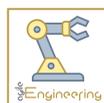


Engineering meets Agile

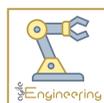
PAPER

Rev. 1.5.0 – 12 Marzo 2025



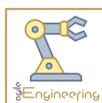
AgileEngineering

an AgileConstellation Star



Sommario

1. Introduzione	4
2. AgileEngineering	5
2.1 AgileEngineering: an AgileConstellation Star	5
2.2 AgileEngineering Fast Prototyping	7
2.3 Le fasi di AgileEngineering	8
3. La fase di Inception	10
3.1 Set Based Design.....	10
3.2 Intentional Architecture	12
3.3 Metriche per la fase di Inception	12
4. La fase di Engineering	13
4.1 L'organizzazione del lavoro.....	14
4.2 I Team	16
4.3 Qualità e metriche	18
4.4 Metriche per la fase di Engineering.....	19
5. La fase di Workout.....	21
5.1 La Produzione Manifatturiera	23
5.2 Metriche per la fase di Workout	23
6. Conclusioni	25



1. Introduzione

Il mondo dell'**Ingegneria dei Sistemi** è continuamente alla ricerca di nuovi modelli operativi che consentano di innovare rapidamente i propri prodotti, efficientando al contempo la filiera produttiva.

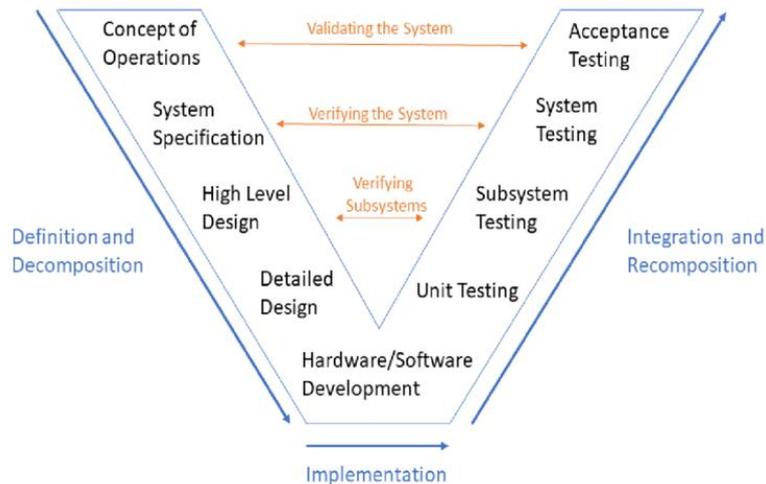
Tipicamente, in esso, si adotta un lifecycle lineare che segue degli step sequenziali ben definiti e rappresentato nel cosiddetto *modello a V* caratterizzato da 2 flussi:

1. **Flusso di Specifica**, che si esplica nelle attività a sinistra
2. **Flusso di Prova**, che si sviluppa lungo la direttrice destra

I due flussi sono collegati dall'**implementazione**, dove gli artefatti vengono realizzati.

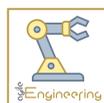
Questo modello è, però, scarsamente flessibile, con una netta divisione tra progettazione/pianificazione ed implementazione, rendendo costose le modifiche e l'adattamento alle esigenze emergenti.

Resta quindi intrinseca la fragilità dovuta al dettaglio dei requisiti, che si scontra con la realtà dove i clienti riescono a capire ciò che realmente vogliono solo nelle fasi più avanzate di realizzazione del prodotto, proprio dove i costi del cambiamento diventano spesso proibitivi. Inoltre, i progetti sono spesso di lunga durata (più anni) e impattano su molti domini differenti, il che enfatizza fortemente l'importanza e la complessità della governance generale e dell'integrazione dei vari elementi costituenti afferenti.



Spingendoci verso l'Agilità, è chiaro che i modelli tradizionali legati al mondo digitale devono essere adeguatamente adattati, concretizzandosi nel metaforico azzeramento dell'*angolo della V*: il *flusso di specifica* e il *flusso di test* tendono a sovrapporsi completamente, sposando al loro interno le attività di sviluppo, riducendo il ricorso a team esterni.

Ciò permette di velocizzare il **sistema dei feedback**, in particolare quello relativo ai clienti, in modo da catturare gli immancabili cambiamenti da apportare quando ancora il prodotto è in una fase "grezza", dove i costi annessi sono meno dirimpenti.



2. AgileEngineering

AgileEngineering supporta operativamente questi aspetti, stimolando i team di ingegneria nell'adozione di una logica *"fail fast"* (o *succeeding fast*), orientando lo sviluppo in relazione ad obiettivi specifici che si sviluppano su elementi validabili di breve, medio e lungo termine.

Essendo i prodotti dell'ingegneria di sistema dei "manufatti tangibili", AgileEngineering suggerisce di dividere le fasi operative di sviluppo in tre momenti specifici: *Inception*, *Engineering* e *Workout*.

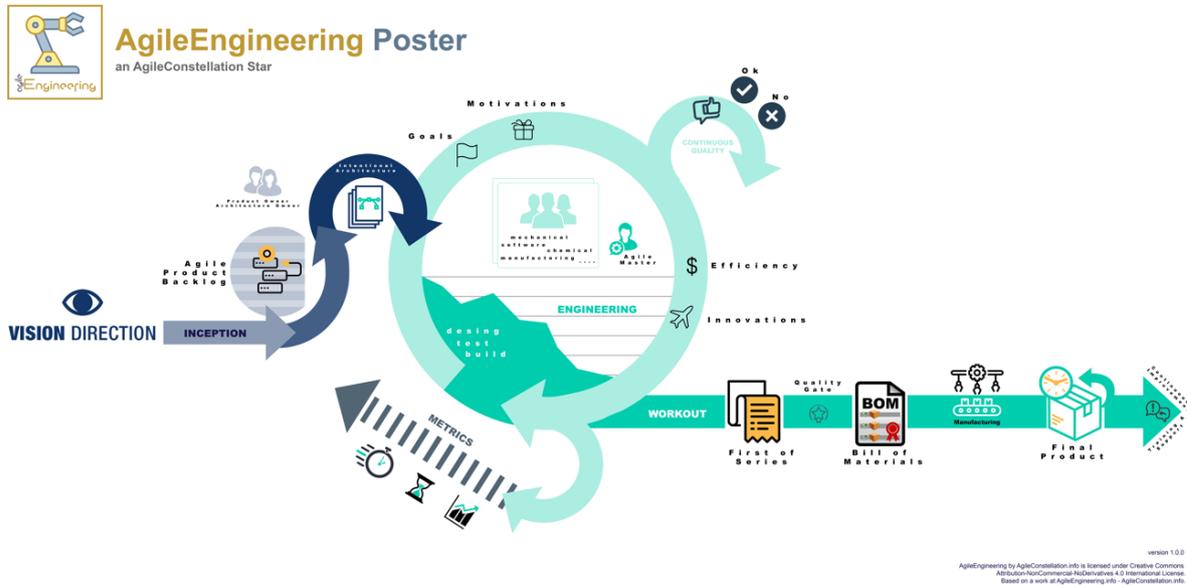
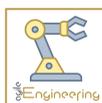


