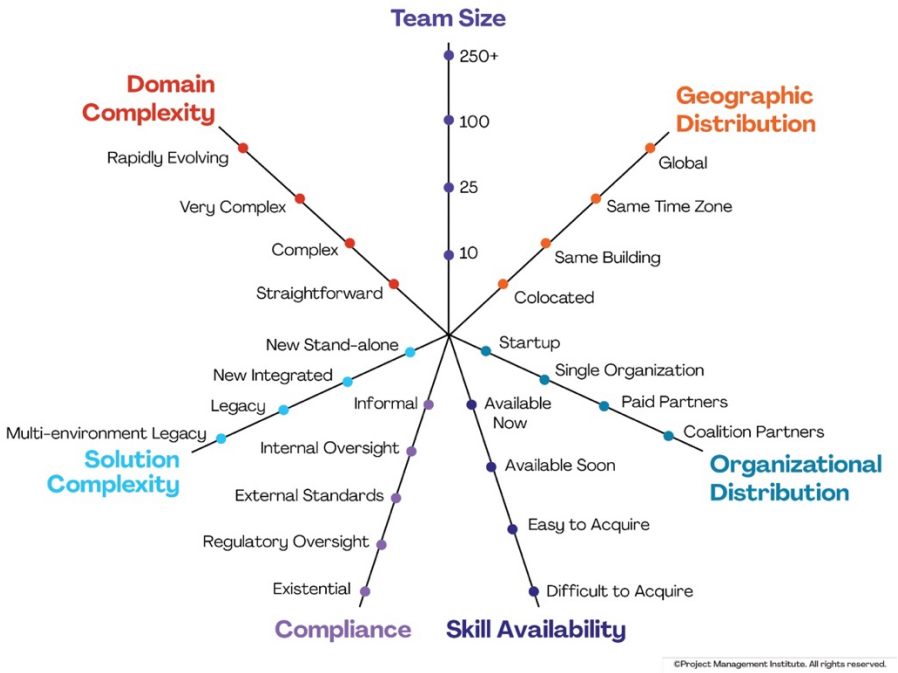

Figura 33 - Disciplined Agile Factors for Tactical Scaling

- **Team Size**. I team possono variare in dimensioni, da due persone a fino a diverse centinaia. I team più grandi (intesi anche come team di team) sono generalmente costituiti per affrontare problemi più complessi e sfide di una maggiore impatto e durata nel tempo.
- **Geographic distribution**. I team possono essere co-localizzati, con i membri del team e i principali stakeholder che condividono uno stesso spazio di lavoro. Possono essere sullo stesso piano in un singolo edificio, su più piani, alcuni dei membri del team possono lavorare in edifici diversi, alcuni possono lavorare da casa e alcuni possono persino lavorare in Paesi diversi.
- **Organizational Distribution**. La provenienza e le modalità di coinvolgimento delle persone funzionali al progetto possono essere diverse. La situazione più semplice è quella di avere tutti i membri del team dello stesso gruppo/divisione all'interno di una singola organizzazione, cosa che capita, ad esempio, in una startup o un nuovo team di prodotto all'interno di un'impresa.
- **Skill Availability**. Il team ha bisogno delle persone giuste con le giuste competenze per raggiungere i risultati che hai assunto. All'estremità ideale dello spettro si hanno persone qualificate "sedute in panchina" in attesa di andare avanti, all'altra estremità potrebbero essere necessari molti mesi, e potenzialmente molti soldi, per costruire il team di cui si ha bisogno.

- **Compliance**. Generalmente ci si confronta con due tipologie di conformità: *autoimposta*, ad esempio la scelta di essere conformi al CMMI o ad ITIL, e quella *normativa*, potenzialmente più difficile da soddisfare.
- **Domain Complexity**. La complessità del dominio (anche *spazio del problema*) può variare ampiamente in relazione agli obiettivi. Un sito web informativo, ad esempio, è generalmente semplice, mentre un sito di e-commerce è più complicato, così come un sistema di controllo del traffico aereo è decisamente più complesso.
- **Solution Complexity**. La soluzione può essere caratterizzata da diversi livelli di complessità intrinseca. All'estremità "semplice" dello spettro si possono identificare le soluzioni standalone nuove, costruite con nuove tecnologie. Le cose diventano più difficili se, ad esempio, è necessario sfruttare quanto precedentemente in essere, come software legacy e servizi aziendali pre-esistenti.

