POLITECNICO DI TORINO

Corso di Laurea Magistrale
in Ingegneria della Produzione Industriale
e dell'Innovazione Tecnologica.

Tesi di Laurea Magistrale

La metodologia Agile per il Project Management

Relatore

prof. Guido Perboli

Candidato

Carla Antoci

Sommario

Indice figure

1. Introduzione

Alla base di questo studio vi è l'analisi della metodologia Agile e la sua diffusione a livello di organizzazione aziendale. In particolare, si pone l'attenzione sulle differenze e similitudini con altre tecniche di gestione: Scrum, Kanban e Business Agility. Concludendosi con un case study in cui si vedono applicati ad un'azienda tutti i principi studiati teoricamente.

Le motivazioni che mi hanno spinto ad approfondire tale tema hanno una duplice natura. L'interesse nei confronti dello sviluppo software è stato influenzato e sicuramente incentivato da alcune esperienze vissute durante il mio tirocinio in azienda, che mi hanno permesso di entrare in contatto con questa realtà. Invece l'interesse per la parte riguardante l'organizzazione aziendale proviene dal mio corso di studi.

L'obiettivo di questa tesi è quello di fornire un'analisi accurata dei dati raccolti, mettendone in evidenza le peculiarità del metodo. L'elaborato, in questo modo, mira a proporre delle nuove chiavi di lettura di questa metodologia che si sta diffondendo ogni giorno di più.

Escludendo questa breve introduzione, la tesi è articolata in quattro capitoli: nel primo capitolo viene fornita una visione generale dell'azienda CGM, dove è avvenuta l'applicazione pratica della metodologia. Nel secondo capitolo mi occupo di sviluppare un resoconto teorico della metodologia Agile e del perché al giorno d'oggi sempre più aziende la stanno adottando. Il terzo capitolo si concentra sull'applicazione della teoria al progetto di implementazione in CGM. Nel quarto capitolo, infine, si procede a commentare i risultati ottenuti dall'analisi dell'elaborazione dei dati, esponendo gli elementi più rilevanti dell'indagine svolta.

Grazie a questo lavoro di ricerca è stato possibile analizzare alcuni importanti fattori legati all'importanza di seguire questa metodologia con i notevoli vantaggi che porta.

2. CompuGroup Medical

2.1 CGM nel mondo

CompuGroup Medical SE & Co. KGaA è un'azienda di sanità elettronica, leader a livello mondiale, con un fatturato intorno ai 746 milioni di euro. I suoi software sono progettati per supportare tutte le attività mediche e organizzative negli ambulatori medici, nelle farmacie, nei laboratori e negli ospedali. CGM ha sedi sparse in 18 nazioni e commercia i suoi prodotti in oltre **56 Paesi**. Tutto ciò è reso possibile dai suoi **6.100 dipendenti** e da un bacino di utenza di circa **1.5M di clienti**.

2.2 CGM in Italia

"Nel mercato italiano ci sono grandi potenzialità di sviluppo, anche nel settore della sanità elettronica. Con soluzioni innovative si possono migliorare molti servizi per i cittadini. CGM ci crede." (Frank Gotthardrt, - Fondatore di CompuGroup Medical AG)

Attraverso le sue aziende, CompuGroup Medical Italia Group (CGM Italia Group) in Italia fornisce soluzioni software e servizi a **30.000 clienti** tra medici delle cure primarie, farmacie, dentisti e medici specialisti, oltre che ad amministrazioni pubbliche e pazienti. Inoltre, i prodotti sviluppati in Italia sono personalizzati per essere venduti anche in Malesia, Spagna e Francia.

Il team di CGM Italia Group è costituito da **550 collaboratori**, suddivisi in **10 sedi principali** (35 in totale) dal nord al sud del Paese.

Da 30 anni migliaia di medici, farmacisti, dentisti e specialisti si affidano all'esperienza di CGM per migliorare la qualità del proprio lavoro, grazie a soluzioni software che contribuiscono ad aumentare l'efficienza, ridurre i costi ed eliminare gli errori, a vantaggio di tutti e del Sistema Sanitario Nazionale stesso.

2.2.1 Aree di intervento

CGM Italia Group è la più grande azienda in Italia che si concentra esclusivamente sulla e-Health e che può vantare un'offerta ampia, variegata e integrata di soluzioni

software e servizi per la rete sanitaria territoriale. Questo si traduce in un'economia di tempo, un'elevazione degli standard lavorativi, un'ottimizzazione dei servizi sanitari offerti ai cittadini. Tutto ciò non solo avvantaggia gli utenti delle soluzioni CGM ma dimostra l'impegno a sostenere Stato e Regioni, affinché sia fornito il miglior supporto medico, con una gestione efficiente della spesa.

Ogni giorno sono gli utenti a contribuire al corretto funzionamento del Sistema Sanitario Nazionale utilizzando i sistemi informativi ambulatoriali e regionali, gli strumenti di rete, connettività e mobilità che CGM Italia Group studia e sviluppa per sostenere il loro lavoro quotidiano.

I principali settori in cui CGM opera sono: Sistemi informativi ambulatoriali, Sistemi informativi regionali e Sistemi di reti, connettività e mobilità per Medici delle Cure Primarie, Farmacie, Dentisti, Medici Specialisti, Cittadini, Stato e Regioni, come mostrato nella figura sottostante:

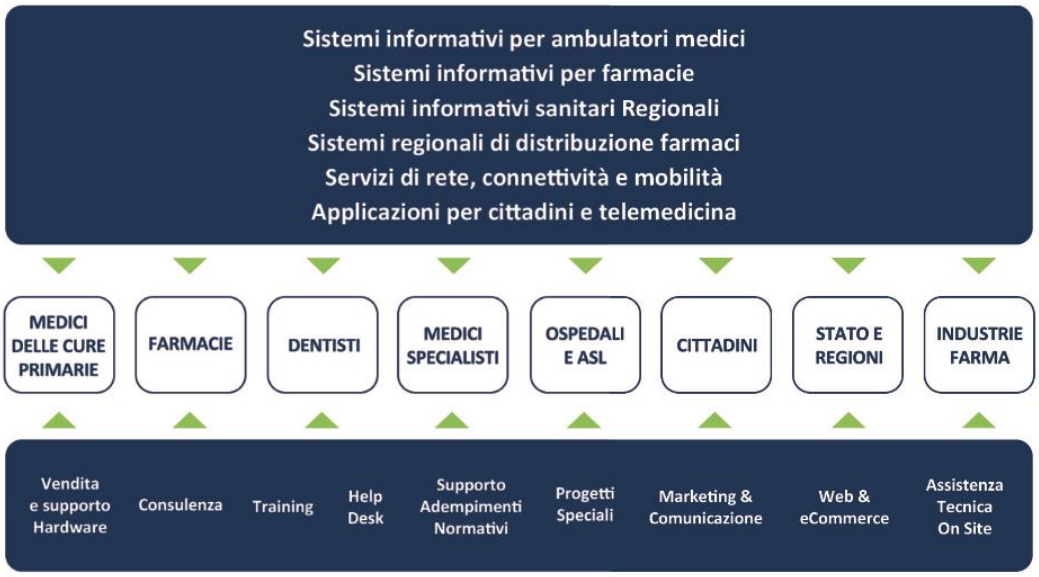

FIGURA 1- PANORAMICA ATTIVITÀ CGM

Dentisti

Centrandosi maggiormente in questo settore, CGM Italia Group risponde alle attuali esigenze dei piccoli e dei grandi studi dentistici mono e multiprofessionali, con XDENT, una innovativa soluzione multipiattaforma per l'odontoiatria che utilizza le tecnologie software più avanzate per rendere lo studio dentistico efficiente e adeguarlo alle nuove sfide del mercato. È un insieme di soluzioni integrate, progettate per sistemi Mac OsX, Windows e Linux, così come per i nuovi dispositivi Smartphone e Tablet, in grado di sfruttare al massimo le caratteristiche operative di ogni dispositivo e dare agli utenti la migliore esperienza d'uso possibile, semplificando i processi operativi e decisionali, riducendo i tempi di lavoro e ottimizzando i costi dello studio.

Con XDENT CLOUD è possibile utilizzare ovunque il sistema informatico tramite una connessione ad internet, mentre CGM X-RAY è la soluzione per gestire in maniera semplice e integrata le immagini diagnostiche di studio. La massima attenzione al paziente e la sua fidelizzazione sono le direzioni in cui si collocano i servizi resi disponibili da XDENT, tra cui XINFO, un sistema innovativo che racchiude in un solo prodotto uno strumento di marketing e di comunicazione, che consente al paziente di essere sempre in contatto con il suo studio dentistico e al medico di essere sempre a fianco dei suoi pazienti.

XDENT supera i confini nazionali e si inserisce nel mercato internazionale. Con un progetto di localizzazione concluso in Malesia, Singapore e Sudafrica, CGM Italia Group esporta la qualità e l'eccellenza di un prodotto italiano in tutto il mondo.

3. Metodologia Agile,

3.1 Che cosa è Agile

Il termine Agile è stato usato per la prima volta nel 2001 con la pubblicazione del Manifesto per l'Agile Software Development, anche se, prima di allora, c'erano già aziende che usavano un modello organizzativo basato su Agile. Gore, Toyota, Morning Star, Haier e Semco sono alcuni esempi di aziende che hanno cominciato prima delle altre a lavorare secondo il **paradigma di autonomia condivisa**, (Allegati) su cui si basa la metodologia Agile.

In sostanza Agile è prima di tutto un **mindset**, un modo di pensare, un paradigma organizzativo descritto da **4 linee guida** che ne costituiscono la filosofia:

1. Guidata dalle persone al posto di politiche e procedure
2. Facilitazione al posto di compiti e controllo
3. Processo iterativo-incrementale e misurazione dei progressi al posto della predizione
4. Rispondere al cambiamento al posto di seguire budget e piani

e definito da **10 principi**.

3.2 Principi su cui si basano le modalità operative del "Mindset Agile"

1. Si inizia sempre dalla centralità del Cliente attraverso i bisogni dei clienti soddisfatti attraverso un consistente miglioramento continuo della "customer experience".

Il **cliente** è sia il punto di partenza che il punto di arrivo. Si tratta di partire dai bisogni reali del cliente e di usare la sua esperienza con il prodotto per creare tempestivamente prodotti e servizi che il cliente può provare e l'organizzazione può migliorare costantemente. In Agile non esiste un prodotto / servizio finito, solo incrementi di miglioramento. Il focus è la creazione di valore per il cliente finale.

2. Le strategie sono altamente adattive, flessibili e capaci di rispondere rapidamente attraverso continui cambiamenti. La "Business Agility" è guidata dal cambiamento continuo.

L'obiettivo è sfidare i concorrenti a costruire una relazione unica con i clienti e fornire soluzioni immediate ai loro problemi. Per rispondere a un nuovo modello economico e sociale, è essenziale una mentalità agile. Essere Business Agile significa anticipare il cambiamento e rispondere di conseguenza. La capacità di combinare velocità e innovazione guidate dai dati è diventata la leva principale del vantaggio competitivo.

3. Operazioni basate su iterazione "time boxed" di breve periodo focalizzate a fornire valore al cliente finale attraverso progressi costanti. Essenziale minimizzare sprechi di sforzi, energie, duplicazioni, rilavorazione, persone e risorse (Ricordate: le persone umane non sono considerate risorse, sono persone).

Il focus è su **come lavora il sistema** nel suo insieme e sullo **scopo** dell'organizzazione. La causa di una variazione di performance è nel sistema. La creazione di valore per il cliente implica la gestione del flusso, che comporta la gestione della domanda di valore del cliente e la sua soddisfazione dal punto di vista del cliente, cioè la fornitura di un buon servizio ad un costo inferiore, perché questo comporta l'eliminazione di tutte le attività non a valore aggiunto prodotte dal sistema tradizionale.

4. Collaborazione cross-funzionale guidata da uno scopo chiaro e condiviso.

Un obiettivo organizzativo chiaro e condiviso è essenziale per garantire che tutti siano concentrati a fare "la cosa giusta" per i clienti. Una collaborazione inter-funzionale è necessaria per assicurare un flusso continuo di conoscenze e informazioni in tutta l'organizzazione. Questo per aumentare la capacità di raccogliere e disseminare le informazioni sui cambiamenti esterni e come rispondere rapidamente e convenientemente a questi cambiamenti.

5. Costruire e mantenere un ambiente che permette: motivazione personale, indipendenza decisionale, fiducia, apprendimento dagli errori, agevole con i dissensi, quindi un ambiente sicuro.

Si tratta di responsabilizzare i team, dare loro fiducia e autonomia, e creare un ambiente sicuro che permetta alle persone e ai team di imparare dagli errori e identificare i cambiamenti che possono essere implementati su base regolare per migliorare la velocità, l'adattabilità e l'innovazione, tutte cose che sono strettamente allineate con le esigenze e le esperienze dei clienti.

6. Il modo più efficiente ed efficace di trasmettere, condividere le informazioni e confrontarsi su di esse è faccia a faccia, ergo minimizzare la burocrazia e, dove non è possibile, automatizzarla.

Il focus dell'agilità è sulla velocità con cui i lavoratori possono intervenire per risolvere i problemi, che è legata al processo decisionale e di controllo, all'autonomia decisionale, alla semplicità procedurale e alla rapida adozione di nuove tecnologie abilitanti. Il confronto diretto è essenziale per prendere decisioni rapide efficacemente.

7. L'unico modo di misurare i progressi è misurare i risultati del lavoro operativo. Per mantenere ritmo e agility costanti occorre una ossessiva focalizzazione all'eccellenza operativa, tecnologia e design.

Agile introduce un nuovo modo di pensare ai processi decisionali che includono un flusso costante di dati sui clienti a livello locale attraverso la creazione di prototipi che vengono testati in risposte rapide e poi trasformati in nuovi flussi di dati e nuova prototipazione.

8. "Agile Business" supporta innovazione e progressi continui e sostenibili. Iterazioni e cambiamenti sono costanti e il ritmo dei progressi continua costantemente. La produzione di idee è diffusa a tutte le persone dell'organizzazione che crea un ambiente dove la sperimentazione continua ha un ruolo chiave nell'innovazione e nel cambiamento costanti.

Le nuove idee emergono spontaneamente dalle interazioni con gli utenti e non sono più di pertinenza dei dipartimenti di ricerca e sviluppo. Ciò significa che gli sperimentatori imparano in tempo reale attraverso le interazioni piuttosto che seguendo procedure preesistenti. Questo modo di agire non solo riduce i rischi, ma permette anche un'**innovazione continua**.

9. I migliori risultati emergono da piccoli team indipendenti, autonomi e auto-organizzati. L'organizzazione è un network di team di scambio che apportano valore al cliente finale.

I team multifunzionali, che sono auto-organizzati e responsabili dell'intero processo end-to-end, sono il modo migliore per fornire valore al cliente finale nel più breve tempo possibile perché sono in grado di prendere decisioni rapide nel momento di rispondere a una domanda e modificare il prodotto/servizio in base alle esigenze del cliente. L'organizzazione come sistema facilita i team nello scambiare informazioni, conoscenze e "know how" mantenendo vivo il network relazionale tra persone e teams.

10. Il miglioramento continuo si ottiene con dei momenti di riflessione, ad intervalli regolari, focalizzati ad apprendere dagli errori, come diventare più efficaci, cosa migliorare, quali pratiche e comportamenti cambiare e come.

I team trovano sempre il tempo per dei momenti di retrospezione in cui scambiarsi feedback e soprattutto identificare le azioni da sperimentare durante le successive iterazioni.

3.2.1 Team Agile

Un team è un gruppo di persone che lavora insieme con un obiettivo comune, coordinando il proprio lavoro e i propri sforzi. Qual è la differenza tra un team tradizionale e un "team agile"? Un "team agile" non ha un team leader o un "project manager" per assegnare le responsabilità. Un "team agile" si auto-organizza, è autonomo e ha la capacità di prendere decisioni indipendenti. Per un'organizzazione focalizzata sull'agilità, questo significa non essere divisi in reparti e uffici stagnanti che non comunicano tra loro. Il team è l'unica divisione aziendale che può essere riconosciuta, e la collaborazione è all'ordine del giorno. Le **reti di team** rendono le organizzazioni più dinamiche e adatte al contesto attuale.

Formare un team

In Agility i team si formano su base volontaria e le persone sono libere di scegliere a che progetto lavorare e conseguentemente a quale team appartenere. Le organizzazioni Agile per costituire un team partono da una **Call To Action**

(CTA). Un CTA non è altro che una chiamata all'azione per coinvolgere le persone in un'organizzazione che sta formando un nuovo team o che ne ha già uno ma ha bisogno di più membri. Tutti i membri del team vengono aggiornati sul progetto per il quale lavoreranno, sugli obiettivi che perseguiranno e sulle competenze di base di cui avranno bisogno per iniziare.

I team si creano e si sciolgono in base alle proposte di nuove idee e progetti, seguendo il flusso delle esigenze dei clienti e delle nuove proposte per risolvere i loro problemi. La stragrande maggioranza dei team che rispondono alle esigenze dei clienti rimangono relativamente stabili, il che significa che il loro rendimento continuerà a migliorare nel tempo. L'autonomia garantisce lo sviluppo del team verso nuove frontiere e rende il lavoro tutt'altro che routinario.

Caratteristica del team agile

Caratteristica distintiva dei team Agile è la collaborazione. La collaborazione non può essere forzata, ma deve essere creata attraverso l'impegno in attività che servono a raggiungere un obiettivo chiaro e condiviso. Ogni persona si assume la responsabilità delle proprie azioni di fronte ai risultati del progetto, pienamente consapevole che ogni persona contribuisce alla crescita e allo sviluppo del team. Il team Agile è un gruppo di persone che impara costantemente e sviluppa nuove idee. È necessario sperimentare per sviluppare l'apprendimento, il che comporta il rischio di commettere errori. Ma è da questi errori che la squadra impara, modifica i comportamenti e le pratiche, migliora e innova. Per sperimentare, è necessario creare un ambiente sicuro in cui esplorare, sbagliare e imparare. L'apprendimento rapido è strettamente connesso alla possibilità di dare e ricevere **feedback** contestuali, puntuali e tempestivi. Parlare di cose scomode e dire ciò che si pensa aumenta il rispetto reciproco, la fiducia, l'intimità intellettuale e sociale e contribuisce a rendere innovativo l'ambiente in cui cresce un team come in una start up. Un ambiente che realizza la collaborazione.