Figura 1 - AgileEngineering Poster

2.1 AgileEngineering: an AgileConstellation Star

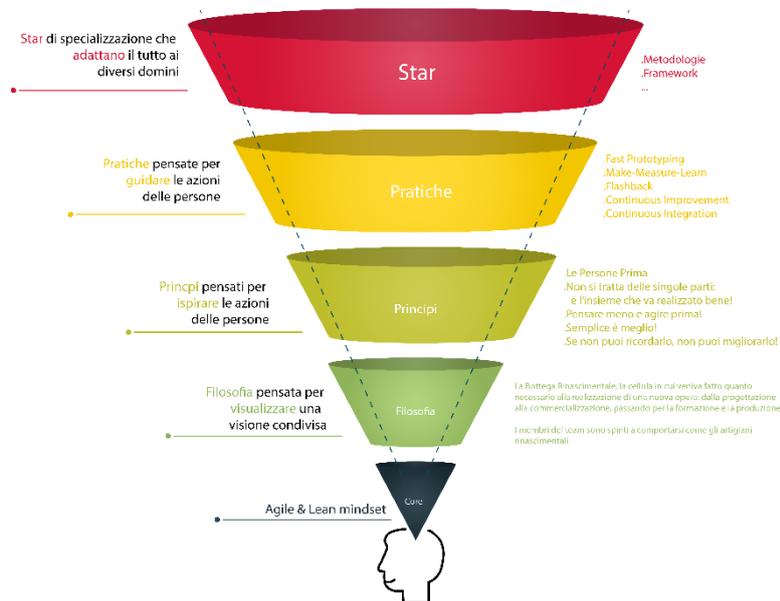
AgileEngineering è costituzionalmente basato sul mindset Agile e Lean, vantando la declinazione sviluppata in ambito del progetto AgileConstellation che guarda ai domini diversi da quelli software e digital in generale.





THE AGILECONSTELLATION FUNNEL DISCOVER THE FOUNDATION

AgileConstellation.info



Version 2.5
The AgileConstellation Funnel & AgileConstellation by AgileConstellation.info is licensed under Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License. Based on a work at www.agileconstellation.info.

Figura 2 - AgileConstellation funnel

AgileEngineering eredita quindi quello che è il mindset, la filosofia, i principi (core) e le pratiche (core) di AgileConstellation, declinando ed estendendo, in particolare, questi ultimi due aspetti nel dominio specifico dell'IoT.

Abbiamo 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
 - *Make-Measure-Learn*, sperimentare rapidamente le diverse ipotesi e le diverse assunzioni



- *Flashback*, azione di allineamento rapido in cui è l'osservatore ad ingaggiarsi nell'osservare l'elemento in lavorazione
- *Continuous Improvement*, migliorare costantemente ogni aspetto
- *Continuous Integration*, integrare costantemente le differenti anime della soluzione

2.2 AgileEngineering Fast Prototyping

In accordo con l'approccio modulare dell'AgileConstellation Manifesto, la *Star AgileEngineering*¹ aggiunge 6 nuove bubble di dominio alla pratica di Fast Prototyping per validare la sostenibilità della soluzione:

- **Security**: incentrato sulla verifica delle ipotesi relative agli aspetti di security che caratterizzano l'intera soluzione e che ne influenzano in modo diretto lo sviluppo.
- **Energy**: incentrato sulla verifica delle ipotesi relative agli aspetti energetici in funzione delle esigenze di continuità operativa degli smart device.
- **Hardware**: incentrato sulla validazione delle ipotesi hardware tramite uno o più *Evaluation Kit* (EVK). La selezione dell'EVK più idoneo passa, in primis, dalla scelta della CPU/MCU, dopodiché si inizia a "costruire" il prototipo lavorando sugli altri componenti (es: RAM, USB, ecc.).
- **Code**: incentrato sulla prototipazione del firmware dei dispositivi e su quella dei servizi a supporto per l'acquisizione dei dati/eventi portanti della soluzione. In questa fase l'utilizzo di framework e di IDE di codifica veloce è fondamentale per abbattere i tempi relativi.
- **Data**: incentrato sugli aspetti legati alla raccolta, alla pulitura ed alla gestione dei *Raw Data* provenienti dai device, implementando gli aspetti di trasmissione e serializzazione degli stessi, con la scelta degli opportuni protocolli e dei formati più adeguati. In particolare, è fondamentale dedicare particolare attenzione alla stima dei volumi dei dati ed alle relative metodologie di analisi, in modo da approcciare alla gestione delle informazioni in chiave *Polyglot Persistence*, fondamentale per garantire la possibilità di elaborare velocemente grandi moli di informazioni.
- **Cloud**: incentrato sugli aspetti Cloud della soluzione, intesa come piattaforma di gestione dei dati, degli eventi e delle action portanti.

¹ www.agileconstellation.org

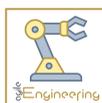




Figura 3 - Le nuove 6 bubble aggiunte dal dominio AgileEngineering

2.3 Le fasi di AgileEngineering

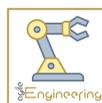
Nella fase di *Inception* viene realizzato quanto necessario per validare il sistema che ci si appresta a realizzare, focalizzandosi sullo sviluppo di quei componenti abilitanti che ne costituiranno l'infrastruttura portante. Il tutto si basa su una "discovery", spesso caratterizzata da un approccio Lean-based, che consente confronti rapidi con gli stakeholder. Si adottano, quindi, metodi di *rapid prototyping* e *digital twin*, sfruttando tecnologie emergenti (come, ad esempio, Stampa 3D e Realtà Mixata) in modo da ridurre i costi e consentire di sviluppare rapidamente nuovi prototipi con cui ingaggiare gli utenti e raccogliere feedback.

In questo ambito il ricorso all'*MVP* (Minimum Viable Change), inteso nella sua corretta accezione di *strumento per l'apprendimento*, è fondamentale e supporta il tutto in modo concreto.

Quando si è raggiunta una condizione di relativa stabilità, si procede all'*Engineering* delle parti più malleabili (ovvero le componenti più facilmente evolvibili), apportando gli opportuni sviluppi di consolidamento su quelle meno duttili (ad esempio, la componentistica standard acquisito dal mercato, *buy-in*).

Infine, la fase di *Workout* si occupa di gestire la produzione industriale (di serie) del tutto e di governare gli aspetti di delivery, nonché quelli di esercizio.

L'operatività nelle diverse fasi è affidata a **team multidisciplinari**, ovvero team composti da un mix di esperti nelle discipline coinvolte (elettronica, meccanica, chimica, software, ecc), che portano avanti le diverse attività necessarie per implementare le capability che caratterizzano il prodotto.