Anche in questo caso, in relazione al livello di complessità relativa dei diversi fattori di scaling, si otterrà un indicatore complessivo di deliverable (o progetto)che permetterà di identificare il livello relativa sul diagramma, nonché il posizionamento in un quadrante specifico grazie all'intersezione con il precedente livello di accuracy.

Factors	**Level (1…5)**
Team Size	
Geographic distribution	
Organizational Distribution	
Skill Availability	
Compliance	
Domain Complexity	
Solution Complexity	
Average rate	
*Rate: **from 1 to 2, Low level of Uncertainty***	
Rate: **3, Medium level of *Uncertainty***	
Rate: **from 4 to 5, *High level of Uncertainty***	

5.3.3 ModernPM Decision Tree

Una volta identificati i livelli delle due dimensioni del deliverable in analisi, è possibile utilizzare il **ModernPM Decision Tree** per scegliere il lifecycle più adatto al suo sviluppo:

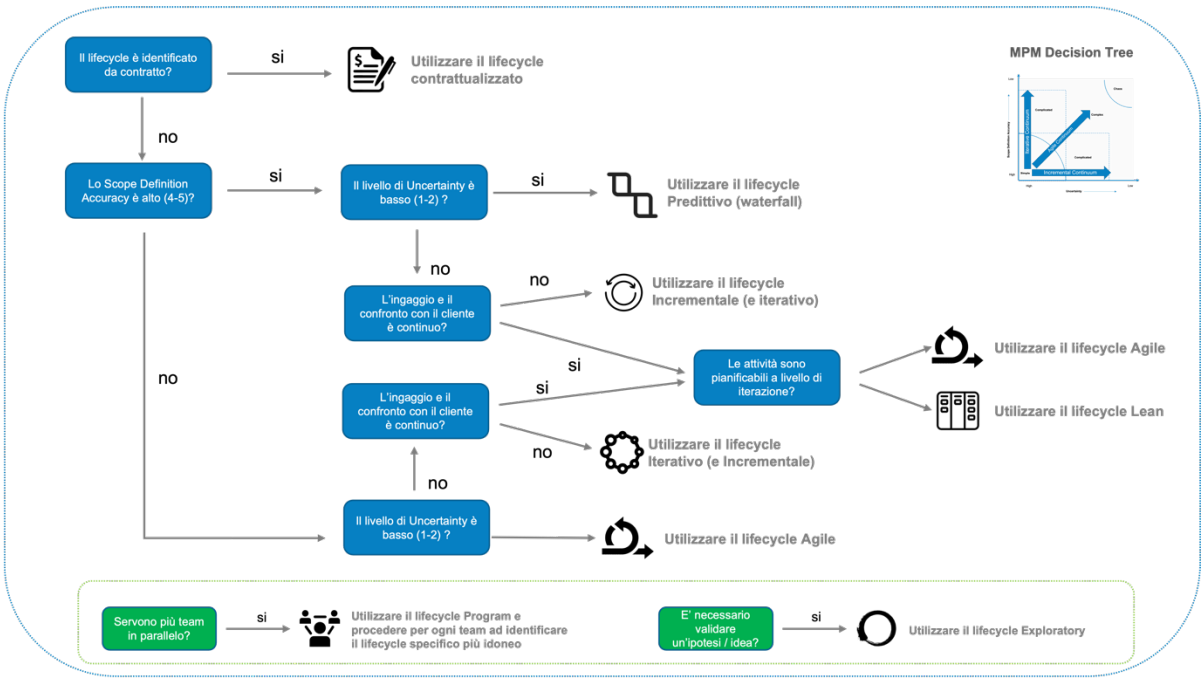

Figura 34 - ModernPM Decision Tree

Nel caso in cui il *decision tree* venga utilizzato a livello di progetto, il percorso seguito porta alla scelta del lifecycle unico di progetto. Se, invece, l'albero viene ripercorso per ogni deliverable, una volta ultimata l'analisi si otterrà un **modello ibrido**, generalmente rientrante in una delle casistiche riportate nella figura seguente:

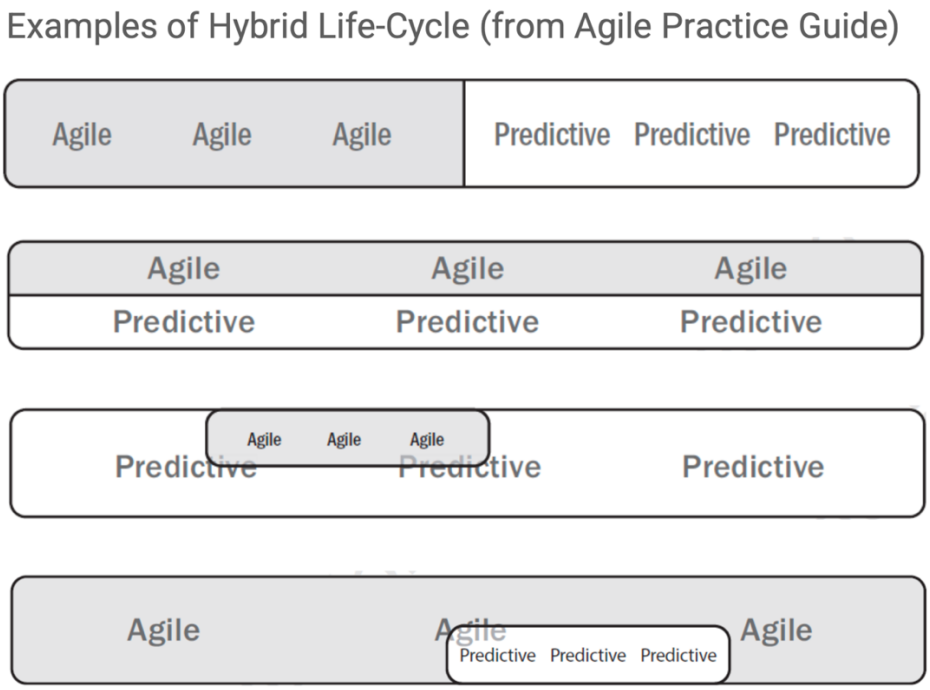

Figura 35 – Esempi di Hybrid Lifecycle (Agile Practice Guide)

5.4 Lifecyles Bouquet

Ultimata l'analisi di contesto tramite i tre bubble, è possibile procedere a *convalidare e formalizzare* l'approccio complessivo per la gestione ed esecuzione del progetto, ovvero il **Lifecycles Bouquet**.

Tale modello è articolato nei termini già visti nella descrizione dei singoli lifecycle, mixandoli opportunamente tra loro:

- **Lifecycles**, scelti come indicato sopra. Si può avere un solo lifecycle per progetto, o un mix di essi (situazione ibrida) in relazione alle caratteristiche dei deliverable da produrre.
- **Roles**, ovvero i ruoli necessari in relazione ai lifecycle selezionati.
- **Tools**, la cassetta degli attrezzi specifica.
- **Metrics**, le metriche necessarie a misurare le performance operative e applicare le necessarie attività di miglioramento continuo.

6 Richiami

6.1 PMI Disciplined Agile

Il **PMI Disciplined Agile toolkit** (**PMI DA toolkit**) guarda all'intera organizzazione, supportandone il relativo percorso verso la Business Agility. Le aree attenzionate dal toolkit (definite *Blade)* sono riportate nel poster seguente:

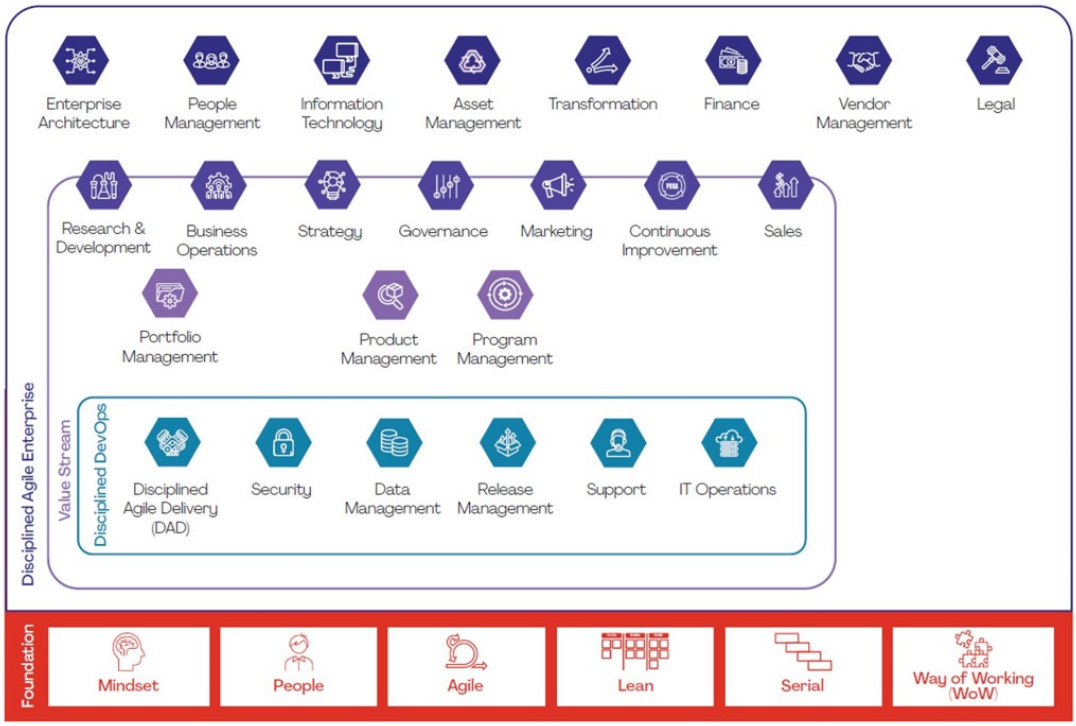

Figura 36 - The Disciplined Agile Toolkit

Come si vede, sono presenti 4 layer che impattano l'intera organizzazione: *Foundation*, *Disciplined DevOps*, *Value Stream* e *Disciplined Agile Enterprise*. In particolare, il layer Foundation mette a fattor comune tutti quegli aspetti che sono il cuore pulsante del toolkit, allineandolo con il mindset di riferimento.

Ogni layer racchiude più "Blade" (aree), presentando una raccolta coerente di opzioni (in particolar modo *pratiche* e *strategie*) da applicare in relazione al contesto specifico. Ogni Blade affronta un ambito "atomico", come, ad esempio: *Data Management*, *Continuous Delivery*, o *Portfolio Management*. La scelta del termine "Blade" è legata concettualmente all'omologo dei "Server Blade" per evidenziare la possibilità di aggiornarlo, cambiarlo, o addirittura sostituirlo, con estrema agilità, in relazione all'evoluzione del contesto in cui si sta operando.

Una particolare attenzione va posta al il layer **Value Stream**, che può essere visto come *l'anima dinamica* di DA, in quanto ricama l'operatività sulle necessità degli stakeholder, o cliente che sia,

abbracciando un mindset chiaramente Lean che suggerisce di concentrarsi sulla gestione del flusso di lavoro e non su come gestire le persone o le micro-attività.