Processo di sviluppo di team Agile

Le fasi di sviluppo di un team possono essere sintetizzate in **tre periodi**:

1) Fase di apprendimento

Un primo periodo in cui il team, supportato dall'"Agile Coach[1]", assume consapevolezza di come lavora e inizia ad apprendere l'importanza di misurare le proprie attività per capire come migliorare. In questa fase, l'Agile Coach assiste il team nel comprendere come queste attività possono essere drasticamente cambiate in una prospettiva Agile. Mostra al team come applicare i fondamenti di Agile nella pratica, assistendoli nell'interiorizzare la "mentalità Agile" come un nuovo modo di pensare l'organizzazione, i processi, il lavoro, l'obiettivo finale del loro lavoro, e il valore per il cliente finale. Questa è la **fase di apprendimento** che richiede una certa disciplina da parte del team, soprattutto in termini di impegno reciproco tra i componenti del team e i risultati.

2) Fase di sperimentazione e adattamento

Una seconda fase vede il team molto più attivo nel proporre soluzioni, discuterle tra di loro e iniziare a sperimentarle. Il team è ancora insicuro e l'"Agile Coach" svolge un ruolo di facilitazione rispetto alle decisioni del team. In questa fase il team ha fatto propri i concetti e i principi che fondano l'Agility ed una **fase di sperimentazione e adattamento**. Ciò che è stato consolidato nella fase precedente arriva ora ad essere quasi interiorizzato e adattato ad ogni membro del team con caratteristiche diverse, personalità diverse e probabilmente culture di appartenenza diverse. Ora il facilitatore lascia che il team sperimenti il proprio modo di "essere" Agile.

3) Fase creativa e innovativa

Nella terza fase il team inizia ad essere autonomo e la focalizzazione supera il "problem solving" per entrare in un'area creativa dove la sperimentazione è diventata una pratica consolidata. È in questa fase che il team sviluppa nuovi metodi di lavoro e li sperimenta. È in questa fase che si mettono le basi per il successivo *scale up*. È in questa fase che emergono le nuove idee che si concretizzano in nuove soluzioni, nuovi prodotti. È in questa fase che si genera innovazione e si aprono possibilità nuove di sviluppo. Questa è la **fase creativa e innovativa** dove il team ha

[1] Un coach Agile aiuta i team e gli individui ad adottare pratiche e metodi Agile nel loro lavoro. L'obiettivo di un Agile coach è quello di rendere un team più efficiente, trasparente e coeso, e di permettere migliori risultati, soluzioni e prodotti/servizi.

ormai acquisito autonomia e indipendenza, l'apprendimento è diventato una costante e permette nuove scoperte.

Costruire su fondamenta solide

Una volta creato il team, è importante ricordare che i "team Agile", proprio come le persone, hanno bisogno di tempo per crescere. Una volta che il team ha raggiunto la fase di efficienza, entra nel processo di sviluppo in tempo reale. I membri sono motivati dalla fiducia reciproca, e conoscono i punti di forza e le debolezze di ciascuno, che usano per identificare diversi modi per migliorare lo sviluppo del software. Per mantenere intatto questo ambiente di lavoro del "team Agile" occorre un po' di disciplina organizzativa.

Alla base dei team Agile con elevate prestazioni vi sono inoltre **solide pratiche di progettazione** come revisioni del codice, creazione di branch di task[2], "continuous integration" e rilasci a cadenza regolare. I team Agile di successo si fondano inoltre su altri due pilastri: "**mentoring" continuo e insiemi di competenze condivise**. Uno dei vantaggi più significativi del lavoro di squadra è lo scambio di conoscenze e il "mentoring" reciproco. Le attività di "mentoring" non sono limitate ai membri junior che devono imparare dai membri senior. Per fare questo, tutti i membri del team imparano gli uni dagli altri, in modo che l'impatto collettivo del team sia maggiore della somma degli impatti dei singoli membri. Allo stesso tempo, le competenze condivise rafforzano la capacità della squadra di affrontare una varietà di compiti. In qualità di tecnici, è sempre importante acquisire nuove competenze per essere più utili all'organizzazione ed essere meglio preparati per supportare il lavoro del team. Ciò consente inoltre di vigilare che nessuno intraprenda un percorso critico, togliendo un peso enorme dalle spalle di tutti.

Collaborazione dei "team Agile" nei diversi reparti

A parte gli sviluppatori e i tester, il team odierni del software include product manager, project manager, specialisti di marketing e specialisti delle operazioni.

---

[2] La ramificazione dei compiti, conosciuta anche come ramificazione dei problemi, collega direttamente questi problemi con il codice sorgente. Ogni storia utente (o correzione di bug) vive all'interno del proprio ramo, rendendo facile vedere quali problemi sono in corso e quali sono pronti per il rilascio.

Prendendo Atlassian come esempio, il loro "team Agile" si concentra su tre fasi del processo di sviluppo del prodotto: creazione, vendita e utilizzo. Ogni fase del prodotto è supportata da tre team (idealmente composti da 5 a 7 membri ciascuno) che formano una triade. Ogni triade adotta un approccio Agile poiché durante la fase di sviluppo del prodotto i team lavorano costantemente su ogni fase apprendendo nuovi concetti sul prodotto stesso e sul mercato.

Team remoti

Lo sviluppo agile è stato originariamente concepito per team raggruppati, o team che erano fisicamente situati nello stesso ufficio. In linea con la premessa che "il metodo più efficiente ed efficace per trasmettere informazioni all'interno di un team di sviluppo è la comunicazione faccia a faccia", i primi team agili avevano il compito di lavorare insieme in stretta vicinanza.

Ma oggi la maggior parte delle aziende ha alcuni o diversi team distribuiti. I team distribuiti possono lavorare su progetti 24 ore su 24. Tuttavia, i benefici dei team distribuiti hanno alcuni compromessi. Per molti di questi team, è difficile adottare la pratica agile delle interazioni faccia a faccia. Altre sfide che sorgono per i team distribuiti includono: coordinare le attività attraverso i fusi orari, stabilire relazioni quando tutti non sono nello stesso ufficio, collaborare attraverso diverse culture di sviluppo, e programmare incontri informali o conversazioni quando entrambi i team sono online allo stesso tempo solo per poche ore. Questi sono problemi reali, ma non irrisolvibili. Di seguito vengono descritte alcune strategie per aiutare a colmare il divario di distanza tra uffici locali e remoti, e idee per aiutare a mitigare anche altri potenziali problemi.

Come strutturare i team globali

Una buona architettura del software impone un design modulare, quindi bisogna strutturare i team allo stesso modo. Ogni ufficio dovrebbe essere autosufficiente nello sviluppo di un singolo pezzo di tecnologia, il che riduce al minimo la quantità di collaborazione necessaria con i team in altri fusi orari e li rende generalmente autonomi. Quando un progetto richiede l'intervento di team in luoghi diversi, essi

possono concentrarsi sui loro punti di integrazione e sulle API[3]. Anche le revisioni del codice giocano un ruolo importante. Dato che le persone sono online a orari diversi, distribuire la conoscenza del codice tra gli uffici rende il supporto e la manutenzione molto più facile. Se un problema di produzione emerge quando il team non è online, un altro ufficio può facilmente intervenire per supportare e risolvere il problema, grazie al "know-how" acquisito dalle revisioni del codice tra team o tra sedi diverse.

Costruire un rapporto

È importante in qualsiasi programma, specialmente in quelli agili, avere un rapporto solido in tutto il team. La connessione personale costruisce la fiducia, minimizza le aspettative mancate, facilita l'auto-organizzazione e aumenta il morale. All'interno dell'ufficio, bisognerebbe prendersi del tempo per conoscere tutti i membri del team. E, per quanto possibile, fare lo stesso con le persone con cui si lavora negli uffici remoti. Le connessioni personali sono importanti. Più forti diventano, maggiore è la possibilità di vedere questi colleghi come qualsiasi altro, piuttosto che colleghi distanti da luoghi sconosciuti senza buone relazioni. Soprattutto, niente sostituisce l'incontro faccia a faccia. I membri del team in ogni ufficio trarranno beneficio da un regolare incontro faccia a faccia, e questo include le videoconferenze così come le visite agli uffici remoti.

Per aiutare a mitigare alcuni problemi di videoconferenza, incoraggiare i membri del team ad avere sessioni settimanali di video chat 1:1. Queste possono essere meno formali e aiutano a facilitare la condivisione delle conoscenze in modo informale. I compagni di squadra possono usare queste opportunità per costruire un rapporto e lavorare meglio insieme. Un faccia a faccia di persona aiuta il team a conoscere i colleghi remoti con maggiore fedeltà, il che, a sua volta, rende le future videoconferenze più efficaci. Che si tratti di una casa o di un prodotto, è necessario definire la visione e delineare i temi strategici. Bisogna pensare ai temi come ad aree di attenzione a livello di organizzazione.

---

[3] La definizione di un'interfaccia di programmazione delle applicazioni (API) è un insieme di protocolli, routine e strumenti per applicazioni software aziendali.

Costruire una cultura di sviluppo unita

Ci sono quattro semplici modi in cui i team possono facilitare il lavoro tra le varie aree geografiche e condividere una cultura di sviluppo comune:

1. sovra comunicare le decisioni in tutte le sedi;
2. ridurre al minimo l'attrito nella creazione dell'ambiente di sviluppo;
3. definire chiaramente la “Definition of Done”;
4. creare linee guida per archiviare rapporti di bug efficaci.

Innanzitutto, quando si passa da un ufficio locale a una cultura distribuita, la comunicazione diventa significativamente più difficile. La prima sfida è addestrare il team a capire che, quando le decisioni vengono prese, devono essere comunicate. Spesso le decisioni importanti vengono prese in conversazioni di corridoio, in riunioni informali di team locali o da singoli individui. Inoltre, può essere facile liquidare le piccole decisioni come non importanti. Comunicare anche i minimi dettagli fino a quando entrambi gli uffici non trovano un sano “groove”. Quando le decisioni vengono prese, tutti in ogni ufficio devono capire la decisione e idealmente perché è stata presa. Non usare le e-mail. È troppo facile perdere informazioni importanti. Usare un sistema di gestione dei contenuti come un “wiki” dove i membri del team possono facilmente navigare per gli aggiornamenti in tutto il team (e ricevere una notifica degli aggiornamenti tramite e-mail). I ritardi causati dai membri del team che lavorano su informazioni obsolete, che incontrano un blocco stradale e poi fanno una domanda, costano al team molto più tempo rispetto alla condivisione proattiva delle informazioni.

In secondo luogo, ambienti di sviluppo coerenti in tutto il team rendono più facile lavorare insieme e rintracciare i problemi. Dedicare del tempo alla creazione di una semplice guida "Getting Started" e domare l'attrito del primo giorno automatizzando il più possibile la configurazione.

In terzo luogo, quando si lavora tra uffici, standard chiari sulla definizione di "completo" rendono più facile gestire le aspettative e costruire rapporti tra i team. Una definizione ferma di "completo" elimina l'ambiguità nel lavoro. Per esempio,

quando si spedisce un rilascio che coinvolge più team, è necessario rendere chiaro cosa significa essere completo: codice scritto, richiesta di pull creata, codice rivisto, testato e inserito nel ramo appropriato.

E infine, distribuire lo sviluppo significa che non tutti sono online quando sorgono problemi. Avere linee guida chiare per le segnalazioni di bug e per la risoluzione dei problemi rende più facile per chiunque nel team rintracciare un problema. La revisione del codice e i buoni test automatizzati condividono anche la conoscenza del codice di base e autorizzano il team interessato a fare la correzione e a convalidare che il cambiamento non abbia effetti collaterali inaspettati. Così, nessun team diventa un blocco.

Massimizzare le ore d'oro

Ogni fotografo sa che "le ore d'oro" - appena prima e dopo l'alba e il tramonto - sono uno dei momenti più efficaci per scattare grandi foto di paesaggi. Le ore d'oro per i team di software distribuiti sono quando i team locali e remoti sono entrambi nei loro rispettivi uffici allo stesso tempo. Quando tutti i team sono in ufficio, questo è un ottimo momento per gli “stand-up”. Per i team che condividono il lavoro tra fusi orari, lo “stand-up” è un ottimo momento per passare il testimone in modo che il team appena arrivato online possa riprendere da dove l'altro team ha lasciato. E tenere lo “stand-up” in videoconferenza rende facile fare domande e aggiornarsi, così tutti sono pronti a partire non appena la riunione è finita. A volte gli uffici sono così distanti tra loro che le riunioni causeranno una qualche forma di disturbo per un team. Ruotare l'orario della riunione in modo che sia un onere condiviso, piuttosto che sottoporre continuamente il team remoto a orari estenuanti. Monitorare attentamente l'impegno dell'intero team allo “stand-up”. Se c'è una tensione eccessiva, o il team non sta ottenendo molto, i membri del team inizieranno a disimpegnarsi e smetteranno di ascoltare o condividere. Lo “stand-up” non deve assolutamente essere una riunione quotidiana. Incontrarsi con il team remoto un paio di volte a settimana e usare gli altri giorni per uno “stand-up” locale. Allo stesso modo, uno “stand-up” non deve essere una routine mattutina. Qualsiasi momento

della giornata sia più conveniente per tutte le persone coinvolte è il momento migliore.

Lavorare con specialisti

I team agili più efficaci sono piccoli, gruppi agili di 5-7 persone con insiemi di competenze diverse, ma sovrapposte. Questa struttura permette al team di sviluppare relazioni strette e di fiducia che fanno leva sulle diverse abilità e accelerano la capacità del team di consegnare il lavoro. A volte, però, gli insiemi di abilità necessari per un progetto non rientrano nelle capacità collettive di un team. Le persone con cui si lavora di solito rientrano in una di queste due categorie: **generalisti e specialisti**. Come si differenziano? Generalista è una persona con un'ampia conoscenza che può lavorare in diverse aree. Specialista è una persona che ha una conoscenza profonda e unica di una particolare area di interesse.

Molte metodologie agili sostengono che tutti i membri del team devono diventare generalisti. A volte, però, ha senso per un team arruolare l'aiuto di uno specialista per le seguenti ragioni: una particolare serie di abilità non è richiesta nel team a tempo pieno, l'azienda ha un numero limitato di persone con un particolare set di competenze, che sono condivise tra i team, oppure è necessaria un'autorizzazione specifica per lavorare in un'area a cui il team generale non ha accesso. In questi casi, ha senso che uno specialista si unisca al team per un periodo di tempo specifico. Aggiungere uno specialista al team, però, comporta alcune sfide. Poiché gli specialisti sono in un team solo per un periodo di tempo specifico, possono diventare rapidamente un "percorso critico", a volte bloccando il progresso dell'intero team. Per esempio, se un team si basa su un amministratore di database per apportare modifiche al database per distribuire un nuovo codice, il progresso del team viene bloccato dall'amministratore del database. Quando il team non può andare avanti perché ha bisogno dell'input di quello specialista, quel flusso di lavoro si ferma. Poiché lo specialista è l'unico nel team che ha le competenze, il team non ha altra scelta che aspettare fino a quando lo specialista non lo sblocca. Gli specialisti si impegnano in un sacco di commutazioni di contesto: diminuendo su un progetto e aumentando su un altro. E passare da un progetto all'altro è costoso. Gli specialisti

non hanno quasi mai la conoscenza intima di un progetto che hanno i membri del team principale. Di conseguenza, gli specialisti possono perdere dettagli importanti. Per mitigare questo, il core team deve esercitare un'energia extra per mantenere lo specialista aggiornato.

3.2.2 Perché si sta diffondendo un modello di organizzazione Agile?

Essere Agile consente alle organizzazioni di **cambiare rapidamente** direzione alla prima avvisaglia di un cambiamento di contesto e circostanze anche inaspettate. Spesso sono in grado di anticipare i cambiamenti. Questo perché hanno assunto una configurazione organizzativa Agile in grado di rispondere rapidamente alla complessità. Non viviamo certo in un'epoca caratterizzata da stabilità, **oggi** il mondo stesso si configura, nella sua globalità, **estremamente precario, in rapida evoluzione**, **instabile** e **complesso.**

Agile vive nei sistemi complessi

Le **organizzazioni** che vi operano per rispondere adeguatamente a questa evoluzione non possono che **riconfigurarsi come sistemi complessi.**

I sistemi complessi sono per natura decentralizzati, lavorano meglio e generano maggiori possibilità rispetto a quelli centralizzati perché sono in grado di evolvere più rapidamente. Ogni parte di un sistema decentralizzato all'interno di un quadro generale di regole semplici e condivise è capace di evolvere adattandosi in fretta.

Dal palco dell'ultimo World Business Forum a Milano, **Gary Hamel** ha chiarito bene che: "nel mondo di oggi **tutto è veloce**, siamo sommersi da informazioni" e "il 94% dei CEO afferma che la propria impresa **non è in grado di innovare** malgrado lo ritenga necessario". Le organizzazioni non riescono ad affrontare il cambiamento, il "change management top down" non funziona più.

Sempre Hamel sostiene che **non sono le persone che fanno fatica a cambiare, sono le organizzazioni,** ma **è sulle persone in prima linea che bisogna puntare**. Persone che in genere non sono formate per innovare e non sono coinvolte nelle decisioni.

Non stupisce che il livello di "engagement" del personale oggi sia ai minimi: solo 15% a livello mondiale si dichiara motivato. Ma come è possibile essere motivati quando la cooperazione è modesta, le responsabilità ristrette, le interconnessioni appena tollerate, le novità portatrici di problema e un fallimento porta all'identificazione di uno o più colpevoli?

I sistemi che funzionano fondandosi sul **paradigma di autonomia condivisa** invece si **basano sull'impatto che hanno dati e persone**, sulla collaborazione diffusa, sulle decisioni e sui rischi condivisi, sulla necessità delle interconnessioni di persone, team, dati e informazioni, sui fallimenti portatori di nuovi insegnamenti, sulla ricerca di novità.

La trasformazione Agile silenziosa

La principale caratteristica del **cambiamento di Mindset** nel paradigma di autonomia condivisa è quella di configurarsi come una **trasformazione silenziosa,** tale perché agisce senza farsi sentire.

I **cambiamenti efficaci non sono delle rivoluzioni** e tantomeno dei progetti di change management, sono delle "trasformazioni silenziose", dei **micro cambiamenti** che si infiltrano lentamente, ma inesorabilmente. Cambiamenti che subentrano senza che ce ne accorgiamo se non quando si sono radicati nella realtà diventando concreti.

I progetti di change management fanno parte della mentalità convenzionale e hanno una forte similitudine con le rivoluzioni, movimenti che agiscono con forza portando un apparente rovesciamento momentaneo. I cambiamenti sostanziali agiscono in profondità e cambiano i comportamenti delle persone. Quasi tutte le rivoluzioni hanno portato a nuove istituzioni che hanno semplicemente preso il posto delle vecchie **senza cambiare la sostanza delle cose.**

Le **trasformazioni silenziose** invece vengono sottovalutate perché invisibili, non combattono, non contrastano, non sono atti di forza, **si insinuano silenziose e si propagano** senza suscitare nessun allarme, fino a quando non ci si accorge del ribaltamento che hanno prodotto. Le trasformazioni silenziose non hanno leader perché sono idee e comportamenti che finiscono per prevalere. Agiscono con il

tempo, più come un processo che come un evento e una volta che si sono diffuse resistere diventa stupido ed anacronistico.