La sfida di **AgileEngineering** è quella di utilizzare tutte le tecniche note del mondo Agile e Lean, per favorire la comunicazione nei team e tra i team, in cui, come detto, ci sono esperti di domini diversi spesso abituati a lavorare in relazione ai propri skill più che in funzione di un obiettivo di prodotto che guardi al cliente finale. A tal fine, per favorire la creazione di una “lingua comune”, è fondamentale il ricorso agli strumenti di *Visual Management*, come task board o, nei casi più evoluti, un vero e proprio approccio Kanban-like che consenta di massimizzare sia la trasparenza sullo stato delle attività, sia l’individuazione dei colli di bottiglia su cui intervenire il prima possibile.

Grazie ad AgileEngineering è possibile ridurre i rischi di diventare non competitivi, andando a stimolare un approccio adattativo in grado di ridurre il lead time e generare soluzioni innovative mantenendo fermi gli standard di qualità di riferimento. Infatti, non bisogna sottovalutare l’impatto che le normative di riferimento hanno sul processo generale, essendo spesso vincolanti e non derogabili per la commercializzazione e l’uso del prodotto stesso.

Tra i principali vantaggi che si osservano dall’adozione di AgileEngineering si annoverano:

- *Riduzione del Lead Time, del Time to Market e del Time to Value*, grazie all’introduzione di strumenti di fast prototyping e l’utilizzo organico dell’MVP
- *Miglioramento dell’efficienza dei processi o workflow*, sfruttando soluzioni ispirate all’Agile e a Lean, unitamente alla loro visualizzazione esplicita.
- *Riduzione del gap comunicativo tra i team di ingegneria e gli stakeholder*, grazie allo sviluppo di feedback rapidi e momenti di allineamento costanti.
- *Miglioramento della qualità*, sia “reale” che “percepita”.

Alcune declinazioni specifiche dell’ingegneria di sistema riguardano, ad esempio, l’Automotive, l’Aerospace, le aziende Missilistiche, le aziende dei Medical Device e così via.

Ognuno di questi domini può trarre vantaggi evidenti da quanto evidenziato fin ora, declinando ulteriormente le specificità nel proprio contesto e identificando gli strumenti espliciti che meglio supportano la visione complessiva e l’operatività corrente.



3. La fase di Inception

Uno dei dubbi che immediatamente sorge quando si parla di Agile applicato nell'ambito dell'Ingegneria di Sistema è come sia possibile sviluppare in modo iterativo un sistema complesso, spesso visto come un tutto o niente, composto da elementi fisici la cui duttilità è tutt'altro che un elemento da sottovalutare.

Spesso, l'approccio seguito, è quello di concentrarsi sulla realizzazione di prototipi-successivi (*Set Based Design*), in cui, di volta in volta, si raffinano aspetti specifici dei diversi componenti per poi arrivare al "Primo di Serie" e alla specifica definizione della *Bill of Materials*.

Per districarsi con questo aspetto, **AgileEngineering** individua una specifica fase di **Inception**, in cui vengono confutati gli aspetti primari della soluzione e realizzata la **dorsale di sviluppo** necessaria a dare lo start operativo di dettaglio.

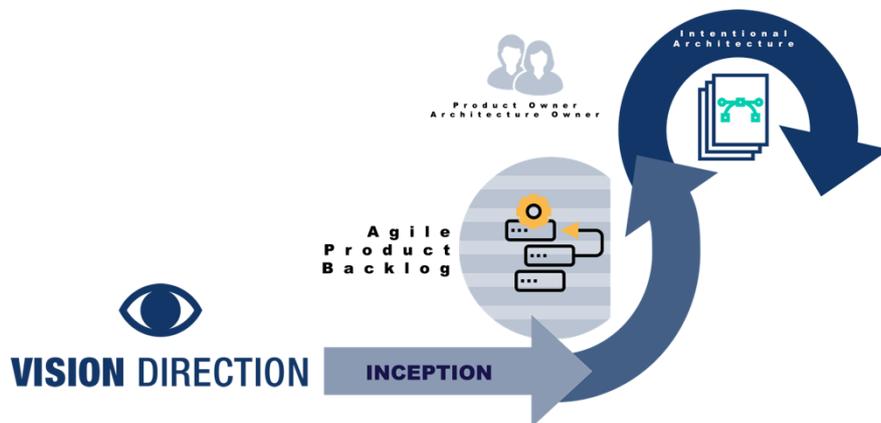


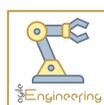
Figura 4 - AgileEngineering Inception phase

In questa fase, un approccio *Lean based* risulta fondamentale, data la variabilità delle azioni da realizzare e la sperimentazione che generalmente caratterizza in questo frangente il prodotto in modo predominante.

3.1 Set Based Design

Nell'azione di identificazione della soluzione si ricorre all'approccio denominato **Set-based Design (SBD)**, noto anche come il "secondo paradosso di Toyota", che permette di considerare un'ampia gamma di decisioni possibili acquisendo, però, un vantaggio tale da ridurre poi gli interventi successivi e quindi il ciclo di sviluppo.

L'SBD si contrappone al tradizionale processo lineare noto come **point-based design** che parte da un concetto iniziale e procede con dettagli sempre maggiori dello stesso. Se, a prima vista, questo metodo sembra molto efficiente, poiché si lavora su un unico concetto dall'inizio alla fine, non si tiene però conto che il mondo non è lineare e le sfide sono spesso più complesse di quello che sembra. Con il point-based design, se durante il processo di progettazione emergono nuovi vincoli



o nuovi requisiti da parte del cliente, si è costretti a ricominciare tutto da capo, a volte tornando addirittura al punto di partenza.