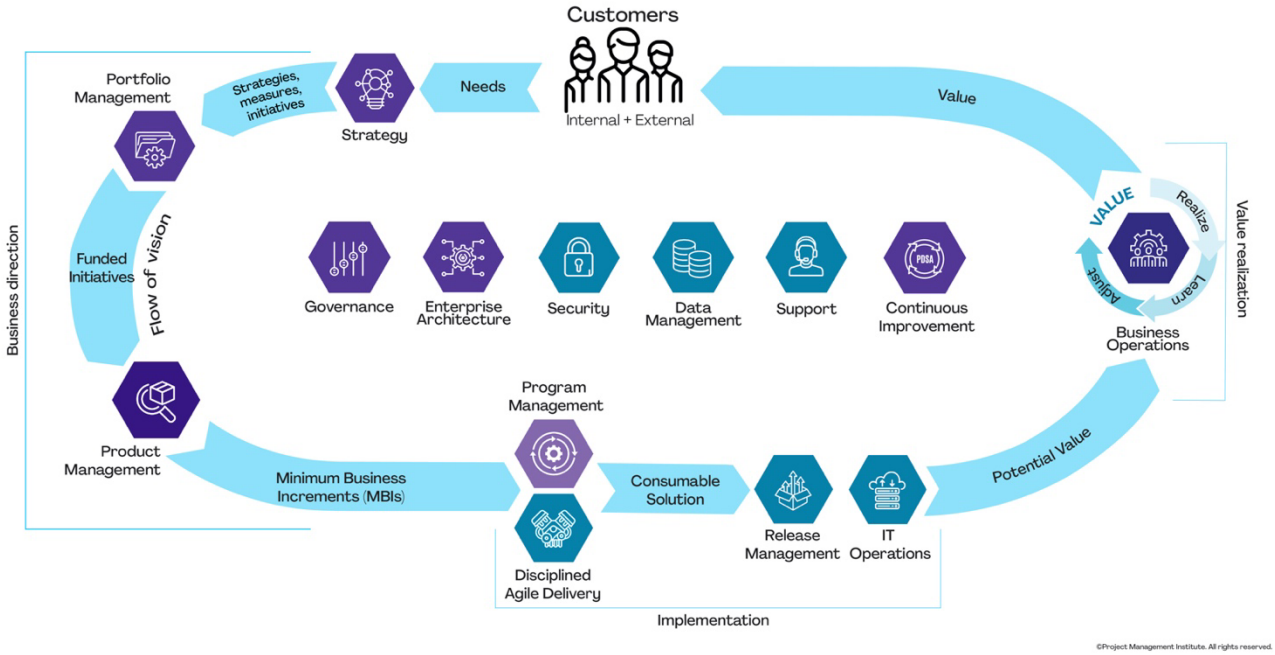

Figura 37 - DA Value Stream

Il Value Stream si concentra sullo sviluppo degli *MBI* (*Minimum Business Increment*), ovvero la quantità minima di valore da *costruire*, *distribuire* ed *utilizzare* in modo che l'iniziativa abbia effettivamente senso dal punto di vista aziendale.

Guardando al lato più operativo di gestione e sviluppo di un progetto, nel Blade *Disciplined Agile Delivery* (DAD) vengono proposti i lifecycle presentanti precedentemente e caratterizzati da tre fasi di avanzamento ideali:

- **Inception**, la fase di inizializzazione del progetto, in cui gli elementi fondamentali per la sua riuscita prendono corpo e si valutano aspetti come il *Rischio* e la *Sostenibilità*;
- **Construction**, la fase di realizzazione, ovvero dove la soluzione viene concretamente implementata;
- **Transition**, la fase che si occupa degli aspetti di *Deployment*.

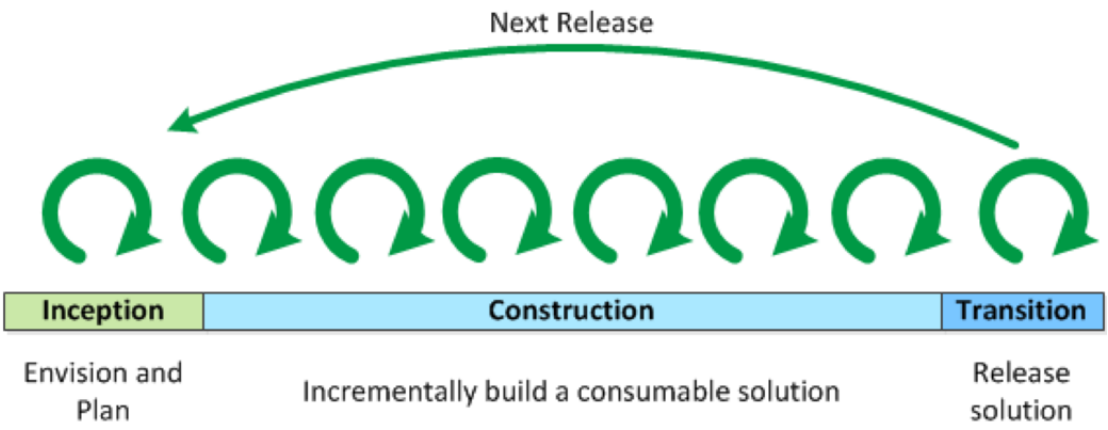

Figura 38 - Fasi di DAD

Se l'utilizzo di fasi può sembrare qualcosa di "strano" all'interno del contesto Agile, bisogna evidenziare come esse non siano da intendersi come *stage-and-gate*, bensì come "*contenitori di scopo*" che non limitano alcuna review delle decisioni prese.