Come affermava Richard Buckminster Fuller: *"... non potrai mai cambiare le cose combattendo contro la realtà esistente. Per cambiare qualcosa, crea un nuovo modello che rende il modello esistente obsoleto."*

Un "mindset Agile" si basa su conoscenze che crescono costantemente e rapidamente, stratificandosi l'una sull'altra il che richiede **nuove conoscenze, nuove capacità e competenze, apprendimento e adattamento continuo.** Non basta adottare nuove metodologie e nuovi strumenti, il "mindset Agile" si costruisce man mano, sperimentando e imparando costantemente, sono i risultati che ti indicano la nuova strada da percorrere, ecco perché un'organizzazione agile non potrà mai fare domani quello che faceva ieri. Per citare Peter Drucker *"il miglior modo di predire il futuro è crearlo"*.

Il vantaggio agile

Basterebbe chiedere a qualsiasi sviluppatore della sua transizione dal "waterfall" all'agile e elencherebbero molti benefici. Invece per le parti interessate al di fuori dell'organizzazione di sviluppo, questi vantaggi non sono così evidenti. Questa sezione serve ad aiutare i business leader a capire il valore di agile e ad allineare i loro obiettivi di business con le pratiche di sviluppo. Venti anni dopo la sua fondazione, le pratiche agili rimangono più rilevanti che mai e le aziende che abbracciano agile continuano a guidare il mercato.

Collegare la strategia di business alla realtà dello sviluppo

Un processo agile ben sintonizzato a livello di team è la base per un efficiente sviluppo agile del software. Tuttavia, per essere efficaci e raggiungere gli obiettivi di mercato e di business desiderati, è essenziale che il lavoro quotidiano di un team sia in sintonia con gli obiettivi strategici dell'organizzazione. La chiave per allineare la strategia di business con ciò che sta accadendo in prima linea nello sviluppo è definire chiaramente temi, obiettivi e metriche.

I **temi** sono grandi aree di lavoro correlate, definite per un periodo di tempo e focalizzate su un particolare risultato. Per esempio, un tema potrebbe essere l'ottimizzazione del flusso del carrello della spesa nei prossimi due trimestri. I temi sono un importante quadro di riferimento per i team. Servono a controllare se il loro lavoro sta contribuendo al progresso delle iniziative aziendali. Gestire il lavoro per tema aiuta anche il management a capire se sono state assegnate risorse sufficienti per il successo, o se i temi sono sotto finanziati.

Obiettivi e **metriche** definiscono uno stato futuro desiderato, concreto e misurabile. Nella gestione agile del portfolio, gli obiettivi globali danno un contesto ai temi e possono essere suddivisi in sotto-obiettivi per guidare azioni misurabili a tutti i livelli dell'organizzazione. Per esempio, una riduzione del 20% dei carrelli abbandonati.

Una volta definiti i temi, gli obiettivi e le metriche di alto livello, i dipartimenti e i team possono derivare i loro particolari sotto-obiettivi dai temi. I sotto-obiettivi e le iniziative chiave aiutano a definire le caratteristiche dei prodotti/progetti. Di conseguenza, ogni team di software dovrebbe capire come ogni singolo compito contribuisce agli obiettivi e ai temi. In altre parole, perché è importante per la strategia generale.

La struttura di cui sopra serve a due scopi:

1. focalizza il tempo su ciò che conta di più, ed evita lo spreco di risorse su non-obiettivi;
2. fornisce il contesto di cui i singoli membri del team hanno bisogno per prendere le decisioni giuste ogni giorno.

Un'organizzazione non può raggiungere i suoi obiettivi senza raggruppare e concentrare le sue risorse. Inoltre, indipendentemente dal tipo di lavoro, gli individui prendono innumerevoli decisioni di compromesso ogni giorno su come i compiti vengono svolti. In definitiva, i manager non possono e non devono essere coinvolti in queste decisioni di micro-livello. Il management può solo fornire le giuste informazioni e l'ambiente giusto per permettere agli individui di agire nel migliore interesse degli obiettivi globali.

Osservazioni e prossimi passi

Collegare lo sviluppo quotidiano con la strategia aziendale è un processo a doppio senso. Guardando dall'alto verso il basso, è essenziale stabilire i confini e le aree di attenzione in cui i team devono lavorare. Queste aree di focalizzazione derivano direttamente dal business plan e dalla strategia aziendale generale. Per ogni area di focalizzazione ci dovrebbe essere un obiettivo finale chiaramente definito e misurabile. Dal basso verso l'alto, assicurarsi che tutti sappiano a quali temi e obiettivi ogni compito contribuisce. Sollevare preoccupazioni se questo non è chiaro, in quanto indica o un disallineamento sugli obiettivi, o una mancanza di attenzione.

La “roadmap” è un buon punto di partenza, specialmente se l'implementazione dei temi e degli obiettivi sembra intimidatoria. La “roadmap” costringerà i “product owners” a riflettere seriamente su come le “epics” e le “user stories” stanno contribuendo alla strategia aziendale e se particolari aree di lavoro sono importanti. I “thems” aiuteranno anche a tracciare gli investimenti di risorse all'inizio del progetto. Tracciare gli investimenti in ogni nuovo tema sulla “roadmap” per assicurare che tutti i “thems” siano ben finanziati e non siano impostati per il fallimento.

Agile è il vantaggio competitivo per l'era digitale

In un nuovo rapporto, Harvard Business Review Analytic Services ha definito lo sviluppo agile del software "il vantaggio competitivo per l'era digitale". Quello che il rapporto delinea è che è avvenuto un cambiamento nelle tendenze di sviluppo. Con così tanta concorrenza là fuori, i team di sviluppo software devono soddisfare le esigenze dei clienti più velocemente. Prima, cioè, che un concorrente li batta sul tempo. Passare a forme più agili di pianificazione e consegna del software aiuta i team a fare proprio questo, ma è anche più di questo. Agile aiuta le organizzazioni in diversi settori a migliorare la qualità del prodotto, il “time to market” e la soddisfazione dei dipendenti. La cosa da tenere a mente, tuttavia, è che "fare bene l'agile non è una soluzione cosmetica, ma qualcosa a cui le aziende devono dedicare tempo e risorse in diverse aree chiave". Quindi, ci vuole un po' di tempo.

Andare agile

Troppo spesso, le aziende partono con la missione di "diventare agili" prima di capire veramente cosa significa. Le crepe si mostrano subito e le aspettative sono

mancate, lasciando tutti a mettere in discussione il valore di "diventare agili" del tutto. La verità è che diventare agili porterà a team più produttivi e a una consegna più veloce dei progetti, ma solo se tutti possono essere d'accordo sulle regole del gioco.

3.2.3 Qual è l'obiettivo delle aziende che adottano Agile

Anche se la diffusione su grande scala di Agile è avvenuta nelle organizzazioni di sviluppo software all'inizio del nuovo millennio, tra il 2015 e il 2018 è avvenuto un vero e proprio "tipping point[4]". Non solo il **94%** delle aziende di sviluppo software negli Stati Uniti e in Inghilterra hanno adottato l'Agility, ma anche molte altre aziende in altri settori hanno iniziato ad introdurre l'Agility nelle loro organizzazioni. La terza onda del movimento Agile ha dimostrato a tutti quanti che il focus delle organizzazioni che adottano Agile si sta spostando sul **business**.

Una volta che un'organizzazione è diventata Agile, non è più solo un modo di lavorare internamente; è anche un modo di fare affari, produrre innovazione e sviluppare nuovi modelli di business basati su interazioni e personalizzazioni non solo con i clienti ma anche con gli utenti, compresi quelli che non sono ancora clienti ma potrebbero esserlo. In questo senso, le interazioni sono più importanti delle transazioni. Le organizzazioni che hanno adottato l'agilità organizzativa provengono da una vasta gamma di settori. Questo significa che l'atteggiamento Agile può essere applicato a qualsiasi organizzazione. Alcune hanno adottato metodologie Agile e una mentalità "approccio Agile end to end[5]", risultando in "Business Agility" con benefici significativi. Altre hanno implementato Agile solo in alcuni processi organizzativi ricevendo comunque eccellenti risultati. Ogni organizzazione ha implementato Agile a modo suo. Non esiste un metodo uguale per tutti. Ognuno trova il proprio framework e il proprio modello operativo utilizzando i molti strumenti che offre il "mindset Agile".

---

[4] Il punto in cui una serie di piccoli cambiamenti o incidenti diventa abbastanza significativa da causare un cambiamento più grande e importante.

[5] Scalata agile che include tutte le funzioni aziendali e di sviluppo/IT necessarie per ottenere un flusso veloce, flessibile, di valore aziendale attraverso il prodotto target (o flusso di valore, capacità o percorso del cliente).

3.4 SAFe Agile

Scaled Agile

I team di sviluppo software hanno dimostrato che l'implementazione di framework agili come scrum e kanban permette loro di fornire soluzioni ai clienti più rapidamente, con maggiore prevedibilità, e dà loro la capacità di reagire rapidamente in risposta a nuove esigenze e sollecitazioni. È relativamente semplice implementare l'agile a livello di un singolo team. Ma la vera sfida è estenderlo a più team in una grande organizzazione. In altre parole, implementare agile su scala.

Gestire un portafoglio agile

Le pratiche agili possono funzionare in un grande portfolio con molti team e molti sviluppatori. Netflix ha coniato la frase "altamente allineato, vagamente accoppiato" per descrivere uno sviluppo agile ben calibrato in una grande organizzazione. Diamo un'occhiata al ruolo del contesto e dell'autonomia, così come ad alcune caratteristiche comuni trovate tra i portafogli agili ad alto funzionamento.

Impostare il giusto contesto

Con così tanta attenzione al team, il senior management in un'organizzazione agile su larga scala è spesso lasciato a chiedersi quale sia il suo ruolo. Nella sua capacità di guidare la strategia e la direzione, l'alta dirigenza può servire come punto focale per la cultura agile dell'organizzazione. La strategia e la direzione si manifestano come temi nello sviluppo agile di un portafoglio di prodotti. I temi aiutano molti team che svolgono un'idea di business a collaborare efficacemente e a raggiungere gli obiettivi dell'organizzazione. Una cultura trasparente permette ai problemi di emergere senza paura di ripercussioni e minimizza gli aspetti negativi della politica aziendale. Di conseguenza, è più facile trovare la giusta soluzione e far andare avanti i team.

Espandere le pratiche agili in tutta l'organizzazione

Le aziende di maggior successo che eseguono l'agile su scala condividono tre tratti comuni.

Primo, **l'intero programma è iterativo**. La gestione tradizionale del portafoglio si concentra sulla gradualità del lavoro dall'alto verso il basso per lunghi periodi di tempo, ma la gestione agile del portafoglio prende il concetto dei cicli di costruzione-misurazione-apprendimento del team agile e lo applica su una scala più ampia. La pianificazione agile a lungo termine è guidata da un team di persone che lavorano insieme, utilizzando un progetto modulare e condividendo i risultati su base regolare. Ed è stato dato loro il permesso di fare compromessi nel contesto del loro lavoro, la tempestività della loro esecuzione e le risorse che sono state date per portare a termine il lavoro. Questo si traduce in un'enorme flessibilità, che sposta l'attenzione dal continuare ad eseguire un piano non flessibile al dare valore e fare progressi tangibili secondo la strategia e gli obiettivi aziendali.

In secondo luogo, **le organizzazioni di successo comunicano attraverso il portafoglio**. Condividono la conoscenza e abbattono le barriere tra i silos organizzativi. Similmente ai meeting dei team agili, il percorso dovrebbe essere condiviso frequentemente in tutta l'organizzazione in modo che gli obiettivi, i progressi e i risultati siano trasparenti per tutti. Questo favorisce il rispetto tra compagni di squadra e colleghi, indipendentemente dal ruolo all'interno dell'organizzazione, e incoraggia le interazioni che sono radicate nell'empatia e nella comprensione.

In terzo luogo, le organizzazioni più agili fanno **frequenti rilasci in tutto il portafoglio**, anche se un rilascio coinvolge il lavoro di più programmi. I cicli di sprint allineati, il lavoro investito in API forti e nel disaccoppiamento tecnico, e un'efficiente pipeline di test[6] e distribuzione automatizzata garantiscono una visibilità costante su chi sta producendo cosa e quando.

Condividere una visione, ma abbracciare le diversità

Come nello sviluppo agile tradizionale, il lavoro nello sviluppo agile del portafoglio è delegato ai team piuttosto che agli individui. Ogni team capisce gli obiettivi più

[6] Le pipeline di integrazione e distribuzione continue (CI/CD) sono procedure pensate per ottimizzare l'erogazione di software attraverso l'approccio DevOps.

grandi dell'organizzazione, e ogni team sviluppa anche una cultura vivace che ottimizza i propri processi e la consegna.

L'autonomia si estende anche nei flussi di lavoro per le organizzazioni di portafoglio. I team che lavorano per varie sezioni del business possono avere i loro propri flussi di lavoro. Vale la pena notare che un team di ingegneria del software avrà un diverso flusso di lavoro e procedure rispetto a un team di marketing, anche se entrambi seguono principi agili come lo sviluppo iterativo e regolari "retrospettive". Oppure due team di sviluppo possono scegliere di dividere il loro flusso di lavoro in stati diversi. Questa diversità dà ai grandi portafogli agili il beneficio della conoscenza condivisa. Provare più cose significa imparare più cose che possono essere condivise in tutta l'organizzazione.

Espandere l'agilità mentre si cresce

Una struttura agile per un grande portafoglio implica lo scaling dei principi agili usati a livello di team in tutta l'organizzazione. La cultura agile è un moltiplicatore di forza. Per garantire il successo, la leadership senior deve lavorare in partnership con tutti i team per costruire una sana cultura agile.

SAFe

Lo Scaled Agile Framework (SAFe) è un insieme di modelli di organizzazione e di flusso di lavoro per implementare pratiche agili su scala aziendale. Il framework è una base di conoscenze che include: una guida strutturata su ruoli e responsabilità, come pianificare e gestire il lavoro, e valori da sostenere.

SAFe promuove l'allineamento, la collaborazione e la consegna attraverso un gran numero di team agili. È stato formato intorno a tre corpi primari di conoscenza: sviluppo agile del software, sviluppo snello del prodotto e pensiero sistemico.

Man mano che le aziende crescono in dimensione, SAFe fornisce un approccio strutturato per scalare l'agile. Ci sono quattro configurazioni in SAFe per ospitare vari livelli di scala: Essential SAFe, Large Solution SAFe, Portfolio SAFe e Full SAFe.

Dean Leffingwell e Drew Jemilo hanno rilasciato SAFe nel 2011 per aiutare le organizzazioni a progettare sistemi e software migliori che soddisfino meglio le mutevoli esigenze dei clienti. A quel tempo, i team usavano i tradizionali processi di gestione dei progetti per consegnare il software. Ma con l'aumento della necessità di rispondere rapidamente alle mutevoli condizioni del mercato, sono emersi nuovi framework per aiutare le aziende a migliorare la consegna delle soluzioni in tutta l'azienda, ed è nato SAFe. Oggi, SAFe è uno dei framework di consegna agile in scala più popolari, e la comunità mondiale di praticanti SAFe continua ad evolverlo.

Valori fondamentali SAFe

I valori fondamentali di SAFe descrivono la cultura che la leadership deve promuovere e come le persone dovrebbero comportarsi all'interno di questa cultura per usare efficacemente il framework.

Allineamento

SAFe richiede che le aziende mettano in atto cadenze di pianificazione e riflessione a tutti i livelli dell'organizzazione. Con queste in atto, tutti capiscono lo stato attuale del business, gli obiettivi, e come tutti dovrebbero muoversi insieme per raggiungere quegli obiettivi. Sincronizzando regolarmente le persone e le attività, tutti i livelli del portafoglio rimangono allineati. Le informazioni fluiscono sia verso l'alto che verso il basso in modo tempestivo.

Qualità inclusa

Nella struttura SAFe, l'agilità non dovrebbe mai venire a scapito della qualità. SAFe richiede ai team a tutti i livelli di definire cosa significa "fatto" per ogni compito o progetto e di concludere pratiche di sviluppo di qualità in ogni accordo di lavoro. Secondo SAFe, ci sono cinque dimensioni chiave della qualità incorporata: flusso, architettura e qualità del design, qualità del codice, qualità del sistema e qualità del rilascio.

Trasparenza

SAFe incoraggia il comportamento di costruzione della fiducia, compresa la pianificazione del lavoro in lotti più piccoli in modo che i problemi possano essere

evidenziati prima, fornendo visibilità in tempo reale sui progressi del “backlog” attraverso i livelli, e i rituali di ispezione e adattamento.

Esecuzione del programma

L'esecuzione del programma è il cuore di SAFe e alimenta tutto il resto del framework. I team e i programmi devono essere in grado di consegnare qualità, software funzionante e valore aziendale su base regolare.

Leadership

SAFe richiede un comportamento di leadership lean-agile perché solo i leader possono cambiare il sistema e creare l'ambiente necessario per abbracciare tutti i valori fondamentali.

Principi SAFe

I principi dello Scaled Agile Framework hanno lo scopo di migliorare l'azienda nel suo complesso ispirando un processo decisionale snello e agile attraverso i confini funzionali e organizzativi.

Principio 1: Considerare un punto di vista economico

Ispirato dalle teorie sul flusso di sviluppo del prodotto dai libri più venduti di Donald Reinertsen, raggiungere il “lead-time” più breve e sostenibile richiede che ogni individuo nella catena decisionale capisca le implicazioni economiche dei ritardi. Consegnare presto e spesso non è sempre sufficiente. Secondo SAFe, mettere in sequenza i lavori per ottenere il massimo beneficio, capire i compromessi economici e operare all'interno di budget ridotti sono tutte responsabilità che devono essere condivise in tutta l'organizzazione.

Principio 2: Applicare il pensiero ai sistemi

SAFe incoraggia le persone che usano il framework ad applicare il pensiero sistemico a tre aree chiave: la soluzione stessa, l'impresa che costruisce il sistema e i flussi di valore. Le soluzioni possono riferirsi a prodotti, servizi o sistemi consegnati al cliente, che siano interni o esterni all'impresa. Le grandi soluzioni hanno molti componenti interconnessi, quindi i membri del team dovrebbero avere una

prospettiva di più alto livello su come la loro parte si inserisce nel quadro generale. Quando si pensa all'impresa che costruisce il sistema, le persone che seguono SAFe dovrebbero considerare le persone, la gestione e i processi dell'organizzazione. Quindi, se un'organizzazione sta cercando di ottimizzare il modo in cui le persone lavorano, potrebbe aver bisogno di eliminare i silos, diventare inter-funzionale e negoziare nuovi accordi di lavoro con fornitori e clienti. Infine, l'impresa dovrebbe definire chiaramente come il valore scorre dal concetto al denaro nei flussi di valore dello sviluppo della soluzione. I leader e la gestione devono massimizzare il flusso di valore attraverso i confini funzionali e organizzativi.