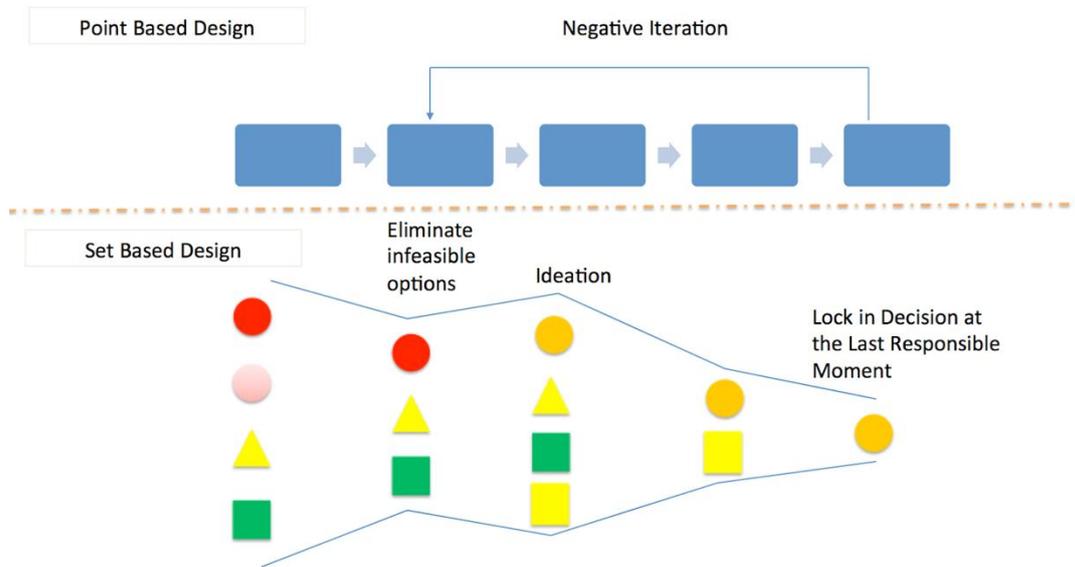


Figura 5 - Point Based vs Set Based Design

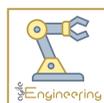
Dualmente, piuttosto che selezionare un'unica opzione promettente e lavorare esclusivamente su quella, il *Set-based Design* esplora un'ampia gamma di opzioni possibili: i *set di soluzioni* vengono progressivamente ridotti fino a convergere verso una soluzione finale. Durante il processo di progettazione, alcune opzioni vengono eliminate a causa di vincoli rigidi, problemi di fattibilità o mancata idoneità. Inoltre, l'ideazione può essere utilizzata per generare nuove opzioni: solo al **Last Responsible Moment** (LRM, l'ultimo momento responsabile), verrà presa una decisione e selezionata una delle opzioni rimaste.

Avviando il processo con un ampio set di opzioni e scartando progressivamente le soluzioni più deboli, è possibile valutare un maggior numero di possibilità e trovare soluzioni migliori.

In caso di vincoli imprevisti o nuove esigenze, si riduce il rischio di dover ricominciare da capo, poiché una o più delle opzioni rimanenti potrebbero già soddisfare i nuovi requisiti. Se necessario, si torna indietro solo di uno o due passaggi, invece di ripartire da zero.

I principi cardini del Set-based Design sono:

- *Definire le regioni fattibili*
- *Esplorare i compromessi progettando più alternative*
- *Comunicare i set di possibilità*
- *Cercare intersezioni tra set fattibili*
- *Imporre il minor numero possibile di vincoli*
- *Ridurre progressivamente i set aumentando il livello di dettaglio*
- *Rimanere all'interno dei set una volta prese le decisioni*



- *Prendere decisioni e selezionare l'opzione finale al Last Responsible Moment.*

In pratica è come se i principi Lean, generalmente pensati per l'ottimizzazione dell'as-is e la riduzione degli sprechi, subiscano un effetto di **shift-left**, agganciandosi allo sviluppo di nuovi prodotti e trovando applicazione nel momento in cui le decisioni possono avere la maggiore influenza sia sul prodotto che sul funzionamento.

La soluzione individuata, insieme agli asset realizzati e al piano desiderato, costituirà il passe-partout per la fase di **Construction** in cui il tutto verrà sviluppato nel dettaglio e rifinito per ottenere il prodotto definitivo in grado di soddisfare l'utente e che sia, chiaramente, industrialmente sostenibile.

L'aspetto centrale della fase di Inception è la capacità di generare know-how tra i membri del team e tra il team ed il cliente, spingendo all'innovazione sostenibile continua.

3.2 Intentional Architecture

Un aspetto chiave della fase di Inception è l'**Intentional Architecture**, un approccio che bilancia la flessibilità delle scelte architetture con la necessità di fornire una struttura solida su cui costruire il prodotto.

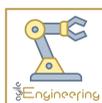
A differenza dell'architettura emergente, che evolve esclusivamente basandosi sugli sviluppi iterativi, l'Intentional Architecture prevede:

- *Definizione delle fondamenta del sistema:* Identificare i moduli e le componenti chiave che devono essere stabili per garantire scalabilità e sostenibilità.
- *Principi guida per le decisioni architetture:* Creare un quadro di riferimento per le scelte tecnologiche e di design.
- *Adattabilità:* Strutturare l'architettura in modo che possa evolversi con il tempo senza richiedere una riprogettazione totale.

3.3 Metriche per la fase di Inception

Le metriche chiave per valutare l'efficacia della fase di Inception includono:

- *Time to Prototype (TTP):* tempo necessario per sviluppare un primo prototipo funzionante.
- *Stakeholder Feedback Cycle (SFC):* tempo medio necessario per raccogliere e integrare i feedback degli stakeholder.
- *Experiment Success Rate (ESR):* percentuale di esperimenti di prototipazione che forniscono risultati validi per le iterazioni successive.
- *Change Cost Index (CCI):* misura del costo medio delle modifiche implementate durante questa fase rispetto a quelle effettuate più avanti nello sviluppo.



4. La fase di Engineering

Nella fase di **Engineering** il prodotto viene implementato nei diversi aspetti *funzionali customer side* seguendo una logica più tipicamente Agile.

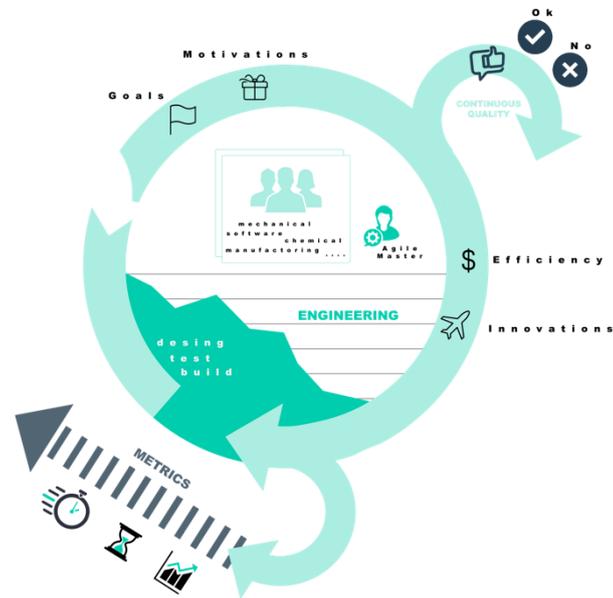


Figura 6 - AgileEngineering - Engineering phase

Questa fase viene guidata da un “**albero funzionale**”, elemento di output/outcome della precedente fase di *Inception*, in cui sono state definiti, a livello medio-alto, i diversi componenti caratterizzanti il sistema.

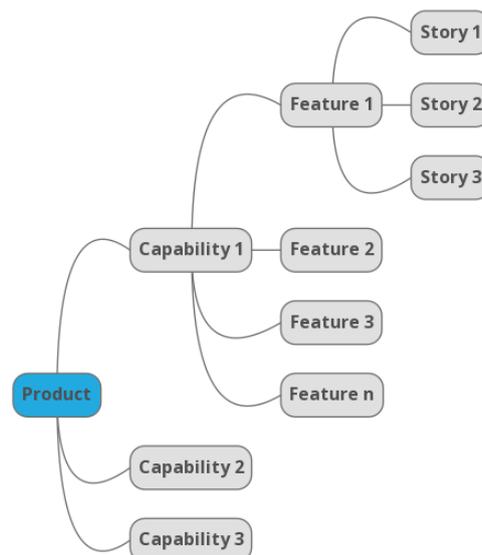
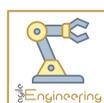


Figura 7 - Albero Funzionale



L'utilizzo dell'albero funzionale aiuta a focalizzarsi sulle funzionalità da realizzare in relazione alle necessità del cliente, favorendo l'organizzazione in team cross-funzionali. Si tratta di un aspetto molto delicato, soprattutto nell'ambito dell'ingegneria di sistema che, normalmente, si basa su una forte settorializzazione delle competenze e suddivisione delle attività in "working-package" affini (leggasi WBS).