Il loro obiettivo è, quindi, quello di focalizzarsi, nei diversi momenti, su specifici *process goal* che tipicamente sono richiesti nello sviluppo di un prodotto (progetto), goal che sono esplicitati dagli annessi *process goal diagram*.

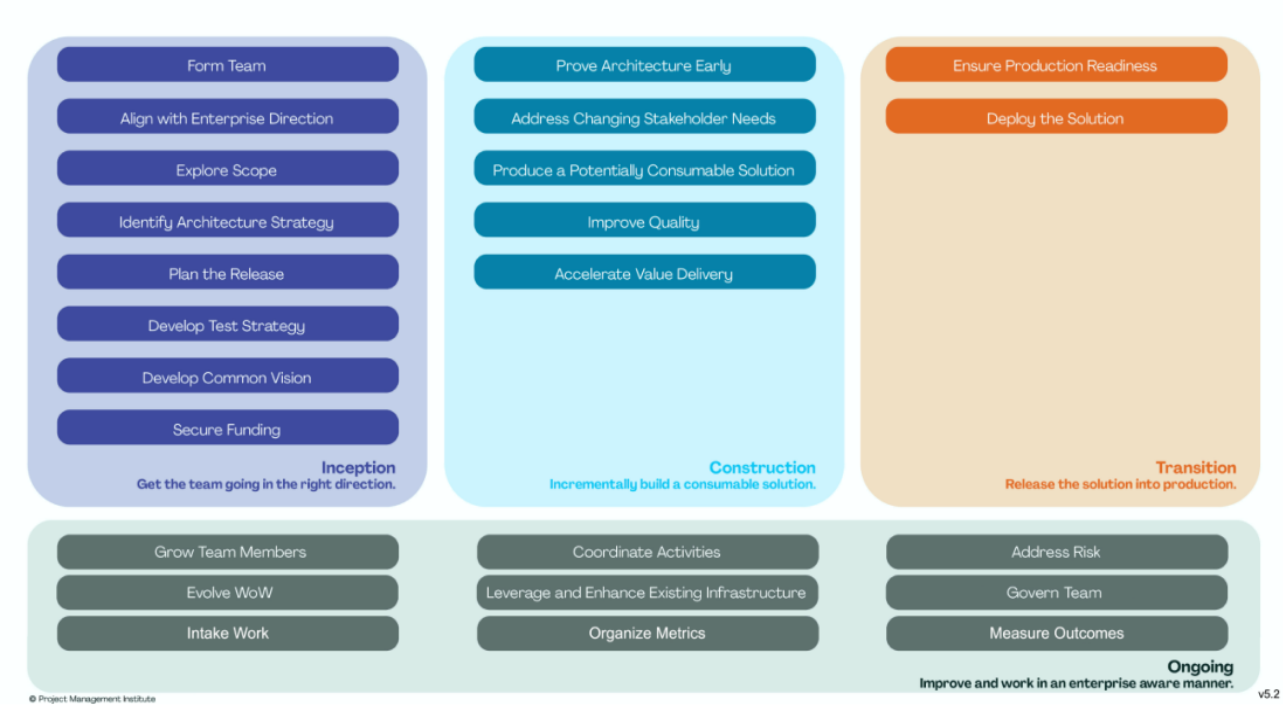

Figura 39 - The process goals of Disciplined Agile Delivery (DAD)

Nella figura precedente sono anche visibili dei goal raggruppati in una "virtual phase" chiamata *Ongoing*: si tratta di obiettivi trasversali che accompagnano il team nell'intera realizzazione del prodotto.