Principio 3: Assumere la variabilità; preservare le opzioni

Per default, progettare sistemi e software è un esercizio incerto. Questo principio affronta questo aspetto introducendo il concetto di progettazione basata su set, che richiede di mantenere più requisiti e opzioni di progettazione per un periodo più lungo nel ciclo di sviluppo. La progettazione basata su set si basa anche su dati empirici per restringere l'attenzione sull'opzione di progettazione finale più avanti nel processo. La progettazione basata su set aiuta a informare il processo decisionale durante i periodi di incertezza identificando le opzioni e i risultati previsti, proprio come una scommessa strategica. Più i team imparano nel tempo, più scelte possono eliminare. Più scelte eliminano, più facile è identificare il miglior percorso da seguire e produrre il miglior risultato possibile per i clienti.

Principio 4: Costruire in modo incrementale con cicli di apprendimento veloci e integrati

Simile al principio 3, questo principio affronta il rischio e l'incertezza attraverso tappe di apprendimento. Non è sufficiente che ogni componente del sistema si dimostri funzionale, l'intero sistema deve essere considerato per valutare la fattibilità delle attuali scelte progettuali. I punti di integrazione devono essere pianificati con cadenza regolare per accelerare i cicli di apprendimento. Questi punti di integrazione sono un esempio del ciclo “plan-do-check-adjust” di Walter A. Shewhart, una struttura per il miglioramento continuo della qualità e un meccanismo per controllare la variabilità dello sviluppo.

Principio 5: Basare le 'milestones' sulla valutazione oggettiva di sistemi funzionanti

La dimostrazione di un sistema effettivamente funzionante fornisce una base migliore per il processo decisionale che un documento di requisiti o qualche altra valutazione superficiale del successo. Includere le parti interessate in queste decisioni di fattibilità all'inizio supporta la costruzione della fiducia e il pensiero di sistema.

Principio 6: Visualizzare e limitare il lavoro in corso (WIP 'Work In Progress'), ridurre le dimensioni dei lotti e gestire le lunghezze delle code

Limitare il lavoro in corso aiuta le parti interessate a vedere esattamente come si sta svolgendo il lavoro. I tre elementi di questo principio rappresentano i modi principali per massimizzare il rendimento e accelerare la consegna del valore - o in altre parole, implementare il "flusso". Quando si applica questo allo sviluppo del software, ciò significa limitare la quantità di lavoro sovrapposto, la complessità di ogni elemento di lavoro e la quantità totale di lavoro affrontato in un dato momento. Piccole dimensioni dei lotti permettono una costante convalida che il lavoro sta andando nella giusta direzione.

Principio 7: Applicare la cadenza, sincronizzare con la pianificazione del dominio trasversale

I team agili applicano naturalmente la cadenza attraverso gli sprint o le iterazioni. Creare una cadenza per tutte le questioni possibili riduce la complessità, affronta l'incertezza, costruisce la memoria muscolare, impone la qualità e infonde la collaborazione. Sincronizzare queste cadenze permette alle persone e alle attività di muoversi come ingranaggi nella ruota dove le informazioni apprese supportano le decisioni e la pianificazione incrementale.

Principio 8: Sbloccare la motivazione intrinseca dei lavoratori della conoscenza

Si tratta di liberare il potenziale dei team e di aiutare la leadership a prendere la prospettiva di allenare e servire i loro team piuttosto che una mentalità di comando e controllo.

Principio 9: Decentrare il processo decisionale

Ridurre la lunghezza delle code e adottare un approccio economico decentralizzando il processo decisionale, dà ai team l'autonomia di cui hanno bisogno per portare a termine il lavoro. I leader dovrebbero conservare la loro autorità decisionale per gli argomenti di importanza strategica e permettere ai team di fare scelte informate su tutto il resto.

Come funziona SAFe?

Le organizzazioni che sono pronte ad implementare SAFe di solito hanno una sponsorizzazione a livello esecutivo, una forte propensione al cambiamento e una base in scrum.

Scaled Agile, Inc. fornisce una roadmap di implementazione SAFe che contiene passi dettagliati su come iniziare e impostare l'organizzazione per un'adozione diffusa in tutti i portafogli. I 12 passi per implementare SAFe includono:

1. Raggiungere il punto di svolta
2. Formare agenti di cambiamento lean-agile
3. Formare dirigenti, manager e leader
4. Creare un centro di eccellenza lean-agile
5. Identificare i flussi di valore e gli ART (Agile Release Trains)
6. Creare il piano di implementazione
7. Preparare il lancio ART
8. Formare le squadre e lanciare l'ART
9. Allenare l'esecuzione dell'ART
10. Lanciare più ART e flussi di valore
11. Estendere al portafoglio
12. Sostenere e migliorare

3.5 Scrum vs Business Agility nello sviluppo del software

Agile e DevOps

La mancanza di definizione per “DevOps” ha portato ad alcune confusioni generali. Non si può negare la connessione storica tra “DevOps” e Agile. Quando Patrick DuBois e Andrew Clay Schafer hanno cercato di connettersi alla conferenza Agile 2008 su "Agile Infrastructure", è nata la connessione con “DevOps”. Anche se Patrick ha poi coniato il termine "DevOps", la conferenza Agile continua ad onorare questa connessione con la prima traccia di “DevOps”.

Pianificare il lavoro non pianificato

Nella comunità DevOps, quelli con esperienza Agile riconoscono che Scrum è utile per tracciare il lavoro pianificato. Alcuni lavori nelle operazioni possono essere pianificati. Ma gran parte del lavoro delle operazioni non è pianificato: picchi di prestazioni, interruzioni di sistema e sicurezza compromessa. Questi eventi richiedono una risposta immediata. Non c'è tempo per aspettare che gli elementi siano prioritari in un backlog o per la prossima sessione di pianificazione dello sprint. Per questo motivo, molti team che hanno abbracciato il pensiero DevOps, guardano oltre scrum a kanban. Questo li aiuta a tracciare entrambi i tipi di lavoro, e li aiuta a capire l'interazione tra loro. Oppure, adottano un approccio ibrido, spesso chiamato scrumban o kanplan (kanban con un backlog). In molti modi, la chiave per l'ampia adozione di scrum può essere che non prescrive pratiche tecniche. Le pratiche di gestione leggera di Scrum spesso fanno una grande differenza per un team. Dove una volta c'erano priorità in competizione da più fonti, ora c'è un unico insieme di priorità nel backlog. E dove una volta c'era troppo lavoro in corso, ora c'è un piano limitato da ciò che il tempo ha dimostrato essere realmente possibile. In combinazione, questi possono portare un team a nuovi livelli di produttività. Tuttavia, il team può trovarsi limitato dalla mancanza di pratiche tecniche, come le revisioni di codifica, i test di accettazione automatici e l'integrazione continua. Altri processi Agili, come l'“Extreme Programming”, si basano sulle “best practice” che

supportano la capacità del team di mantenere un ritmo sostenibile e di fornire trasparenza e visibilità al management e alle parti interessate.

Product owner e Service owner

Da studi risulta che aiuta avere due ruoli diversi per i prodotti che bisogna gestire. I "Product Owner" sono bravi a capire le caratteristiche di cui gli utenti hanno bisogno, ma non sono così bravi a pesare queste caratteristiche rispetto alle caratteristiche non funzionali come le prestazioni, l'affidabilità e la sicurezza. Questo approccio "a due proprietari" non è l'unico percorso verso DevOps. Ciò che è importante è capire queste caratteristiche non funzionali come "caratterizzanti" ed essere in grado di pianificare e dare priorità ad esse proprio come qualsiasi user story funzionale. Come tutti i metodi Agile, Scrum ha un meccanismo incluso di "miglioramento del processo" chiamato "retrospective". Quindi, è ragionevole credere che alcuni team Scrum attingeranno a DevOps come fonte di ispirazione e useranno le "retrospective" di Scrum come opportunità per sintonizzarsi e adattarsi a DevOps. Tuttavia, è pratico rendersi conto che la maggior parte dei team ha bisogno di un'iniezione di idee esterne.

DevOps e Continuous Delivery

Quando è fatto bene, la disciplina del "Continuous Delivery" (CD) aiuta a limitare il lavoro in corso, mentre l'automazione del "deployment" aiuta ad elevare i vincoli. In questo modo, il CD aiuta un team di software a consegnare più frequentemente e con maggiore qualità, invece di dover scegliere tra i due. Tuttavia, proprio come i team che si concentrano solo su Scrum possono perdere il contesto più ampio di Agile, così anche i team che si concentrano su "Continuous Delivery" possono perdere il contesto più ampio di DevOps. La pratica del CD non affronta direttamente i problemi di comunicazione tra il business e un team di software. Se il business ha un ciclo di pianificazione lungo un anno, guidato dal budget, allora un team che consegna ogni commit in produzione potrebbe dover aspettare mesi prima che il business possa reagire. Troppo spesso queste reazioni arrivano cancellando il progetto, o peggio raddoppiando il team del progetto. L'Agile Fluency Model indica il primo livello di fluidità come "Focus on Value", dove i team si concentrano sulla trasparenza e l'allineamento. Senza questa fluidità, il CD può facilmente evolvere in

un ciclo infinito di miglioramento tecnico che non produce alcun valore apprezzabile per il business. Un team potrebbe diventare bravo a consegnare velocemente con alta qualità, ma per un prodotto che ha un basso valore per gli utenti finali o per il business. Anche quando ci sono molti utenti che dicono cose buone, quella valutazione di basso valore potrebbe essere possibile solo ad un livello di portafoglio aziendale più ampio.

I tre modi

DevOps è più che automatizzare la pipeline di "deployment". Nelle parole di John Allspaw, DevOps riguarda: "… gli operativi (ops) che pensano come gli sviluppatori (devs). Sviluppatori che pensano come gli operativi". Elaborando questo pensiero, Gene Kim spiega i tre principi di DevOps:

The first way	System Thinking	emphasizes the performance of the entire system, as opposed to the performance of a specific silo of work or department – this can be as large as a division or as small as an individual contributor.
The second way	Amplify Feedback Loops	creating the right to left feedback loops. The goal of almost any process improvement initiative is to shorten and amplify feedback loops so necessary corrections can be continually made.
The third way	Culture of Continual Experimentation and Learning	creating a culture that fosters two things: continual experimentation, taking risks and learning from failure; and understanding that repetition and practice is the prerequisite to excellence.

FIGURA 2 - PRINCIPI DI DEVOPS

La Prima Via: automatizzare il flusso da dev a ops. L'automazione gioca un ruolo ovvio nell'aiutare ad accelerare un sistema di deployment. Ma c'è più di una semplice automazione nel pensiero di sistema.

La Seconda Via è caratterizzata dalla pratica, "Anche i Devs indossano i cerca-persone". Anche se l'uso letterale dei cercapersone può non essere necessario, significa coinvolgere gli sviluppatori in questioni operative. Questo aiuta gli sviluppatori a capire le conseguenze delle loro scelte di sviluppo. Non si tratta solo di consapevolezza. Gli sviluppatori portano anche la loro comprensione interna del sistema agli sforzi di risoluzione dei problemi, così una risoluzione può essere trovata e implementata più velocemente.

La Terza Via riguarda il fare esperimenti incrementali nel sistema nel suo complesso, non solo come modifiche all'applicazione che si muove attraverso la pipeline. In altre parole, è relativamente facile vedere quanto tempo richiede l'automazione e usare un'infrastruttura sempre più potente per continuare a migliorarla. È più difficile capire come i passaggi di mano tra ruoli o organizzazioni introducano ritardi. Questo significa che i team "ispezionano e adattano" l'intero flusso di lavoro di consegna, cercando opportunità per migliorare la collaborazione umana. Del resto, il CD richiede l'abitudine di adattarsi e migliorare. Se il team non riflette su come diventare più efficace, e quindi sintonizzare e regolare il suo comportamento su qualsiasi altra cosa, allora neanche il CD crescerà e prospererà. Il team deve sentirsi autorizzato a risolvere i propri problemi.

DevOps è Agile applicato oltre il team di software

Scrum corrisponde principalmente al principio Agile, "accogliere i requisiti che cambiano, anche in ritardo nello sviluppo". I processi agili sfruttano il cambiamento per il vantaggio competitivo del cliente. La consegna continua corrisponde principalmente al principio Agile: "La nostra massima priorità è quella di soddisfare il cliente attraverso la consegna rapida e continua di software di valore". Questo significa che Agile è rivolto più all'abbracciare il cambiamento in entrata e in uscita che sulle cerimonie come gli stand-up e la pianificazione degli sprint.

Infatti, ci sono 10 principi nel Manifesto Agile ma piuttosto che cercare di scegliere tra i principi, dovrebbero essere considerati nel loro insieme. Insieme questi principi rappresentano un atteggiamento verso il cambiamento che è comune sia per Agile che per DevOps. Queste persone sono state bloccate cercando di gestire sistemi fragili che sono anche i più importanti per il business, dove sono necessari i cambiamenti più urgenti. Come tale, questa idea Agile di abbracciare il cambiamento non è "cambiamento per il gusto di cambiare". Si tratta di ritenere lo sviluppo responsabile della qualità dei loro cambiamenti, mentre si migliora la capacità complessiva di fornire valore al business. Questo focus sul valore di business è un altro aspetto condiviso da Agile e DevOps. Infine, né Agile né DevOps sono obiettivi di business di per sé. Entrambi sono movimenti culturali che possono ispirare la

vostra organizzazione con mezzi migliori per raggiungere i vostri obiettivi. Agile e DevOps funzionano meglio in combinazione, piuttosto che come avversari

Business Agility

DevOps oggi è nel mondo dello sviluppo software il modello più vicino al concetto di Business Agility. Va sottolineato che Business Agility significa integrare l'eccellenza operativa con i flussi di informazioni che scambiamo con i nostri clienti, adattando costantemente le "operations" alla conoscenza che sviluppiamo dei bisogni dei clienti attraverso i dati. Sebbene la definizione tradizionale dice che la Business Agility è la capacità di cambiare rapidamente strategia, struttura, prodotto, competenze, ruoli, processi, questa definizione dice poco su come ciò accada. Mentre comprendere come funziona l'organizzazione come sistema e come il sistema risponde ai bisogni del cliente finale ci permette di capire che questo può accadere solo attraverso un utilizzo consapevole dei dati di impatto che diventano i veri "drivers" delle azioni dei teams e che tutto ciò accade in tempi molto rapidi attraverso un ciclo continuo che si configura in questo modo:

FIGURA 3 - CICLO CONTINUO BUSINESS AGILITY

Cos'è Scrum?

Scrum è una struttura che aiuta i team a lavorare insieme. Scrum incoraggia i team ad imparare attraverso le esperienze, ad auto-organizzarsi mentre lavorano alla risoluzione di un problema, e a riflettere sulle loro vittorie e sconfitte per migliorare continuamente.

Mentre lo Scrum di cui si parla è più frequentemente usato dai team di sviluppo software, i suoi principi e le sue lezioni possono essere applicati a tutti i tipi di lavoro di squadra. Questo è uno dei motivi per cui Scrum è così popolare. Spesso pensato come una struttura agile di gestione dei progetti, Scrum descrive un insieme di riunioni, strumenti e ruoli che lavorano insieme per aiutare i team a strutturare e gestire il loro lavoro.

Cosa sono gli sprint?

Uno sprint è un breve periodo di tempo in cui un team scrum lavora per completare una certa quantità di lavoro. Gli sprint sono il cuore della metodologia Scrum e delle metodologie agili, e sfruttare gli sprint nel modo giusto aiuterà il vostro team agile a sviluppare software in modo migliore e con meno lavoro. Molti associano gli sprint di scrum allo sviluppo agile del software, tanto che spesso si pensa che scrum e agile siano la stessa cosa, ma non lo sono. Agile è un insieme di principi e Scrum è una struttura per ottenere risultati.

Le molte somiglianze tra i valori agili e i processi di scrum portano ad una giusta associazione. Gli sprint aiutano i team a seguire il principio agile di "consegnare frequentemente software funzionante", così come a vivere il valore agile di "rispondere al cambiamento piuttosto che seguire un piano". I valori Scrum di trasparenza, ispezione e adattamento sono complementari all'Agile e centrali al concetto di sprint.

Come pianificare ed eseguire gli sprint di scrum

La pianificazione dello sprint è un evento collaborativo in cui il team risponde a due domande fondamentali: quale lavoro può essere fatto in questo sprint e come verrà fatto il lavoro scelto? Scegliere i giusti elementi di lavoro per uno sprint è uno sforzo collaborativo tra il "product owner", lo scrum master e il team di sviluppo. Il product owner discute l'obiettivo che lo sprint dovrebbe raggiungere e gli elementi del product backlog che, una volta completati, farebbero raggiungere l'obiettivo dello sprint. Il team crea quindi un piano per come costruirà gli elementi del backlog e li porterà a termine prima della fine dello sprint. Gli elementi di lavoro scelti e il piano su come farli sono chiamati sprint backlog. Alla fine della pianificazione dello sprint

il team è pronto per iniziare a lavorare sul backlog dello sprint, cambiando lo stato degli elementi del backlog a "In-progress" e "Done". Durante uno sprint, il team controlla come sta procedendo il lavoro. L'obiettivo di questa riunione è di far emergere qualsiasi blocco e sfida che potrebbe avere un impatto sulla capacità del team di consegnare l'obiettivo dello sprint.

Dopo uno sprint, il team dimostra ciò che ha completato durante la revisione dello sprint. Questa è l'opportunità per il tuo team di mostrare il proprio lavoro ale parti interessate e ai compagni di squadra prima che vada in produzione e di identificare le aree di miglioramento per il prossimo sprint.

Fare e non fare

Anche padroneggiando le basi teoriche, la maggior parte dei team inciamperà quando inizieranno a gestire gli sprint.

Fare:

- Assicurarsi che il team stabilisca e capisca l'obiettivo dello sprint e come sarà misurato il successo. Questa è la chiave per mantenere tutti allineati e andare avanti
- Assicuratevi di avere un backlog ben curato con le vostre priorità e dipendenze in ordine. Questo può essere una grande sfida che potrebbe far deragliare il processo se non è gestito correttamente.
- Assicuratevi di avere una buona comprensione della velocità, e che rifletta cose come i congedi e le riunioni del team.
- Usate la riunione di Sprint Planning per definire i dettagli del lavoro che deve essere fatto. Incoraggiare i membri del team ad abbozzare i compiti per tutte le storie, i bug e i compiti che arrivano nello sprint.
- Lasciate fuori il lavoro dove non sarete in grado di ottenere le dipendenze, come il lavoro di un altro team, i progetti e l'approvazione legale.
- Infine, una volta presa una decisione o un piano, assicurarsi che qualcuno registri queste informazioni nello strumento di gestione del progetto o di collaborazione, come i ticket Jira. In questo modo, sia la decisione che la logica sono facili da vedere per tutti in seguito.