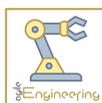
L'albero funzionale è chiaramente accompagnato da una *vista architeturale* che aiuta a capire i componenti da utilizzare per l'implementazione delle funzionalità richieste, permettendo di fare una prima valutazione di fattibilità e sostenibilità, alla base anche delle attività di sviluppo della **dorsale**, già presentate nella discussione inerente alla fase di Inception.

4.1 L'organizzazione del lavoro

La composizione dei Team è un aspetto particolarmente delicato da curare: generalmente, si hanno più team concentrati su parti funzionali del sistema che però non hanno un contatto continuativo con il cliente finale, anche perché per quest'ultimo l'oggetto minimo da poter valutare è una **Capability**, che rappresenta un sotto-insieme funzionale (ma anche a livello di componenti) significativo. Tale Capability ha, tipicamente, un tempo di sviluppo ampio, che può superare anche l'anno, per cui bisogna trovare il modo di spostare il focus di review e allineamento quanto meno sulle *Feature* (che hanno ragionevolmente una declinazione temporale di qualche mese).

Anche in questo caso, comunque, si hanno intervalli di tempo considerevolmente più ampi di quanto generalmente previsti nel mondo Agile (2, 3 settimane di media), cosa che porta a riconsiderare il ruolo di Product Owner di ogni team più in un'ottica di **Engineer Owner** che aiuta il team nel dettaglio delle attività e nel coordinamento con gli altri team o aree di lavoro coinvolte. Viene quindi a configurarsi una "**product ownership board**" composta dagli Engineer Owner, generalmente uno per ogni team che, nell'insieme, si coordinano con il Product Owner (di prodotto) che li supporta nelle relazioni con il cliente e nella revisione dell'**Agile Portfolio** tramite il quale vengono definiti gli obiettivi nei diversi riferimenti temporali (*Annuale, Quarter, Mensile, Iterazione, ecc*), nonché i backlog operativi.

In particolare, si ricorre alla *Flight Level Architecture*, in cui si passa dal livello Strategico a quello Operativo (e viceversa) per una completa visione di come sta progredendo lo sviluppo del prodotto.



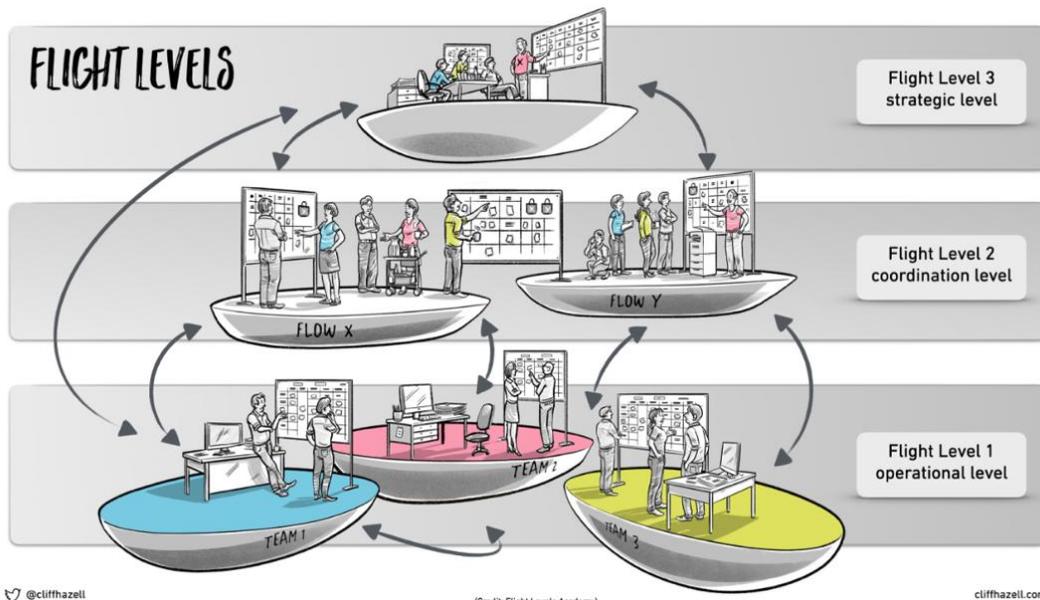
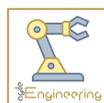
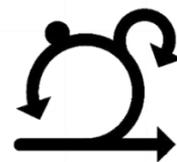
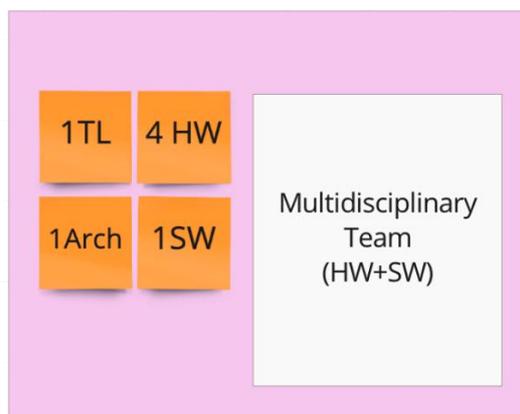


Figura 8 - The Flight Levels Architecture

Ogni capability è riconducibile al concetto di **Minimum Business Increment (MBI)**, rappresentando un **elemento atomico** che ha significato dal punto di vista della sostenibilità del business, ovvero sia per il cliente ma anche per gli obiettivi dell'organizzazione.

Su di esso si concentra il vero focus di sviluppo in chiave cross-disciplinare (software, elettronico, meccanico, chimico, ecc), grazie ad una sua progressiva scomposizione che dà vita al *backlog operativo* in cui si esplicano gli interventi diretti da realizzare sul prodotto.

Un team che si ispira ad AgileEngineering è, quindi, un team composto da esperti dei vari settori afferenti che si coordinano tra loro per completare una capability nelle sue diverse dimensioni, abbattendo notevolmente la complessità di integrazione e di test che si riscontra quando lo sviluppo è invece centrato su componenti di dominio realizzati in modo indipendenti tra loro.



4.2 I Team

In linea con la filosofia Agile, è fondamentale distinguere attentamente tra: *risorse immateriali* e *risorse umane (Persone)*, in modo da non perdere mai di vista l'attenzione allo sviluppo di un ambiente collaborativo in grado di migliorarsi costantemente.