6.2 PMI PMBOK

A partire dal **PMBOK 7**, il body of knowledge più apprezzato del project management ha intrapreso una evoluzione in forte discontinuità con il passato.

Il cambiamento più evidente è la nuova impostazione basata su **principi** piuttosto che su **processi**, cosa che da vita ad una nuova filosofia di base: gli standard di gestione del progetto devono concentrarsi sugli outcome e quindi sulla generazione di valore tangibile per il cliente.

La logica operativa che distingue questa nuova edizione è quella del: "**Begin with the End in Mind**"[12], che evidenzia come sia sempre necessario chiedersi il perché si voglia realizzare qualcosa, e non sforzarsi esclusivamente sulla gestione degli aspetti operativi e sulla generazione di output che poi potrebbero non generare vero valore.

I cambiamenti apportanti rispetto alle edizioni precedenti sono davvero tanti e meritano gli opportuni approfondimenti che esulano da questa breve descrizione. Volendo fare una rapida sintesi si riscontrano però alcuni elementi chiave:

	PMBOK ® Guide Sixth Edition	PMBOK ® Guide Seventh Edition
Approccio generale	• Prescrittivo, non descrittivo • Enfasi sul come, non sul cosa o perché	• Principi per guidare forma mentis, azioni e comportamenti riflessi su body of knowledge per delivery di progetto, agile, lean, struttura centrata sul cliente, ecc…
Basi per la progettazione	• Processi specifici convertono input e output utilizzando tool e tecniche • Focalizzazione sul processo e orientamento più guidato dalla conformità	• Domini di aree di attività interdipendenti e correlate, sia con risultati di performance che con una overview di strumenti, tecniche, artefatti e framework più usati.
Ambiente di progetto	• Ambiente di progetto, interno ed esterno	• Ambiente di progetto interno ed esterno
Target di riferimento	• Principalmente project manager	• Chiunque sia coinvolto in un progetto, con un'attenzione specifica ai membri e ai ruoli del team, inclusi il capo progetto, lo sponsor e il product owner
Livelli di cambiamento	• Revisione incrementale sulla base delle edizioni precedenti	• Revisione basata sui principi per riflettere l'intero scenario di consegna del valore

[12] Stephen Covey - 1989

Guida alla personalizzazione	• Riferimenti all'adattamento (tailoring), ma non una guida specifica di riferimento	• Guida specifica all'adattamento

Il focus passa, quindi, in modo esplicito dagli *output* (il cosa) agli *outcome* (il perché) di progetto, rendendo di fatto i *processi* (il come) degli strumenti per raggiungere il fine ultimo di creare valore per gli stakeholder e non considerando loro stessi il fine del progetto.

Inoltre, i processi non sono più parte del nucleo del PMBOK, ma sono invece parte degli *Standards + Platform PMI*, permettendo di sviluppare in modo organico la dualità tra approccio predittivo ed approccio adattativo, grazie al focus sui metodi e approcci di gestione dei progetti agili, che negli anni sono stati adottati da molte organizzazioni nella gestione dei progetti.

7 Conclusione

Come speriamo sia emerso dalla lettura del paper, l'obiettivo del suo contenuto è quello di presentare un percorso strutturato, ed una serie di strumenti operativi, per scoprire come oggi il Project Management sia capace di accompagnare le sfide più complesse in una logica adattiva.

Se si è interessati nell'approfondimento e nell'applicazione concreta, è possibile contattarci all'indirizzo e-mail info@agileconstellation.info o attraverso i nostri canali social che trovate riportati sul sito ufficiale.

AgileBIM è distribuito con Licenza Creative Commons Attribuzione - Non commerciale - Non opere derivate 4.0 Internazionale.

Gli autori non possono essere ritenuti responsabili in alcun modo dell'utilizzo di quanto riportato in questo documento, in quelli annessi e nei canali digitali relativi. L'utilizzatore ha la totale responsabilità del proprio operato e libera i suddetti da ogni tipologia di incombenza diretta e indiretta.

Basato sul lavoro disponibile su info@agileconstellation.info

Permessi ulteriori rispetto alle finalità della presente licenza possono essere disponibili presso AgileConstellation.info