Non fare:

- Non inserire troppe storie, sovrastimare la velocità, o inserire compiti che non possono essere completati nello sprint.
- Non dimenticare la qualità o il debito tecnico. Assicuratevi di mettere in bilancio il tempo per il QA (Quality Assurance) e per i “task” non funzionali, come i bugs.
- Non lasciare che la squadra abbia una visione confusa di ciò che è nello sprint. Assicurarsi che tutti si stiano muovendo nella stessa direzione.
- Inoltre, non assumere una grande quantità di lavoro sconosciuto o ad alto rischio. Spezzare le storie troppo grandi o con un'alta incertezza, e non aver paura di lasciare un po' di quel lavoro per il prossimo sprint.
- Se senti delle preoccupazioni dal team, sia che si tratti di velocità, di lavoro a bassa incertezza, o di lavoro che pensano sia più grande di quello che hanno stimato, non ignorarlo. Affrontare il problema e ricalibrare quando necessario.
- Ottimizzare gli sprint con l'automazione

Pianificazione dello sprint

La pianificazione dello sprint è un evento in scrum che definisce cosa può essere consegnato nel prossimo sprint e come questo lavoro sarà realizzato. La pianificazione dello sprint è un evento in Scrum che dà il via allo sprint. Lo scopo della pianificazione dello sprint è di definire cosa può essere consegnato nello sprint e come questo lavoro sarà realizzato. La pianificazione dello sprint è fatta in collaborazione con l'intero team scrum.

Prima di passare all'azione bisogna impostare lo sprint. Decidere quanto sarà lungo lo sprint, l'obiettivo dello sprint e dove cominciare. La sessione di pianificazione dello sprint dà il via allo sprint impostando l'agenda e il focus. Se fatta correttamente, crea anche un ambiente dove il team è motivato, sfidato e può avere successo. Cattivi piani di sprint possono far deragliare il team impostando aspettative irrealistiche.

Cosa - Il ‘product owner’ descrive l'obiettivo (o scopo) dello sprint e quali elementi del backlog contribuiscono a quell'obiettivo. Lo scrum team decide cosa può essere fatto nello sprint successivo e cosa faranno durante lo sprint per farlo accadere.

Come - Il team di sviluppo pianifica il lavoro necessario per raggiungere l'obiettivo dello sprint. In definitiva, il piano di sprint risultante è una negoziazione tra il team di sviluppo e il ‘product owner’, basato sul valore e sullo sforzo.

Chi - Non si può fare la pianificazione dello sprint senza il ‘product owner’ o il team di sviluppo. Il ‘product owner’ definisce l'obiettivo basato sul valore che cerca. Il team di sviluppo ha bisogno di capire come possono o non possono raggiungere quell'obiettivo. Se uno dei due manca da questo evento, la pianificazione dello sprint diventa quasi impossibile.

Input - Un ottimo punto di partenza per il piano dello sprint è il product backlog in quanto fornisce una lista di “cose” che potrebbero potenzialmente essere parte dello sprint corrente. Il team dovrebbe anche guardare il lavoro esistente fatto nell'incremento e avere una visione della capacità.

Output - Il risultato più importante per la riunione di pianificazione dello sprint è che il team può descrivere l'obiettivo dello sprint e come inizierà a lavorare verso quell'obiettivo. Questo è reso visibile nello sprint backlog.

Preparazione per la riunione di pianificazione sprint

Gestire un grande evento di pianificazione dello sprint richiede un po' di disciplina. Il ‘product owner’ deve essere preparato, combinando i feedback della precedente revisione dello sprint, il feedback delle parti interessate e la sua visione del prodotto, in modo da preparare la scena per lo sprint. Per la trasparenza, il product backlog dovrebbe essere aggiornato e raffinato per fornire chiarezza. La rifinitura del backlog è un evento opzionale in Scrum, perché alcuni backlog non ne hanno bisogno. Tuttavia, per la maggior parte dei team, è meglio riunire il team per rivedere e raffinare il backlog prima della pianificazione dello sprint.

Fissare un limite di tempo per la pianificazione dello sprint

La pianificazione dello sprint dovrebbe essere limitata a non più di due ore per ogni settimana dello sprint. Così, per esempio, la riunione di pianificazione dello sprint per uno sprint di due settimane non dovrebbe essere più lunga di due ore. Questo è chiamato "timeboxing", o stabilire una quantità massima di tempo per il team per realizzare un compito, in questo caso, la pianificazione dello sprint. Lo scrum master ha la responsabilità di assicurarsi che la riunione avvenga: il timebox è compreso. Se il team è contento prima che il timebox sia finito, allora l'evento è finito. Un timebox è un tempo massimo consentito; non c'è un tempo minimo consentito.

Concentrarsi sui risultati, non sul lavoro

Durante la pianificazione dello sprint è facile rimanere “impantanati” nel lavoro concentrandosi su quale compito dovrebbe venire per primo, chi dovrebbe farlo, e quanto tempo ci vorrà. Per un lavoro complesso, il livello di informazione che si conosce all'inizio può essere basso, e gran parte di esso è basato su supposizioni. Scrum è un processo empirico, il che significa che non si può pianificare in anticipo, ma piuttosto imparare facendo, e poi alimentare queste informazioni nel processo.

Le stime sono necessarie, ma non fingere di sapere più di quanto non si sappia

La pianificazione dello sprint richiede un certo livello di stima. Il team ha bisogno di definire cosa può o non può essere fatto nello sprint: sforzo stimato vs capacità. La stima è spesso confusa con gli impegni. Le stime sono per loro natura previsioni basate sulla conoscenza a disposizione. Più ci sono incognite, meno è probabile che la stima sia corretta. Una buona stima richiede un ambiente basato sulla fiducia dove le informazioni sono date liberamente e le ipotesi sono discusse nella ricerca dell'apprendimento e del miglioramento. Se le stime sono usate in modo negativo e conflittuale dopo che il lavoro è stato completato, allora è probabile che le stime future saranno o molto più grandi per assicurarsi che non siano mai più sbagliate o il tempo impiegato per crearle sarà molto più lungo mentre il team ripensa, preoccupandosi, alle implicazioni di sbagliare.

Cos'è un product backlog?

Un product backlog è una lista prioritaria di lavoro per il team di sviluppo che deriva dalla roadmap e dai suoi requisiti. Gli elementi più importanti sono mostrati in cima

al product backlog in modo che il team sappia cosa consegnare per primo. Il team di sviluppo non lavora attraverso il backlog al ritmo del 'product owner' e il 'product owner' non spinge il lavoro al team di sviluppo. Invece, il team di sviluppo conduce il lavoro dal backlog del prodotto con capacità che riesce a mettere in campo, sia continuamente (kanban) o per iterazione (scrum).

Iniziare con le due "R"

La roadmap e i requisiti di un team forniscono la base per il backlog del prodotto. Le iniziative della roadmap si suddividono in diverse epics, e ogni epic avrà diversi requisiti e user stories. Per fare un esempio pratico, guardiamo la roadmap per un prodotto fittizio chiamato Teams in Space.

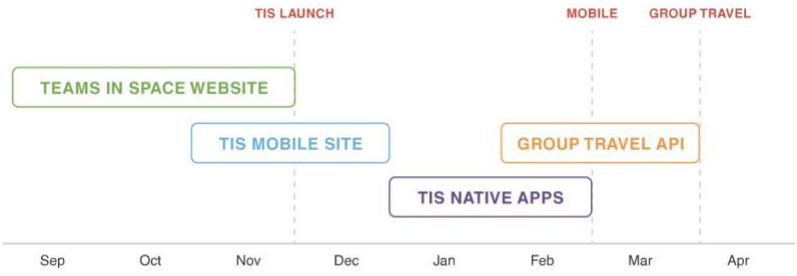

FIGURA 4 - ESEMPIO DI ROADAMP PER UN PRODOTTO FITTIZIO

Poiché il sito web di Teams in Space è la prima iniziativa nella roadmap, vorremo suddividere questa iniziativa in epiche e in user stories per ognuna di queste epiche.

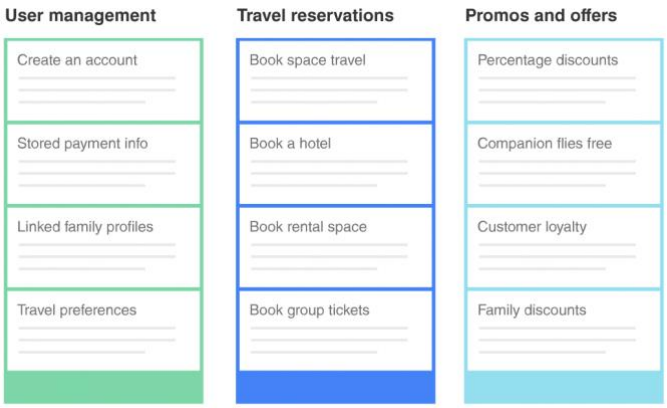

FIGURA 5 - ESEMPIO DI EPICS IN CUI È DIVISA L'INIZIATIVA

Il ‘product owner’ organizza ogni user stories in una singola lista per il team di sviluppo. Il ‘product owner’ può scegliere di consegnare prima un'epica completa (a sinistra). Oppure, potrebbe essere più importante per il programma testare la prenotazione di un volo scontato che richiede storie da diverse epiche (destra).

Cosa può influenzare la prioritizzazione del ‘product owner’? Priorità del cliente, urgenza di ottenere un feedback, difficoltà relativa di implementazione, relazioni simbiotiche tra gli elementi di lavoro. Quindi il ‘product owner’ ha il compito di dare priorità al backlog. I product owner efficaci cercano input e feedback dai clienti, dai designer e dal team di sviluppo per ottimizzare il carico di lavoro di tutti e la consegna del prodotto.

Mantenere il backlog in salute

Una volta che il backlog del prodotto è costruito, è importante mantenerlo regolarmente per tenere il passo con il programma. I product owner dei prodotti dovrebbero rivedere il backlog prima di ogni riunione di pianificazione dell'iterazione per assicurare che la priorità sia corretta e che il feedback dell'ultima iterazione sia stato considerato. La revisione regolare del backlog è spesso chiamata “backlog grooming” nei team agili.

Una volta che il backlog diventa più grande, i product owner hanno bisogno di raggruppare il backlog in elementi a breve termine e a lungo termine. Gli elementi a breve termine hanno bisogno di essere completamente sviluppati prima di essere etichettati come tali. Questo significa che le user stories complete sono state progettate, la collaborazione con il design e lo sviluppo è stata risolta, e le stime dallo sviluppo sono state fatte. Gli elementi a lungo termine possono rimanere un po' vaghi, anche se è una buona idea ottenere una stima approssimativa dal team di sviluppo per aiutarli a stabilire le priorità. La parola chiave qui è "approssimativa": le stime cambieranno una volta che il team capirà completamente e inizierà a lavorare su questi elementi a lungo termine.

Il backlog serve come collegamento tra il ‘product owner’ e il team di sviluppo. Il ‘product owner’ è libero di ridefinire le priorità del lavoro nel backlog in qualsiasi

momento a causa del feedback dei clienti, della rifinitura delle stime e dei nuovi requisiti.

Come i backlog dei prodotti mantengono il team agile?

I product owner più avveduti curano rigorosamente il product backlog del loro programma, rendendolo un contorno affidabile e condivisibile degli elementi di lavoro per un progetto. Le parti interessate metteranno in discussione le priorità, e questo è un bene. Favorire la discussione su ciò che è importante fa sì che le priorità di tutti siano sincronizzate. Queste discussioni favoriscono una cultura di prioritizzazione di gruppo assicurando che tutti condividano la stessa mentalità sul programma.

Il product backlog serve anche come base per la pianificazione dell'iterazione. Tutti gli elementi di lavoro dovrebbero essere inclusi nel backlog: user stories, bugs, modifiche al design, debito tecnico, richieste dei clienti, elementi di azione dalla 'retrospective', ecc. Questo assicura che gli elementi di lavoro di tutti siano inclusi nella discussione generale per ogni iterazione. I membri del team possono quindi fare dei compromessi con il 'product owner' prima di iniziare un'iterazione con una conoscenza completa di tutto ciò che deve essere fatto.

Revisioni di sprint agili

Le sprint review non sono "retrospective". Una sprint review serve a dimostrare il duro lavoro dell'intero team. I membri del team si riuniscono per delle dimostrazioni informali e descrivono il lavoro che hanno fatto per quell'iterazione. È un momento per fare domande, provare nuove funzionalità e dare feedback. Tre passi per migliorare le revisioni di sprint con il tuo team agile.

Passo 1: "definition of done"

Tagliare il traguardo e completare il lavoro richiede una buona pianificazione, una chiara definizione di "Done" e un'esecuzione mirata. La maggior parte di questo avviene durante la pianificazione dello sprint, ma per avere uno sprint review di successo, i team devono fare un po' più che pianificare. Hanno bisogno di sviluppare una chiara cultura su come consegnare il lavoro e su cosa significa "essere fatto". I

team efficaci conducono processi chiari e una cultura di sviluppo per ogni elemento di lavoro. La cultura del team intorno alla qualità e al completamento dovrebbe elevarsi al di sopra di ogni storia utente, elemento di progettazione e bug. Questa cultura riflette il modo in cui il team si avvicina e consegna il software.

Una chiara definizione di "Done" aiuta i team a concentrarsi sull'obiettivo finale per ogni elemento di lavoro. Quando il product owner aggiunge lavoro al backlog del team, definire i criteri di accettazione è una parte fondamentale del suo processo. In Jira, il team tiene traccia dei criteri di accettazione e delle note di test in linea con il resto della storia utente all'interno di Jira. In questo modo, l'intero team ha una chiara visione del successo di ogni questione. Criteri di accettazione sono metriche che il 'product owner' usa per confermare che la storia è stata implementata in modo soddisfacente. Note di test sono una guida breve e mirata del team di assistenza alla qualità che permette all'ingegnere di sviluppo di scrivere meglio il codice delle funzionalità e i test automatici.

Passo 2: celebrare la squadra

Le revisioni di sprint sono un ottimo momento per celebrare il team e i risultati di ognuno durante un'iterazione. Gli sprint review non sono sinonimo di "retrospective", quindi assicuratevi di ospitare la sprint review dopo un'iterazione, ma prima della vostra "retrospective". I partecipanti esterni sono sempre i benvenuti, ma la riunione di solito consiste nel 'product owner', l'intero team di sviluppo e lo scrum master. Gli sprint review proteggono la salute e il morale del team, sono tutte incentrate sulla costruzione della squadra. La revisione non è conflittuale, è un evento collaborativo attraverso cui le persone mostrano il loro lavoro, fanno domande e ricevono feedback.

Passo 3: raggiungere le aree geografiche

Le aziende con team distribuiti hanno sfide speciali per quanto riguarda lo scaling delle cerimonie agili nelle varie aree geografiche. Le revisioni di sprint non fanno eccezione. Anche se sono distribuiti, gli sprint review sono una parte importante della cultura del team. I membri del team creano video informali e li condividono su una pagina di Confluence affinché tutto il team li veda. Questi video informali

tengono tutti aggiornati sui progressi dello sviluppo nonostante le differenze di orario. Vedere una demo di una “feature” in prima persona dallo sviluppatore rafforza il team in due modi:

1. Comprensione del prodotto: l'intero team sente l'intenzione, la logica e l'implementazione della feature. Amplia la comprensione dell'intero prodotto da parte di tutti.
2. Team Building: i video creano connessioni più personali in tutto il team. Ognuno di noi può vedere chi c'è dietro ogni aspetto di un prodotto. I ponti creati da questa pratica ci rendono un gruppo più stretto e coeso nonostante le geografie.

Stand-up per team agili aiutano a scoprire i blocchi e a rafforzare il team agile.

Lo stand-up è comunemente noto come scrum quotidiano, e rafforza il "noi" per mantenere tutti consapevoli del panorama e dei progressi della squadra. Detto in un altro modo, uno stand-up è una riunione giornaliera che coinvolge il core team: product owner, sviluppatori e lo scrum master. Inoltre, rafforza il team quando tutti condividono i progressi che stanno contribuendo al team. Il rinforzo quotidiano della condivisione dei successi e dei piani individuali mantiene tutti entusiasti del contributo complessivo della squadra all'organizzazione.

A livello individuale, è importante arrivare allo stand-up del giorno sapendo cosa dire. Mantiene alta l'energia dello stand-up e tutti sono impegnati.

Cos'è uno scrum master?

Lo scrum master assicura che il framework di scrum sia seguito. Scrum ha un insieme chiaramente definito di ruoli e rituali che dovrebbero essere seguiti e lo scrum master lavora con ogni membro dello scrum team per guidare e allenare la squadra attraverso il framework scrum. Gli scrum master sono i facilitatori di scrum, il framework agile leggero, con un focus sulle iterazioni a tempo chiamate sprint. Come facilitatori, gli scrum master agiscono come allenatori per il resto del team. “Leader servitori”, come dice la Guida Scrum. I buoni scrum master sono impegnati nella fondazione e nei valori di Scrum, ma rimangono flessibili e aperti alle opportunità per il team di migliorare il loro flusso di lavoro.

Lo scrum master: Tenere tutto insieme

Lo scrum master è il ruolo responsabile di incollare tutto insieme e assicurare che lo scrum sia seguito bene. In termini pratici, questo significa che aiuta il 'product owner' a definire il valore, il team di sviluppo a fornire il valore e il team di scrum a migliorare. Lo scrum master è un "servant leader" che non solo descrive uno stile di leadership di supporto, ma descrive ciò che fa quotidianamente.

E' al servizio del 'product owner' aiutandolo a capire meglio e a comunicare il valore, a gestire il backlog, ad aiutarlo a pianificare il lavoro con il team e a scomporre quel lavoro per fornire l'apprendimento più efficace.

Al servizio del team di sviluppo, lo scrum master lo aiuta ad auto-organizzarsi, a concentrarsi sui risultati, a raggiungere un "incremento fatto" e a gestire i blocchi.

Lo scrum master è al servizio anche dell'organizzazione in generale, aiutandola a capire cos'è lo scrum e a creare un ambiente che supporti lo scrum.

Lo scrum master si concentra su:

Trasparenza - Per ispezionare e adattare efficacemente è importante che le persone giuste possano vedere cosa sta succedendo. Ma questo è in realtà molto più difficile di quanto sembri. Lo scrum master ha il compito di assicurare che lo scrum team lavori in modo trasparente. Gli esempi includono la creazione di mappe delle storie e l'aggiornamento delle pagine di Confluence con le idee "retrospective".