Resta però un fatto evidente: non sempre avere un team con tutte le competenze necessarie ad affrontare lo sviluppo di un Prodotto è efficiente da un punto di vista di ottimizzazione della loro allocazione. Ciò è, inoltre, valido anche per la stessa *motivazione delle persone*: una persona che da un contributo marginale durante una iterazione e deve “trovarsi” le attività da fare, non sviluppa un valido senso di appartenenza perché non riesce a trovare la propria vocazione operativa specifica.

Si evidenziano così 4 fattori specifici, qui definiti **GEMI** da tenere in conto nella creazione dei team, in modo da stimolare adeguatamente le persone e creare un “ambiente” idoneo al raggiungimento di obiettivi sempre più d'avanguardia:

- **Goals**, ovvero la capacità di avere degli obiettivi condivisi, chiari e ben identificati
- **Efficiency**, tutte le attività devono sempre tenere in conto l'ottimizzazione delle risorse disponibili in relazione ai vincoli di contesto
- **Motivations**, ovvero la capacità (qui i manager giocano un ruolo chiave) di motivare adeguatamente le Persone, creando un opportuno percorso di crescita “sociale” e professionale
- **Innovations**, lavorare sempre su iniziative sfidanti che stimolano un continuo riallineamento dello *State of Flow*, ovvero la condizione di maggior stimolo per le persone.



Figura 9 - GEMI factors



I diversi specialisti sono organizzati il più possibile in *team multidisciplinari* focalizzati su obiettivi, anche se bisogna sottolineare uno specifico effetto. Se è vero, infatti, che questo favorisce molto la capacità di essere focalizzati su un obiettivo complessivo di valore (e non locale), è anche vero che quando le attività sono estremamente specialistiche (ad esempio relativi ad aspetti derivanti da normative impattanti sul prodotto) non sempre tale scelta si rileva vincente, in quanto non è possibile “replicare” le competenze relative per ogni team e, oggettivamente, non trova ragionevole sostenibilità, di costo e di ingaggio.

Si ha così la necessità di far convivere dei veri *cross-domain* team, composti ad esempio da esperti in: *meccanica, elettronica, software, chimica, ecc...* con un gruppo di supporto che supporti i diversi team e sia prontamente disponibile. È fondamentale evidenziare che non si sta parlando di creare una “funzione” organizzativa, ma una sorta di *loan team* comunque dedicato al progetto.

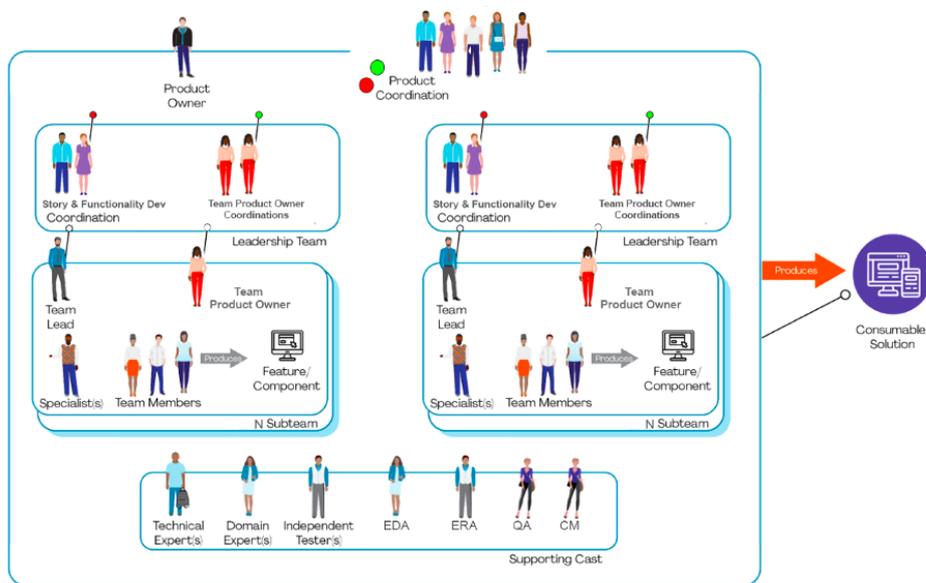


Figura 10 - Cross-domain teams and Loan team (by PMI Disciplined Agile)

I diversi cross-domain team si auto-organizzarono, quindi con un approccio fortemente bottom-up, con il supporto di un Product Owner (o *Engineer Owner*) che li aiuta a restare focalizzati sugli obiettivi generali e dirimere il più possibile le dipendenze tra le attività specifiche su cui si sono ingaggiati.

Si possono quindi avere due scenari di riferimento:

- Un *prodotto sviluppato da un unico team composto da tutti gli specialisti necessari*. Questo scenario, seppur semplificando le diverse problematiche di dipendenza e coordinamento, è il meno realistico viste le complessità affrontate dall'ingegneria di sistema e la necessità di avere tempi ragionevoli nello sviluppo dei nuovi prodotti.



- Un *prodotto sviluppato da più team in parallelo*. In questo caso, quello più comune, è fondamentale sviluppare una opportuna *leadership* di coordinamento per consentire ai diversi team di efficientare le proprie attività grazie ad una collaborazione efficace e snella.
- In entrambi gli scenari, resta valido il già citato *loan team* di supporto su argomenti specialistici non necessari nel quotidiano.

4.3 Qualità e metriche

La **qualità** è un elemento fondamentale che deve essere posta al centro della visione di prodotto, anche se la relativa declinazione dipende molto dal dominio specifico, dagli standard afferenti e dalle regolamentazioni annesse.

Nonostante ciò, è possibile definire un modello astratto (ispirato alla *Test Pyramid*²) che qui chiameremo *Quality Pyramid* e che può essere rappresentato come segue:

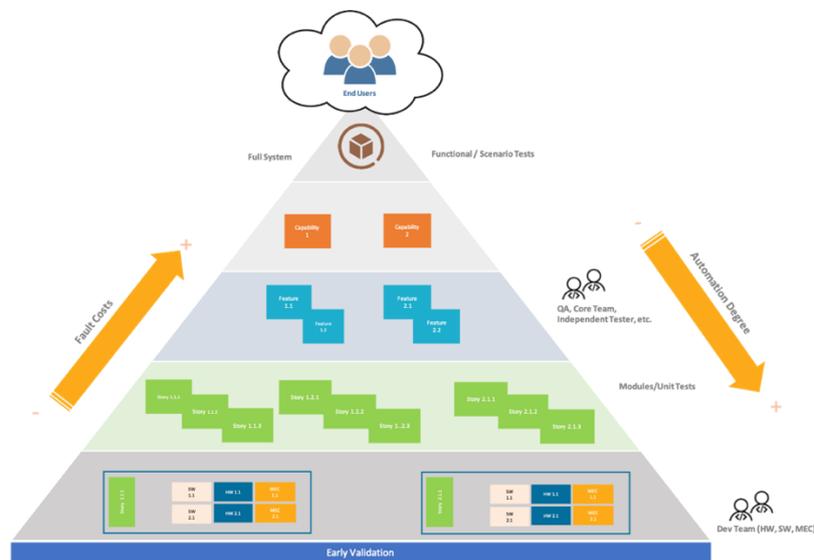
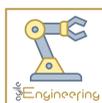


Figura 11 - Quality Pyramid

L'aspetto fondamentale è che la qualità passa attraverso una *serie di test e validazioni a più livelli*, cercando di estremizzare l'automazione nei livelli più bassi della piramide dove, normalmente, si realizzano i diversi componenti del prodotto e dove gli sviluppatori usano le pratiche e gli strumenti più adatti al dominio afferente.

È evidente che più si riesce a ottimizzare il testing ai livelli più bassi, meno problemi dovrebbero sorgere quando ci si sposta verso la punta della piramide, dove, generalmente, il focus è di tipo *end-to-end* (verso l'utilizzatore finale) ed eventuali problematiche hanno un costo di risoluzione molto più alto, alcune volte non assorbibile.

² <https://martinfowler.com/articles/practical-test-pyramid.html>



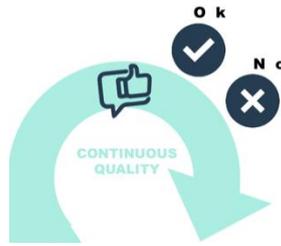


Figura 12 - Continuous Quality

In modo molto astratto, AgileEngineering considera la *Quality* come un “gate” abilitante (*Quality Gate*) che si basa su un approccio orientato alla *Continuous Quality*: non esiste il “ni” nella qualità, ovvero o si rispettano le relative attese, oppure il componente/modulo in analisi non è pronto per l’integrazione con gli altri o per la consegna al cliente.

L’approccio alla Continuous Quality indica l’attenzione ad un miglioramento continuo nelle pratiche e negli strumenti affini, evidenziando ancora una volta come l’evoluzione contingente, e pragmatica, sia alla base di una buona gestione di tutto quanto afferisce a un prodotto complesso, rispetto al quale la conoscenza si affina proprio durante la sua realizzazione.

Il miglioramento non può però essere lasciato (unicamente) all’intuito, ma deve basarsi su dei dati oggettivi, dei numeri, che possano evidenziare dei *fatti* e non dare adito ad *opinioni* che sono sempre affette dalla propria capacità e declinazione interpretativa.

Proprio per questo, AgileEngineering definisce una serie di metriche di base che aiutano a valutare oggettivamente la capacità di sviluppo e di reazione dei team di lavoro:



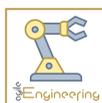
Figura 13 - Metrics

4.4 Metriche per la fase di Engineering

Il modello Quality Pyramid definisce i livelli di validazione del prodotto, enfatizzando la Continuous Quality.

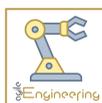
Le relative metriche sono:

- *Cycle Time (CT)*: tempo medio impiegato per completare una singola iterazione di sviluppo.
- *Lead Time (LT)*: tempo totale dall’inizio dello sviluppo fino alla consegna di una feature.
- Mean Time to Recovery (MTTR): tempo medio impiegato per risolvere un problema critico.
- *Deployment Frequency (DF)*: numero di rilasci effettuati in un determinato periodo di tempo, utile per valutare la capacità di iterazione del team.



- *Defect Escape Rate (DER)*: percentuale di difetti rilevati dopo il rilascio rispetto a quelli individuati in fase di sviluppo.
- *Code Churn Rate (CCR)*: quantità di codice modificato o riscritto entro un breve periodo dopo il primo commit, indicatore dell'efficacia della pianificazione tecnica.
- *Test Coverage (TC)*: percentuale di codice coperta da test automatici per garantire una maggiore affidabilità del software.
- *Team Velocity (TV)*: quantità di lavoro completata dal team in un'iterazione, spesso misurata in story points o task completati.
- *Failure Recovery Rate (FRR)*: tempo medio necessario per ripristinare il sistema in caso di fallimenti critici.

Queste metriche aiutano a mantenere un controllo efficace sulle performance del team di sviluppo e sulla qualità del prodotto.



5. La fase di Workout

Come più volte sottolineato, l'output primario sviluppato da un'azione di ingegneria di sistema è composto da diversi elementi afferenti a diverse discipline, molto spesso anche fortemente diverse tra loro.

Per questo la delivery (intesa come consegna del prodotto e quanto necessario per il suo corretto uso da parte dei clienti) è decisamente articolata e richiede l'utilizzo di specifiche tecniche di **validazione** e **preparazione**, nonché di **training** e **supporto**.

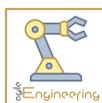
L'insieme di queste azioni, che portano il prodotto "nelle mani del cliente" viene definita in AgileEngineering come fase di **Workout**.



Figura 14 – AgileEngineering - Workout phase

Nella fase di Workout si identificano alcuni elementi fondamentali, che, nonostante vadano contestualizzati, sono di riferimento per una struttura gestione della delivery:

- *First of Series (Primo di Serie)*: il "prototipo" industriale del prodotto. In pratica si ha un prodotto industrializzato rappresentativo di quanto verrà successivamente realizzato in modo massivo (o comunque di quanto verrà consegnato al cliente) su cui poter effettuare tutti le valutazioni di pre-produzione.
- *Quality Gate*: un insieme di azioni, test, validazioni e verifiche effettuate sul primo di serie che porta allo sviluppo di tutti gli elementi definitivi e le relative specifiche per la manifattura.
- *Bill of Materials*: sono le specifiche finali, e quanto necessario, per la manifattura.
- *Manufacturing*: è il processo di produzione industriale del prodotto.
- *Final Product*: è il prodotto industriale finale.
- *Training e Support*: le azioni di training e supporto all'utilizzo del prodotto da parte dei clienti.
- *Product Continuous Improvements*: supportare il prodotto finale con continui miglioramenti ed aggiornamenti



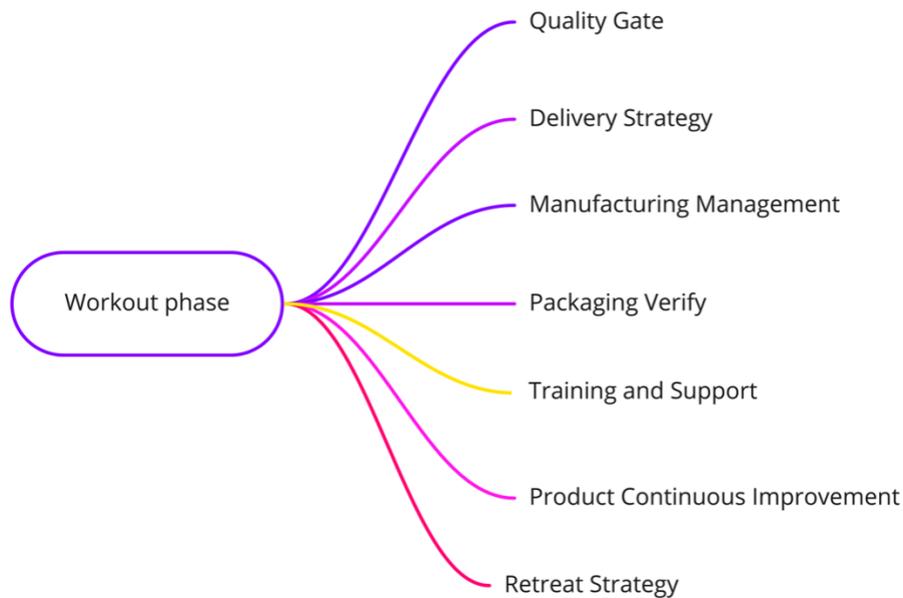


Figura 15 - Workout goals

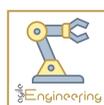
Come si vede, la fase di Workout è di per sé estremamente articolata e necessita del supporto di molte professionalità eterogenee.