Empirismo - Fondamentale per gli approcci scrum e agile è l'idea che il miglior modo di pianificare è fare il lavoro e imparare da esso. Il processo empirico non è facile e richiede che lo scrum master alleni il team scrum a scomporre il lavoro, descrivere i risultati chiari e rivedere quei risultati.

Auto-organizzazione - Dire ad un team di sviluppo che possono auto-organizzarsi non significa che il team si auto-organizzerà. Infatti l'auto-organizzazione arriva col tempo e richiede aiuto e supporto. Lo scrum master incoraggerà i membri del team ad uscire dalla loro zona di comfort e a provare cose diverse e ad esporre e sfidare idee predefinite sui confini dei ruoli e sulle responsabilità.

Valori - Scrum definisce 5 valori di coraggio, concentrazione, impegno, rispetto e apertura non perché sono belli da avere, ma perché creano un ambiente di sicurezza fisiologica e fiducia. Questo ambiente è necessario per far prosperare l'agilità. Seguire i valori è responsabilità di tutti nel team di scrum, ma lo scrum master ha un ruolo attivo nell'incoraggiare e ricordare a tutti l'importanza di questi valori.

Lo scrum master serve il 'product owner' nella pianificazione dello sprint e nelle revisioni dello sprint assicurando che il valore sia chiaramente descritto e la direzione stabilita. Serve il team di sviluppo nello scrum quotidiano assicurando che il lavoro stia avvenendo e che i blocchi vengano rimossi. Si assumono anche la responsabilità dei blocchi che sono al di fuori della capacità del team di risolvere. Lo scrum master assicura che ogni opportunità di miglioramento sia resa trasparente allo scrum team e che la 'retrospective' abbia una chiara serie di risultati che possono essere eseguiti.

Responsabilità dello scrum master

Nel mondo agile ideale, il team gestirebbe il proprio processo e il proprio strumento. Eppure, si è scoperto che molti team che fanno il salto all'agile spesso si affidano allo scrum master come proprietario del loro processo. Gli scrum master spesso eseguono alcuni o tutti fra i seguenti compiti:

1. Standup - Facilitare gli standup quotidiani (o lo scrum quotidiano) come necessario.
2. Riunioni di pianificazione di iterazione/sprint - Proteggere il team dal sovraccarico di impegni. Aiutare nella stima e nella creazione di sotto-compiti.
3. Revisioni di sprint - Partecipa alla riunione e cattura il feedback.
4. Retrospettive - Annota le aree di miglioramento e gli elementi di azione per gli sprint futuri.
5. Amministrazione della scheda - Lavora come amministratore della scheda scrum. Assicurarsi che le schede siano aggiornate e che lo strumento di scrum, software Jira o altro, funzioni bene.
6. 1 a 1 - Incontra individualmente i membri del team e le parti interessate quando necessario. Appiana i disaccordi del team sul processo e sugli stili di

lavoro. Mentre molti praticanti di scrum sono anti-1a1, poiché credono che queste comunicazioni dovrebbero avvenire durante gli standup, alcuni team, in particolare quelli nuovi, preferiscono avere queste interazioni regolari faccia a faccia con specifici membri del team. Lo scrum master può decidere che queste interazioni individuali sono cruciali per lo sviluppo del team e per conoscersi a vicenda.

7. Consulenza interna - Gli scrum master dovrebbero essere preparati a dare consulenza ai membri del team e le parti interessate interni su come lavorare al meglio con lo scrum team.
8. Reporting - Analisi regolare dei grafici di burndown e di altri strumenti di pianificazione del portfolio per capire cosa viene costruito e con quale cadenza.
9. Blocchi - Lo scrum master aiuta il team eliminando i blocchi esterni e gestendo i blocchi interni attraverso miglioramenti del processo o del flusso di lavoro.
10. Lavoro impegnativo - Se il team di scrum non sta ronzando, questo è un problema dello scrum master. Forse questo significa aggiustare i computer rotti, spostare le scrivanie, o anche regolare il termostato. Gli scrum master dovrebbero essere a loro agio nel fare qualsiasi cosa per aiutare il loro team e non dovrebbero evitare di prendere un caffè o qualche spuntino se è quello di cui il team ha veramente bisogno.

Ho bisogno di uno scrum master?

Qualsiasi formatore di scrum insegnerà che un team di scrum deve avere uno scrum master. Quando si inizia con lo scrum, può essere di grande aiuto avere qualcuno nel ruolo che ha visto lo scrum funzionare prima. Meglio ancora, ha visto molti esempi di funzionamento. Per questo motivo, gli scrum master sono spesso assunti come consulenti, piuttosto che come impiegati a tempo pieno.

Ma ogni team scrum è diverso. Molti team esperti gestiscono le responsabilità elencate sopra come un'unità, e sono orgogliosi e si divertono nella gestione condivisa del processo. Il ruolo di scrum master ruota attraverso il team, con i membri del team che facilitano standup e retrospective a turno.

Sfortunatamente, l'incomprensione del ruolo di scrum master spesso porta i manager esistenti ad assumere che sia il loro ruolo. Per capire meglio perché questo può essere un problema, confrontiamo lo scrum master con i ruoli non scrum che potete già avere nella vostra organizzazione, e perché è importante mantenere il ruolo separato.

Scrum Master vs Product Manager

Più un product manager è coinvolto nel team di sviluppo, meglio è. Questo coinvolgimento dovrebbe essere sulla falsariga di un product owner che sostiene le esigenze del cliente, il "perché" del prodotto. Quando il coinvolgimento si confonde con l'operatività, il "come" per un team, allora c'è un problema. Questo tipo di mentalità di utilizzo tende a nascondere i problemi: difetti, hand-off e incognite. L'intreccio tra scopo e processo tende a bloccare scopo, programma e qualità. Questo è il motivo per cui lo scrum master e il product owner riempiono due bisogni diversi in un team scrum, che sono spesso combinati con la gestione tradizionale del software. Tuttavia, quando sorgono blocchi o cambiamenti, è necessaria una chiara divisione tra la gestione del processo e la direzione del prodotto.

Scrum Master vs Project Manager

La controparte non tecnica (o non agile) dello scrum master è il project manager. Entrambi questi ruoli si concentrano sul "come" portare a termine il lavoro e risolvere i problemi del flusso di lavoro attraverso il processo e la facilitazione. Sia un project manager tradizionale che uno scrum master hanno la responsabilità di aiutare i loro team a portare a termine il lavoro, ma i loro approcci sono molto diversi. Il project manager imposta e tiene traccia delle scadenze e delle 'milestones', riferisce sui progressi e coordina la comunicazione del team. Tuttavia, lo fa da un luogo di controllo, in un ruolo di gestione più tradizionale. Lo scrum master aiuta il team a migliorare e snellire i processi con cui raggiungono i loro obiettivi. Lo fa come membro del team, o collaboratore, idealmente non come qualcuno che ha il controllo. I migliori team scrum sono auto-organizzati, e quindi non reagiscono bene alla gestione dall'alto.

Con uno scrum master che aiuta ogni team a gestire il proprio processo, l'intera organizzazione può realizzare dei seri guadagni. Oltre a spedire valore ai tuoi clienti su base regolare (l'obiettivo principale di scrum), i compagni di squadra e i manager saranno liberi di concentrarsi su ciò che sanno fare meglio. I product manager possono concentrarsi sulla strategia e gli sviluppatori possono scrivere il loro miglior codice.

Retrospettive Agile

Una 'retrospective' è ogni volta che il tuo team riflette sul passato per migliorare il futuro. Perché fare una 'retrospective'? Nel 2001, con un tratto di penna, è nata la 'retrospective' agile. L'ultimo dei dodici principi dello sviluppo agile recita come segue: "A intervalli regolari, il team riflette su come diventare più efficace, quindi sintonizza e regola il suo comportamento di conseguenza". Il manifesto agile è chiaro: per vivere al meglio i valori agili, i team dovrebbero incontrarsi regolarmente per fare il check-in e fare aggiustamenti. Più comunemente, i team di sviluppo applicano questo principio ospitando regolari riunioni "retrospective".

Più recentemente, il concetto di "retrospective" è uscito dai team di sviluppo ed è entrato in tutti gli aspetti del business e del lavoro di squadra. Esistono team di marketing che retroagiscono sulle campagne o team di gestione che retroagiscono su grandi presentazioni. Molti dei concetti fondamentali del manifesto agile sono rafforzati attraverso le riunioni "retrospective", che apportano i seguenti valori: individui e interazioni piuttosto che processi e strumenti: rispondere al cambiamento piuttosto che seguire un piano.

Ruoli di Scrum

Scrum ha tre ruoli: product owner, scrum master e i membri del team di sviluppo. Mentre questo è abbastanza chiaro, cosa fare con i titoli di lavoro esistenti può confondere. Molti team chiedono se devono cambiare i loro titoli quando adottano lo scrum, ma non si deve fare

Ruoli scrum vs titoli di lavoro

I tre ruoli di scrum descrivono le responsabilità chiave per coloro che fanno parte del team di scrum. Non sono titoli di lavoro. Questo significa che qualsiasi titolo di lavoro, anche quelli esistenti, può eseguire uno dei ruoli. Poiché l'essenza di Scrum è l'empirismo, l'auto-organizzazione e il miglioramento continuo, i tre ruoli danno una definizione minima delle responsabilità e dell'affidabilità per permettere ai team di consegnare efficacemente il lavoro. Questo permette ai team di assumersi la responsabilità di come si organizzano e di continuare a migliorarsi.

Costruire un team scrum

Scrum è una struttura su cui i team possono costruire i loro processi. Fornisce la struttura di base per riunioni regolari, artefatti e chi fa cosa. Ciò che non fa è fornire un modello unico per i team in cui lavorare. Per esempio, se il team sta lavorando su un'applicazione assicurativa web, avrà bisogno di persone che conoscono la tecnologia, i sistemi back-end e il dominio del business. Se, d'altra parte, il team stesse lavorando sulla prossima generazione di Donkey Kong, le competenze necessarie sarebbero molto diverse. Includerebbero un grafico, un ingegnere del suono e uno sviluppatore grafico. Poiché i problemi sono diversi, anche le strutture del team e le competenze necessarie sono diverse.

Questo diventa ancora più difficile quanto più complesso è il problema che un team sta cercando di risolvere. Le squadre potrebbero non conoscere le competenze o la quantità di lavoro necessario in anticipo, e hanno bisogno della flessibilità per cambiare rotta una volta che ne sanno di più. Per fornire una certa struttura a questo mondo complesso, mutevole e spesso fastidioso, Scrum dà una struttura leggera con i tre ruoli scrum di membro del team di sviluppo, ‘product owner’ e scrum master.

Il team di sviluppo: ridefinire “sviluppatore”

Il team di sviluppo sono le persone che fanno il lavoro. A prima vista, si può pensare che il “team di sviluppo” significhi ingegneri. Ma questo non è sempre il caso. Il team di sviluppo può essere composto da tutti i tipi di persone, tra cui designer, scrittori, programmatori, ecc.

Il team di sviluppo dovrebbe essere in grado di auto-organizzarsi in modo da poter prendere decisioni per portare a termine il lavoro. Pensate ad un team di sviluppo

come ad un team di supporto alla produzione che viene chiamato durante la notte perché qualcosa è andato storto. Il team di sviluppo, come il team di supporto alla produzione, può prendere decisioni e fornire la correzione/il valore per il problema in questione. L'auto-organizzazione non significa mancare di rispetto all'organizzazione, ma piuttosto dare potere alle persone più vicine al lavoro per fare ciò che è necessario per risolvere il problema.

Il product owner: stabilire una direzione chiara

I team agili sono, per progettazione, flessibili e reattivi, ed è responsabilità del 'product owner' assicurarsi che stiano fornendo il massimo valore. Il business è rappresentato dal 'product owner' che dice allo sviluppo cosa è importante consegnare. La fiducia tra questi due ruoli è cruciale.

Il product owner dovrebbe non solo capire il cliente, ma anche avere una visione del valore che lo scrum team sta fornendo al cliente. Il product owner bilancia anche i bisogni delle altre parti interessate nell'organizzazione. Quindi il product owner deve prendere tutti questi input e dare priorità al lavoro. Questa è probabilmente la loro responsabilità più importante perché priorità conflittuali e direzioni non chiare non solo ridurranno l'efficacia del team, ma potrebbero anche rompere l'importante rapporto di fiducia che il business ha con il team di sviluppo.

I team agili sono progettati per ispezionare e adattarsi, ciò significa che un cambiamento di priorità può portare a un cambiamento massiccio della struttura del team, dei prodotti di lavoro, così come del risultato finale. È quindi cruciale, perché i team scrum abbiano successo, che **solo una persona stabilisca la priorità. Quella persona è il 'product owner'.**

Le responsabilità del product owner:

Gestire il backlog di scrum - Questo non significa che è l'unico ad inserire nuovi "product backlog items" nel "backlog". Ciò significa che il 'product owner' dovrebbe sapere tutto ciò che è nel backlog e le altre persone che aggiungono elementi al backlog del prodotto dovrebbero assicurarsi di comunicare con il 'product owner'.

Gestione del rilascio - Lo sprint non è un ciclo di rilascio, ma invece un ciclo di pianificazione. Questo significa che i team di scrum possono consegnare in qualsiasi momento. Idealmente, dovrebbero consegnare frequentemente durante lo sprint permettendo la revisione dello sprint per esaminare l'uso reale del cliente e il feedback. Tuttavia, la consegna continua non è sempre possibile e sono necessari altri modelli di rilascio. È importante per il 'product owner' sapere quando le cose possono e devono essere rilasciate.

Gestione delle parti interessate - Qualsiasi prodotto avrà molte parti interessate coinvolte che vanno da utenti, clienti, governance e leadership organizzativa. Il 'product owner' dovrà lavorare con tutte queste persone per assicurare efficacemente che il team di sviluppo stia fornendo valore. Questo può significare una grande quantità di gestione delle parti interessate e di comunicazione.

Iniziare con i ruoli Agile Scrum

I tre ruoli Scrum sono abbastanza semplici nel descrivere le tre principali aree di responsabilità in qualsiasi team Scrum, ma spesso è difficile mapparli nel proprio titolo di lavoro.

Se hai un sacco di grandi abilità per fornire valore al cliente e questo è ciò che ti entusiasma, allora dovresti essere un membro del team di sviluppo Scrum. In effetti, il team è l'elemento più importante di ogni organizzazione agile, in quanto fornisce effettivamente valore ai clienti e alle parti interessate. Ciò significa che l'anzianità è determinata da quanto tu fornisci valore o aiuti gli altri a farlo.

Se siete appassionati del cliente, della gestione delle parti interessate e del dominio del business, allora il ruolo di product owner sarebbe più adatto ai vostri desideri. Nella maggior parte delle organizzazioni, questa persona deve avere il rispetto e la fiducia del business, in modo da poter prendere decisioni. Il ruolo richiede anche un certo livello di capacità per negoziare i compromessi e rendere tutti felici.

Se vuoi aiutare i team a lavorare efficacemente insieme e vuoi anche cambiare il mondo con scrum e agile, allora il ruolo di scrum master è quello che fa per te, un ruolo molto incentrato sulle persone con una forte enfasi sul coaching, l'insegnamento e la facilitazione.

4. Case study

4.1 CGM Agile Playbook

Di seguito sono descritte le linee guida e le “best practices” (“plays” o “giochi”) che CGM ha adottato per introdurre la “Business Agility” in azienda (figura 6).

FIGURA 6 - ROADMAP DELLE PRACTICE DI CGM

Gestione agile dei progetti di sviluppo

La gestione agile dei progetti è un approccio iterativo alla gestione dei progetti di sviluppo del software che si concentra sui rilasci continui e sulla considerazione del feedback dei clienti ad ogni iterazione. I team di sviluppo software che abbracciano le metodologie di “project management” agile aumentano la loro velocità di sviluppo, espandono la collaborazione e favoriscono la capacità di rispondere meglio alle tendenze del mercato.

Gestione agile del programma di sviluppo

I primi ad adottare lo sviluppo agile sono stati piccoli team autonomi che lavoravano su piccoli progetti autonomi: hanno dimostrato che il modello agile può funzionare. Più recentemente, organizzazioni più grandi stanno scalando l'agile oltre i singoli team o progetti, e cercano modi per applicarlo a interi programmi.

Waterfall vs. Agile

Gli stili tradizionali di gestione dei progetti, come il waterfall, costruiscono utilizzando le fasi. Questo stile di sviluppo del prodotto si svolge tutto in un singolo rilascio ("big bang") ad alto rischio. Una volta che un progetto supera una fase, è difficile rivisitarlo perché i team spingono sempre in avanti verso la fase successiva.

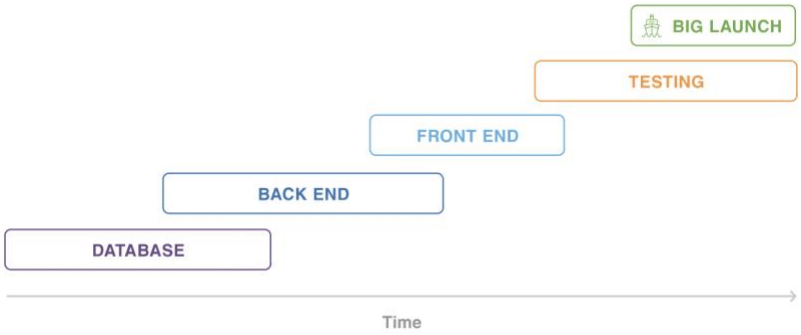

FIGURA 7 - FASI SECONDO WATERFALL

Gli stili tradizionali di gestione dei progetti spesso creano “percorsi critici”, dove il progetto non può andare avanti fino a quando un problema bloccante non viene risolto. Il cliente finale non può interagire con il prodotto finché non è completamente finito. In questo modo, problemi importanti nel design del prodotto, e nel codice, non vengono scoperti fino al rilascio.

Diversamente con uno stile di gestione del progetto agile, si adotta un approccio iterativo allo sviluppo con intervalli regolari di feedback. Queste iterazioni permettono al team di essere dirottato su (e produttivo in) un'altra area del progetto mentre un problema bloccante viene risolto.