In particolare, va sottolineato che la parte di manifattura richiede quasi sicuramente l'ingaggio di un partner esterno per la produzione del prodotto, cosa che sposta il focus su aspetti di gestione e relazione dei fornitori esterni. Normalmente il *primo di serie* serve per passare dagli strumenti di prototipazione rapida utilizzati nella fase di Engineering a elementi industriali, custom o di mercato, per capirne gli impatti e valutarne i costi. Qui la scelta di un fornitore di fiducia è fondamentale per far fronte alle immancabili problematiche tipiche della transizione e lavorare in ottica agile di feedback costanti.

Un aspetto particolarmente delicato è quello della *realizzazione fisica del prodotto* (supportato in particolare dalle pratiche annesse al goal di *Manufacturing Management*). In linea generale il prodotto finale sarà sicuramente basato su un mix tra *Market Components e Custom Components*, individuando sul mercato la maggior parte di componenti da poter riutilizzare tramite un loro adattamento (per ridurre i costi di produzione), mentre la parte restante verrà realizzata specificamente in funzione delle esigenze del prodotto.

In ogni caso l'aspetto della produzione manifatturiera diventa un elemento centrale della fase di Workout e come tale va adeguatamente gestito creando un ponte tra il team di progettazione e quello di produzione (interno o esterno che sia) in modo da risolvere le iniziali inevitabile problematiche di realizzazione.

Questo, in particolare, porterà al *Primo di Serie* (che potrebbe essere seguito anche da ulteriori prototipi se necessario) che, oltre ad avere lo scopo di valutare l'efficacia funzionale del prodotto,



aiuterà a settare i corretti processi produttivi e rivedere eventuali ipotesi realizzative per un efficientamento complessivo.

Tutto ciò trova casa nel Quality Gate goal che dovrà portare alla definizione puntuale di tutti gli aspetti validati, di prodotto e processo, creando la *Bill of Materials* per l'avvio della produzione.

5.1 La Produzione Manifatturiera

Sia che la produzione avvenga da parte di un produttore terzo, sia che, invece, se ne occupi un diverso team/dipartimento della stessa azienda, bisogna integrare la stessa nelle proprie attività di governance, per non incorrere in spiacevoli sorprese che vadano a minare quanto fatto fino a questo momento.

La relativa gestione operativa è, tipicamente affidata ad un (chief) *Project Manager* che ha il compito di instaurare un proficuo rapporto di *collaborazione* e *fiducia* tra le diverse realtà produttive, a partire dalla realizzazione del *Primo di Serie*, in cui, nonostante si lavori ancora su base prototipale, si inizia a condividere il know-how necessario alla successiva realizzazione industriale (che può essere anche massiva) del prodotto.

Normalmente l'azione di *Manufacturing* segue una logica più *Lean-oriented* (Lean manufacturing nello specifico) puntando sull'efficientamento della produzione e, quindi, sulla razionalizzazione dei costi annessi (costo marginale).

Una volta che il prodotto è operativo, ovvero installato ed in uso presso il cliente, il supporto allo stesso diventa bidirezionale, abilitando la logica del *Product Continuous Improvement*:

- *Aggiornamento predittivo*: avviene per azioni del produttore e può essere svolto anche in remoto (si pensi all'aggiornamento del firmware).
- *Aggiornamento correttivo*: avviene in seguito alla segnalazione di un malfunzionamento o un'anomalia segnalata dal cliente.

L'obiettivo dell'approccio presentato è quello di far sì che gli eventuali aggiornamenti siano focalizzati, nella quasi totalità, sugli elementi più *duttili* del sistema, evitando di agire sulle parti che comportano alti costi di intervento (si pensi ad esempio alla meccanica), a meno una revisione totale del prodotto.

Un'azione critica da non sottovalutare è quella di strutturare adeguatamente, e per tempo, il piano di *Training e di Supporto* diretto ai clienti per l'utilizzo del prodotto. Ciò è fondamentale per aumentare il livello di qualità percepita, che trasforma un prodotto tecnicamente di qualità in un prodotto di successo apprezzato dai clienti.

5.2 Metriche per la fase di Workout

Le metriche della fase di Workout sono:

- *First Pass Yield (FPY)*: percentuale di prodotti che superano la prima fase di produzione senza necessità di rilavorazioni.
- *Defect Density (DD)*: numero di difetti rilevati per unità di prodotto o lotto di produzione.



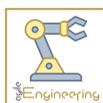
- *Manufacturing Cycle Time (MCT)*: tempo totale impiegato per completare la produzione di un'unità, dall'inizio alla fine.
- *Supplier Quality Index (SQI)*: qualità dei componenti forniti dai partner industriali, misurata in base alla conformità ai requisiti.
- *Production Downtime (PD)*: tempo totale di inattività delle linee di produzione dovuto a guasti, manutenzione o problemi tecnici.
- *Customer Return Rate (CRR)*: percentuale di prodotti restituiti dai clienti a causa di difetti o malfunzionamenti.
- *On-Time Delivery (OTD)*: percentuale di prodotti consegnati ai clienti entro la data stabilita.
- *Cost of Poor Quality (CoPQ)*: costo totale derivante da rilavorazioni, scarti e resi.
- *Satisfaction Index (SI)*: valutazione della soddisfazione dei clienti sulla base di feedback e indagini post-vendita.
- *Mean Time Between Failures (MTBF)*: tempo medio tra un guasto e l'altro per i prodotti consegnati.



6. Conclusioni

Come accennato nell'introduzione, questo paper ha il compito di raccontare gli aspetti focali dell'Agile applicato al mondo dell'Ingegneria di Sistema, evidenziandone gli aspetti di cardine relativamente alle diverse aree afferenti.

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.





AgileEngineering è distribuito con Licenza [Creative Commons Attribuzione - Non commerciale - Non opere derivate 4.0 Internazionale](https://creativecommons.org/licenses/by-nc-nd/4.0/).

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 [AgileConstellation.info](https://agileconstellation.info)

Permessi ulteriori rispetto alle finalità della presente licenza possono essere disponibili presso [AgileConstellation.info](https://agileconstellation.info)