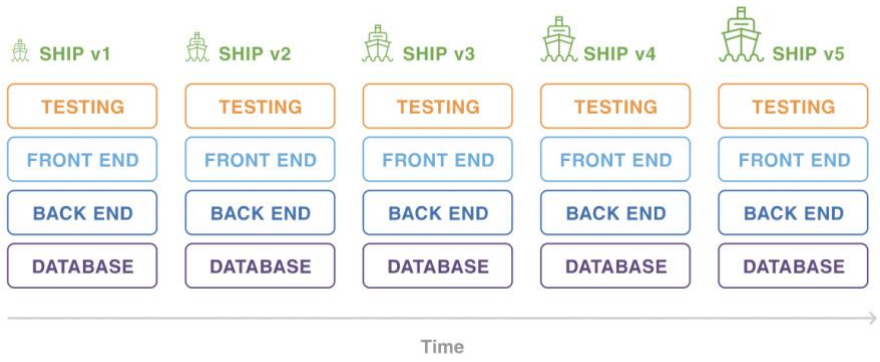

FIGURA 8 - FASI SECONDO AGILE

Oltre a rimuovere i percorsi critici, le iterazioni permettono di interagire con il prodotto durante lo sviluppo. Questo, a sua volta, dà al team opportunità costanti di costruire, consegnare, imparare e aggiustare. I cambiamenti del mercato non possono più prendere alla sprovvista e i team sono preparati ad adattarsi rapidamente ai nuovi requisiti.

Un beneficio ancora maggiore è la condivisione delle competenze all'interno del team di sviluppo software. La sovrapposizione delle competenze del team aggiunge flessibilità al lavoro in tutte le parti del codice del team. In questo modo, il lavoro e il tempo non vengono sprecati se la direzione del progetto cambia.

Come costruire un programma agile

Quando un programma passa dal "project management" tradizionale all'agile, il team e le parti interessate devono abbracciare due concetti importanti.

Il primo è che l'obiettivo del "product owner" è di ottimizzare il valore dell'output del team di sviluppo. Il team di sviluppo conta su questo ruolo che dà la priorità al lavoro più importante.

Il secondo è che il team di sviluppo può accettare il lavoro solo quando ha la capacità di farlo. Il "product owner" non spinge il lavoro al team o lo impegna a scadenze arbitrarie. Il team di sviluppo svolge il lavoro dal backlog del programma man mano che ne può accettare di nuovo.

In questi paragrafi vengono esplorati i meccanismi che i programmi agili usano per organizzare, eseguire e strutturare il lavoro in modo iterativo.

Roadmap

Una roadmap delinea come un prodotto o una soluzione si sviluppa nel tempo. Le roadmap sono composte da iniziative, che sono grandi aree di funzionalità, e includono linee temporali in cui una funzionalità sarà disponibile. Man mano che il programma si sviluppa è accettato che la roadmap cambi, a volte in modo limitato, a volte in modo ampio. L'obiettivo è quello di mantenere la roadmap focalizzata sulle condizioni attuali del mercato e sugli obiettivi a lungo termine.

Requisiti

Ogni iniziativa nella roadmap si scompone in un insieme di requisiti. I requisiti agili sono descrizioni leggere di funzionalità richieste, piuttosto che i documenti di 100 pagine associati ai progetti tradizionali. Si evolvono nel tempo e sfruttano la comprensione condivisa del cliente e del prodotto desiderato da parte del team. I requisiti agili rimangono snelli mentre tutti nel team sviluppano una comprensione condivisa attraverso la conversazione e la collaborazione continua. Solo quando l'implementazione sta per iniziare vengono completati tutti i dettagli.

Backlog

Il backlog stabilisce le priorità per il programma agile. Il team include tutti gli elementi di lavoro nel backlog: nuove funzionalità, bug, miglioramenti, compiti tecnici o architettonici, ecc. Il ‘product owner’ dà la priorità al lavoro nel backlog per il team di sviluppo. Il team di sviluppo poi usa il backlog prioritario come unica fonte per il lavoro che deve essere fatto.

Flusso di lavoro

I team agili progrediscono con le metriche. I limiti del lavoro in corso (WIP) mantengono il team e l'azienda concentrati sulla consegna del lavoro ad alta priorità. Grafici come burndown e control charts aiutano il team a prevedere la scadenza di consegna, e i diagrammi di flusso continuo aiutano ad identificare i colli di bottiglia.

Queste metriche mantengono tutti concentrati sui grandi obiettivi e aumentano la fiducia nella capacità del team di consegnare il lavoro futuro.

Iniziare in modo semplice, iniziare ora

Quando si implementa un flusso di lavoro per il team, iniziare sempre in modo semplice. Flussi di lavoro troppo complessi sono difficili da capire e adottare. Per i team di software, sono raccomandati questi stati di base del flusso di lavoro:

- To do: Lavoro che non è stato iniziato

- In progress: Lavoro che è attivamente esaminato dal team

- Code review: Lavoro completato e in attesa di revisione

- Done

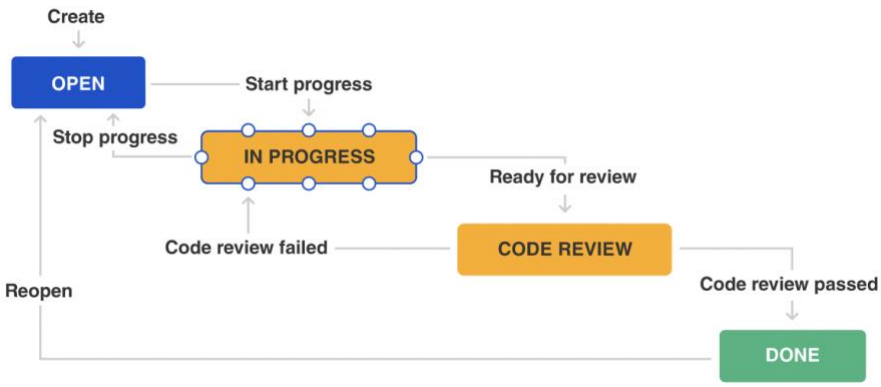

FIGURA 9 - SCHEMA IMPLEMENTAZIONE FLUSSO DI LAVORO

Lavoro che è completamente finito e soddisfa la definizione del team di 'done'

Discutere ogni punto critico nella 'retrospective' del team, e tenere a mente che ogni team avrà valori leggermente diversi in base al proprio progetto, al famework tecnologico e al metodo in cui gli piace lavorare. Ecco perché è importante scegliere un 'issue tracker' che abbia una configurazione flessibile del flusso di lavoro. Troppi team compromettono il loro stile di lavoro per adattarsi ad un particolare set di strumenti, il che è frustrante per tutti. Questo può portare i membri del team ad evitare di usare del tutto quello strumento, aggravando la frustrazione in tutto il team e in generale creando confusione. I team che sono nuovi all'agile o che non hanno

competenze inter-funzionali spesso finiscono con "mini cascate" nel loro flusso di lavoro.

Ottimizzare il flusso di lavoro

Quando si è a proprio agio con il flusso di lavoro di base e si è pronti a personalizzarlo, bisogna creare degli stati per ogni tipo di lavoro nel processo del team. Ideazione, design, sviluppo, revisione del codice e test sono funzionalmente diversi e possono essere stati individuali. Puntare ad un insieme snello di stati che comunichino chiaramente in quale fase si trova una parte di lavoro. Gli stati del progetto possono anche essere condivisi con il resto dell'organizzazione.

Il passo successivo nell'ottimizzazione del flusso di lavoro è assicurare un flusso costante di lavoro. I limiti del lavoro in corso (WIP) stabiliscono un numero minimo e massimo di problemi in un particolare stato del flusso di lavoro, assicurando che in ogni stato si abbia abbastanza lavoro per mantenere il team pienamente utilizzato, ma non così tanto da perdere la concentrazione nel destreggiarsi tra le priorità. Far rispettare i limiti WIP mostrerà rapidamente quali processi rallentano il lavoro complessivo attraverso la pipeline. Quando il team impara ad ottimizzare circa i suoi limiti WIP, il rendimento aumenterà.

Le sfide dello scaling di un flusso di lavoro

Le organizzazioni che hanno diversi team agili affrontano sfide speciali con i flussi di lavoro. I team agili che lavorano insieme possono trarre benefici dal condividere lo stesso flusso di lavoro. Usare lo stesso flusso di lavoro può rendere più facile la transizione del lavoro tra i team agili, perché usano le stesse convenzioni per definire e consegnare il lavoro. Creare un processo comune di solito comporta un po' di dare e avere da entrambi i team. Impareranno l'uno dall'altro e alla fine usciranno con un flusso di lavoro migliore. Non importa come sia il flusso di lavoro, anche il processo di sviluppo dovrebbe essere agile. Discutere nelle “retrospective” di tanto in tanto, e adattarsi man mano che la cultura e la composizione del team cambiano.

Epic, Stories, Thems e Initiatives

Le Stories, chiamate anche "user stories", riguardano brevi requisiti o richieste scritte dalla prospettiva di un utente finale. Le Epics sono una parte consistente del lavoro da realizzare che può essere suddiviso in un numero di tasks più piccoli (chiamati storie). Le Iniziatives sono collezioni di epics che si rivolgono verso un obiettivo comune. I Thems sono grandi aree di attenzione che abbracciano l'organizzazione.

Epic vs. Story

In un certo senso, le stories e le epics in agile sono simili alle storie e alle epiche nei film o nella letteratura. Una storia è una semplice narrazione; una serie di storie correlate e interdipendenti costituisce un'epica. Lo stesso vale per la vostra gestione del lavoro, dove il completamento di storie correlate porta al completamento di un'epic. Le stories raccontano l'arco del lavoro completato mentre l'epic condivide una visione di alto livello dell'obiettivo unificante. In un team agile, le stories sono qualcosa che il team può impegnarsi a finire entro uno sprint di una o due settimane. Spesso gli sviluppatori lavorano su dozzine di storie al mese. Le epics, al contrario, sono poche e richiedono più tempo per essere completate. I team spesso hanno due o tre epics che sviluppano per completare ogni trimestre.

Organizzare il lavoro in stories ed epics aiuta anche a comunicare efficacemente all'interno dell'organizzazione. Se si stanno riportando i progressi del team al responsabile di progetto, si parlerà in epics. Se si sta parlando con un collega del team di sviluppo, si parlerà a livello di storia.

Epic vs. Initiative

Nello stesso modo in cui le epics sono fatte di storie, le initiatives sono fatte di epics. Le initiatives offrono un altro livello di organizzazione sopra le epics. In molti casi, un'initiative coinvolge epics da più team per raggiungere un obiettivo molto più ampio e più grande di qualsiasi epic stessa. Mentre un'epic è qualcosa che si può completare in un mese o un trimestre, le initiative sono spesso completate in più trimestri o un anno.

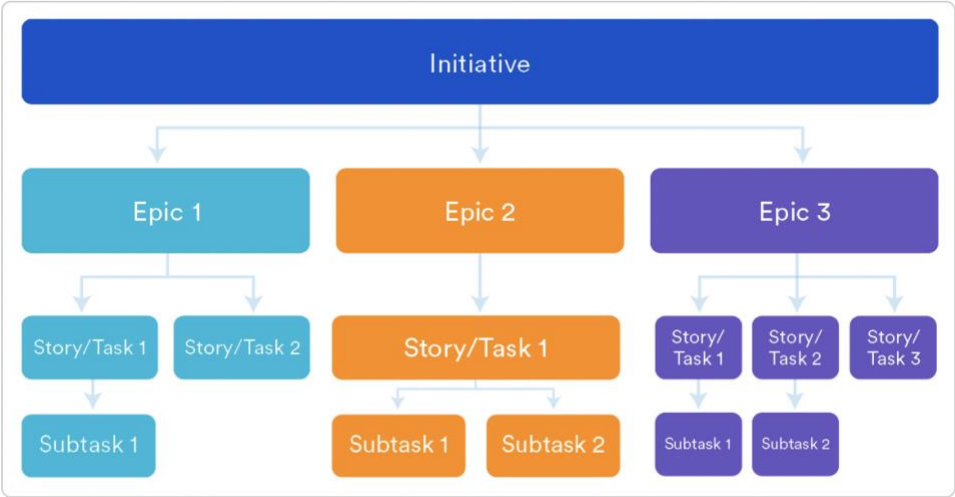

FIGURA 10 - ESEMPIO DI EPICS IN UN'INIZIATIVA

Initiative vs. Them

In molte organizzazioni i fondatori e il team di gestione incoraggiano il perseguimento di qualche aspirazione. Le initiatives hanno un design strutturale. Ospitano epics, e il completamento di quelle epics porterà al completamento dell'initiative. I thems sono uno strumento organizzativo che ti permette di etichettare gli elementi del backlog, le epics e le initiatives per capire quale lavoro contribuisce a quali obiettivi organizzativi. I thems dovrebbero ispirare la creazione di epics e initiatives, ma non hanno una relazione 1 a 1 con loro.

Cos'è un'epic agile?

Un'epic è una parte consistente del lavoro che può essere suddivisa in un certo numero di storie più piccole, o talvolta chiamate "Issues" in Jira. Le epics spesso comprendono più team, su più progetti, e possono anche essere tracciate su più schede. Le epics sono quasi sempre consegnate in una serie di sprint. Man mano che un team impara di più su un'epica attraverso lo sviluppo e il feedback dei clienti, le storie utente saranno aggiunte e rimosse come necessario. Questa è la chiave con le epics agili: lo scopo è flessibile, basato sul feedback del cliente e sulla cadenza del team

Capire le epics all'interno di un programma agile completo

Un'epic dovrebbe dare al team di sviluppo tutto ciò di cui ha bisogno per avere successo. Da una prospettiva pratica, è il livello superiore della loro gerarchia di lavoro. Tuttavia, capire come un'epic si riferisce ad altre strutture agili fornisce un contesto importante per il lavoro quotidiano di sviluppo. Un insieme di epics completate può completare un'iniziativa specifica, che mantiene il prodotto complessivo in sviluppo e in evoluzione con le richieste del mercato e dei clienti in cima ai temi organizzativi.

Creare un'epic agile

Creare epics intorno agli obiettivi trimestrali di un team è un ottimo inizio. Quando si crea un'epica, considerare quanto segue:

- Reporting - Creare epics per i progetti che i manager e i dirigenti vorranno tenere d'occhio.
- Story telling - Usare le epics, e le stories che si sviluppano in esse, come un meccanismo per raccontare la storia di come siete arrivati allo stato attuale di una caratteristica o di un prodotto.
- Cultura - Lasciare che sia la cultura organizzativa a dettare la dimensione e la granularità di un'epica.
- Tempo - La maggior parte dei team di sviluppo si affidano a strutture di stima invece che al tempo, ma è un controllo soggettivo utile per assicurarsi che le epics richiedano un paio di settimane per essere completate. Non troppo lungo e non troppo breve.

Scomporre un'epic agile

Suddividere un'epic in stories più pratiche aiuta a capire un progetto e a mantenere lo slancio, ma può essere un compito scoraggiante per i non esperti. Non c'è una soluzione unica per creare stories da un'epic, ma ci sono molte buone opzioni da considerare. Non c'è una definizione universale che traccia una linea tra una grande storia e un'epic. In generale, qualsiasi parte del lavoro che il team stima in "settimane" (o più) da completare, piuttosto che "ore" o "giorni" dovrebbe essere considerato un'epic e suddiviso in storie più piccole.

Misurare le epics agili

I grafici di burndown possono essere usati per visualizzare le epics, e servono a mantenere i team motivati e le parti interessate informate. Un buon grafico di burndown della epica è dove l'agilità dell'organizzazione brilla veramente. Un grafico di burndown della epica mostra la quantità effettiva e stimata di lavoro da fare in uno sprint o epic. L'asse x orizzontale in un grafico di burndown indica il tempo, e l'asse y verticale indica le storie o i problemi.

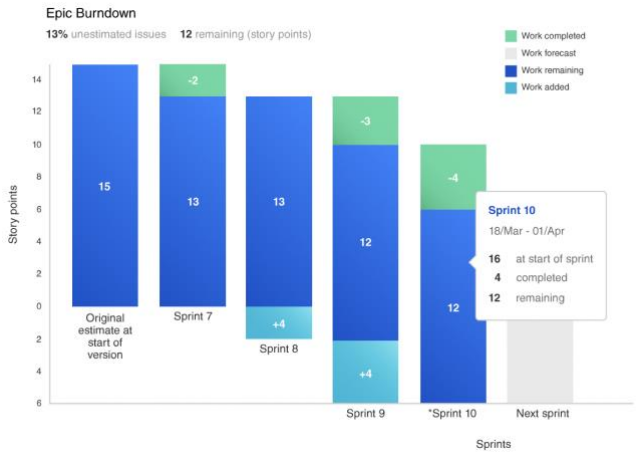

FIGURA 11 - GRAFICO DI BURNDOWN DI UN EPIC

Usare una Burndown Chart serve per tracciare il lavoro totale rimanente e per proiettare la probabilità di raggiungere l'obiettivo dello sprint. Tracciando il lavoro rimanente durante l'iterazione, un team può gestire il suo progresso e rispondere di conseguenza. Monitorando una Burndown Chart, diventa chiaro come il team sta procedendo e dove sono i blocchi. Avere questi punti chiaramente visibili mantiene tutti concentrati sullo stesso obiettivo e facilita una conversazione aperta sull'evoluzione del prodotto e sulle previsioni di completamento.

Le epics non sono il fondamento assoluto di un programma agile, ma sono i driver pratici per la maggior parte dei team agili e DevOps. Capire dove si inseriscono in un programma agile sano crea un contesto per il vostro lavoro, mentre la loro scomposizione in storie crea lo slancio.

User Story

Una user story è la più piccola unità di lavoro in un framework agile. È un obiettivo finale, non una caratteristica, espresso dalla prospettiva dell'utente del software. Una

user story è una spiegazione informale e generale di una caratteristica del software scritta dalla prospettiva dell'utente finale o del cliente. Le user stories sono poche frasi in un linguaggio semplice che delineano il risultato desiderato. Non entrano nei dettagli. I requisiti vengono aggiunti dopo, una volta concordati dal team.

Lo scopo di una user story è di spiegare come un pezzo di lavoro fornirà un particolare valore al cliente. Notare che i "clienti" non devono essere utenti finali esterni nel senso tradizionale, possono anche essere clienti interni o colleghi all'interno della vostra organizzazione che dipendono dal vostro team.

Le storie si adattano perfettamente alle strutture agili come Scrum e Kanban. In Scrum, le user stories sono aggiunte agli sprint e "burnt" durante la durata dello sprint. I team Kanban trascinano le user stories nel loro backlog e le eseguono attraverso il loro flusso di lavoro. È questo lavoro sulle user stories che aiuta i team scrum a migliorare la stima e la pianificazione degli sprint, portando a previsioni più accurate e a una maggiore agilità. Grazie alle storie, i team kanban imparano a gestire il work-in-progress (WIP) e possono perfezionare ulteriormente i loro flussi di lavoro. Le user stories sono anche gli elementi costitutivi di framework agili più grandi come le epics e le initiatives. Queste strutture più grandi assicurano che il lavoro quotidiano del team di sviluppo contribuisca agli obiettivi organizzativi costruiti nelle epics e nelle initiatives.

Perché creare user stories?

Le storie danno al team un contesto importante e associano i compiti al valore che questi compiti portano. Le user stories hanno una serie di benefici chiave come mantenere l'attenzione sull'utente. Una lista di cose da fare mantiene il team concentrato sui compiti che devono essere controllati, ma una collezione di storie mantiene il team concentrato sulla risoluzione dei problemi per gli utenti reali. Con l'obiettivo finale definito, il team può lavorare insieme per decidere come meglio servire l'utente e raggiungere quell'obiettivo. Le storie incoraggiano il team a pensare in modo critico e creativo a come risolvere al meglio un obiettivo finale.

Lavorare con le user stories

Una volta che una storia è stata scritta, è il momento di integrarla nel flusso di lavoro. Generalmente una storia viene scritta dal product owner, dal product manager o dal program manager e sottoposta a revisione.

Durante una riunione di pianificazione dello sprint o dell'iterazione, il team decide quali storie affronterà in quello sprint. I team discutono i requisiti e le funzionalità che ogni storia utente richiede. Questa è un'opportunità per diventare tecnici e creativi nell'implementazione della storia da parte del team. Una volta concordati, questi requisiti sono aggiunti alla storia.

Un altro passo comune in questo incontro è dare un punteggio alle storie in base alla loro complessità o al tempo di completamento. Le squadre usano le dimensioni della T-shirt, la sequenza di Fibonacci, o il poker di pianificazione per fare stime adeguate. Una storia dovrebbe essere dimensionata per essere completata in uno sprint.

Come scrivere le user stories

Bisogna prendere in considerazione quanto segue quando si scrivono le storie utente:

- Definizione di “done” - La storia è generalmente “fatta” quando l'utente può completare il compito delineato, ma bisogna assicurarsi di definire cosa sia.
- Delineare le sotto-attività o i compiti - Decidere quali passi specifici devono essere completati e chi è responsabile per ciascuno di essi.
- Personaggi utente - Per chi? Se ci sono più utenti finali, considerare di fare più storie.
- Passi ordinati - Scrivere una storia per ogni passo di un processo più ampio.
- Ascolta il feedback - Parlare con gli utenti e catturare il problema o il bisogno nelle loro parole. Non c'è bisogno di tirare a indovinare le storie quando si possono ricavarle dai clienti.
- Tempo - Molti team di sviluppo evitano del tutto le discussioni sul tempo, affidandosi invece ai loro framework di stima. Dal momento che le storie dovrebbero essere completabili in uno sprint, le storie che potrebbero richiedere settimane o mesi per essere completate dovrebbero essere suddivise in storie più piccole o dovrebbero essere considerate come epics.

Una volta che le user stories sono chiaramente definite, assicurarsi che siano visibili per l'intero team.

Story points e stima

La stima è difficile. Per gli sviluppatori di software, è tra gli aspetti più difficili, se non il più difficile, del lavoro. Bisogna prendere in considerazione una serie di fattori che aiutano i 'product owner' a prendere decisioni che influenzano l'intero team e il business. Con tutto questo in gioco, non c'è da meravigliarsi che tutti, dagli sviluppatori alla direzione superiore, siano inclini a fare i capricci per questo. Ma questo è un errore. La stima agile è solo questo: una stima. Non c'è l'obbligo di lavorare nei fine settimana per compensare la sottostima di un pezzo di lavoro. Detto questo, guardiamo alcuni modi per rendere le stime agili il più accurate possibile.

Collaborare con il 'product owner'

Nello sviluppo agile, il 'product owner' ha il compito di dare la priorità al backlog - la lista ordinata di lavoro che contiene brevi descrizioni di tutte le caratteristiche e correzioni desiderate per un prodotto. I product owner catturano i requisiti dal business, ma non sempre capiscono i dettagli dell'implementazione. Così una buona stima può dare al 'product owner' una nuova visione del livello di sforzo per ogni elemento di lavoro, che poi alimenta la loro valutazione della priorità relativa di ogni elemento. Quando il team di ingegneri inizia il processo di stima, di solito sorgono domande sui requisiti e sulle storie degli utenti. E questo è un bene perché queste domande aiutano l'intero team a capire meglio il lavoro. Per i 'product owner' in particolare, suddividere gli elementi di lavoro in pezzi granulari e le stime in punti storia li aiuta a dare priorità a tutte le aree di lavoro.

La stima agile è uno sport di squadra

Coinvolgere tutti (sviluppatori, designer, tester) nel team è la chiave. Ogni membro del team porta una prospettiva diversa sul prodotto e sul lavoro richiesto per consegnare una user story. Per esempio, se la gestione del prodotto vuole fare qualcosa che sembra semplice, come supportare un nuovo browser web, lo sviluppo e il QA devono pesare perché la loro esperienza ha insegnato loro quali sono i problemi che possono sorgere. Lasciare parte del team di prodotto più ampio fuori

dal processo di stima crea stime di qualità inferiore, abbassa il morale perché i collaboratori chiave non si sentono inclusi, e compromette la qualità del software.

Story points vs. ore

I team di software tradizionali danno stime in un formato temporale: giorni, settimane, mesi. Molti team agili, tuttavia, sono passati agli story points. Gli story points sono unità di misura per esprimere una stima dello sforzo complessivo richiesto per implementare completamente un elemento del product backlog o qualsiasi altro pezzo di lavoro. I team assegnano gli story points in relazione alla complessità del lavoro, alla quantità di lavoro e al rischio o all'incertezza. I valori sono assegnati per suddividere più efficacemente il lavoro in pezzi più piccoli, in modo da poter affrontare l'incertezza. Nel tempo, questo aiuta i team a capire quanto possono raggiungere in un periodo di tempo e costruisce il consenso e l'impegno per la soluzione. Può sembrare non intuitivo, ma questa astrazione è effettivamente utile perché spinge il team a prendere decisioni più difficili sulla complessità del lavoro.

Story points e planning poker

Le squadre che iniziano con gli story points usano un esercizio chiamato planning poker. Il planning poker è una pratica che dovrebbe essere comune in tutte le aziende. Il team prende un elemento dal backlog, lo discute brevemente e ogni membro formula mentalmente una stima. Poi ognuno tira fuori una carta con il numero che riflette la sua stima. Se tutti sono d'accordo, si prosegue. In caso contrario, bisogna prendersi un paio di minuti per capire la logica dietro le diverse stime.

Stimare in modo più intelligente, non più difficile

Nessun compito individuale dovrebbe essere più di 16 ore di lavoro. È semplicemente troppo difficile stimare singoli elementi di lavoro più grandi di questo con un alto grado di fiducia. E questa fiducia è particolarmente importante per gli elementi in cima al backlog. Quando qualcosa è stimato sopra la soglia di 16 ore del team, questo è un segnale per rompere in pezzi più granulari e ristimare. Per gli articoli più profondi nel backlog, basta dare una stima approssimativa. Nel momento in cui il team comincia effettivamente a lavorare su questi elementi, i requisiti

possono cambiare, così come l'applicazione. Quindi le stime precedenti non saranno così accurate. Non perdere tempo a stimare il lavoro che probabilmente cambierà. È sufficiente dare al 'product owner' una cifra approssimativa che può usare per dare priorità alla roadmap del prodotto in modo appropriato.

Imparare dalle stime passate

Le "retrospective" sono un momento per il team per incorporare intuizioni dalle iterazioni passate - inclusa l'accuratezza delle loro stime. Molti strumenti agili (come Jira Software) tengono traccia degli story points, il che rende molto più facile riflettere e ricalibrare le stime.

Metriche

Le metriche sono un argomento delicato. Da un lato, tutti sono stati su un progetto in cui non è stato tracciato alcun tipo di dato, ed è stato difficile dire se si è sulla buona strada per il rilascio o se sta diventando più efficienti man mano che si va avanti. Tracciare e condividere valide metriche agili può ridurre la confusione e far luce sui progressi (e le battute d'arresto) del team durante il ciclo di sviluppo. Ma come fanno questo?

'Done' racconta solo metà della storia. Si tratta di costruire il prodotto giusto, al momento giusto, per il mercato giusto. Rimanere in pista per tutto il programma significa raccogliere e analizzare alcuni dati lungo la strada. In qualsiasi programma agile, è importante tracciare sia le metriche di business che le metriche agili. Le metriche di business si concentrano sul fatto che la soluzione soddisfi i bisogni del mercato, mentre le metriche agili misurano gli aspetti del processo di sviluppo. Inoltre, le metriche di business di un programma dovrebbero essere radicate nella sua roadmap. Per ogni iniziativa sulla roadmap, includere diversi indicatori di performance chiave (KPI) che corrispondono agli obiettivi del programma. Inoltre, includere criteri di successo per ogni requisito di prodotto, come il tasso di adozione da parte degli utenti finali o la percentuale di codice coperto da test automatici. Questi criteri di successo alimentano le metriche agili del programma. E più i team imparano, meglio possono adattarsi ed evolvere.

Come usare le metriche agili per ottimizzare la consegna

1. Burndown dello sprint

I team Scrum organizzano lo sviluppo in sprint a tempo. All'inizio dello sprint, il team prevede quanto lavoro può completare durante uno sprint. Un rapporto di burndown dello sprint traccia quindi il completamento del lavoro durante lo sprint. L'asse x rappresenta il tempo, e l'asse y si riferisce alla quantità di lavoro rimasto da completare, misurato in punti storia o ore. L'obiettivo è di avere tutto il lavoro previsto completato entro la fine dello sprint. Un team che rispetta costantemente le sue previsioni è un buon risultato per l'organizzazione. Ma non lasciare che questo tenti di falsificare i numeri dichiarando un elemento completo prima che lo sia realmente. Può sembrare bello a breve termine, ma a lungo termine ostacola solo l'apprendimento e il miglioramento.

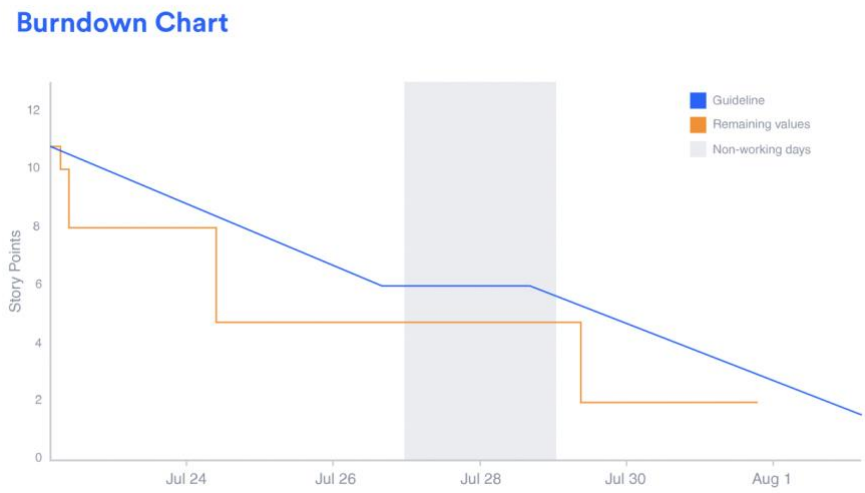

FIGURA 12 - GRAFICO DI 'BURNDOWN'

2. Burndown della epic e di rilascio

I grafici di burndown epic e di release (o versione) tracciano il progresso dello sviluppo su un ambito più grande del burndown dello sprint, e guidano lo sviluppo sia per i team scrum che per quelli kanban. Poiché uno sprint (per i team scrum) può contenere lavoro da diverse epics e versioni, è importante tracciare sia il progresso dei singoli sprint che delle epics e delle versioni. Mentre il team si muove attraverso il progetto, il 'product owner' può decidere di assumere o rimuovere il lavoro in base

a ciò che sta imparando. I grafici di burndown delle epics e di rilascio mantengono tutti consapevoli del flusso e riflusso del lavoro all'interno dell'epic e della release.

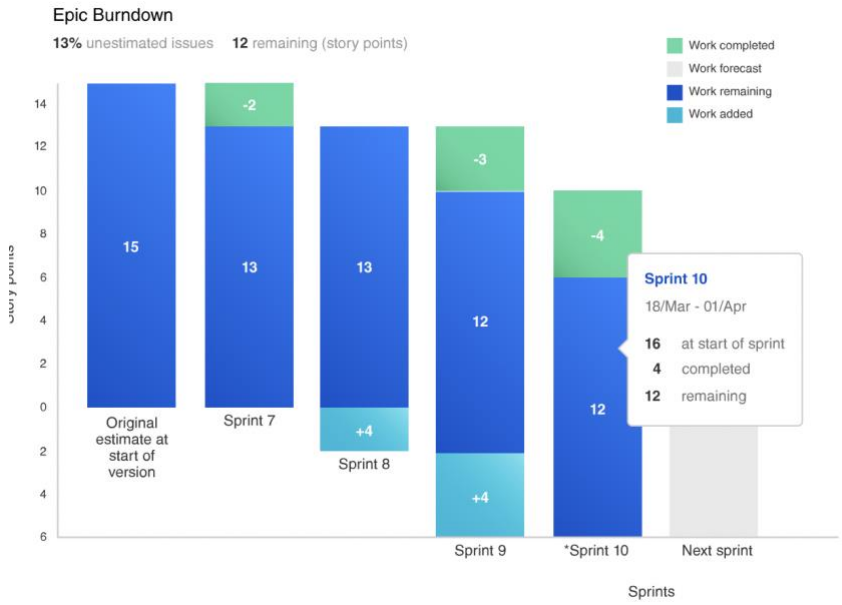

FIGURA 13 - GRAFICO BURNDOWN DI EPIC E RELEASE

3. Velocity

La velocità è la quantità media di lavoro che un team di scrum completa durante uno sprint, misurata in story points o ore, ed è molto utile per le previsioni. Il 'product owner' può usare la velocità per prevedere quanto velocemente un team può lavorare attraverso il backlog, perché il report traccia il lavoro previsto e completato su diverse iterazioni. È importante monitorare come la velocità si evolve nel tempo. I nuovi team possono aspettarsi di vedere un aumento della velocità mentre il team ottimizza le relazioni e il processo di lavoro. I team esistenti possono tracciare la loro velocità per assicurare prestazioni costanti nel tempo, e possono confermare che un particolare cambiamento di processo ha apportato o meno dei miglioramenti. Una diminuzione della velocità media è di solito un segno che qualche parte del processo di sviluppo del team è diventato inefficiente e dovrebbe essere riportato nella prossima 'retrospective'. La velocità di ogni squadra è unica. Se il team A ha una velocità di 50 e il team B ha una velocità di 75, non significa che il team B ha un throughput maggiore. Poiché la stima di ogni squadra è unica, lo sarà anche la loro velocità. Resistere alla tentazione di confrontare la velocità tra i team. Misurare il

livello di sforzo e l'output del lavoro basato sull'interpretazione unica di ogni team degli story points.

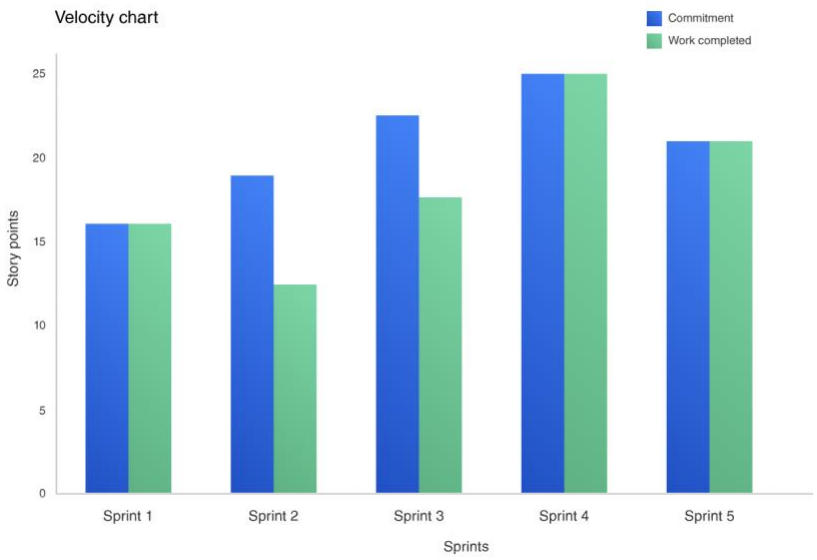

FIGURA 14 - GRAFICO DELLA VELOCITÀ

4. Grafico di controllo

I grafici di controllo si concentrano sul tempo di ciclo dei singoli problemi - il tempo totale da “in corso” a “fatto”. I team con tempi di ciclo più brevi hanno la probabilità di avere un più alto rendimento, e i team con tempi di ciclo consistenti su molte questioni sono più prevedibili nella consegna del lavoro. Mentre il tempo di ciclo è una metrica primaria per i team kanban, anche i team scrum possono beneficiare di un tempo di ciclo ottimizzato. Misurare il tempo di ciclo è un modo efficiente e flessibile per migliorare i processi di un team perché i risultati dei cambiamenti sono percepibili quasi immediatamente, permettendo di fare subito ulteriori aggiustamenti. L'obiettivo finale è quello di avere un tempo di ciclo coerente e breve, indipendentemente dal tipo di lavoro (nuova funzionalità, debito tecnico, ecc.).

5. Diagramma di flusso cumulativo

Il diagramma di flusso cumulativo dovrebbe apparire liscio da sinistra a destra. Bolle o vuoti in qualsiasi colore indicano carenze e colli di bottiglia, quindi quando se ne vede uno, bisognerebbe cercare modi per appianare le bande di colore attraverso il grafico.

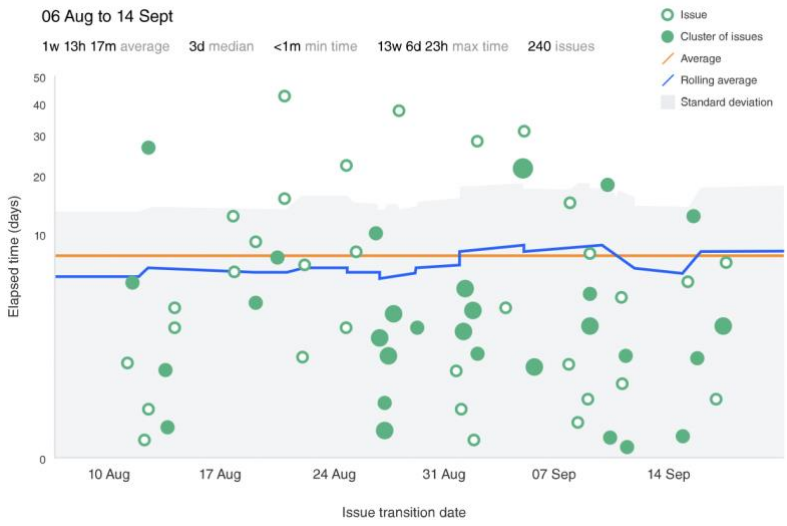

FIGURA 15 - GRAFICO DI CONTROLLO DEL TEMPO CICLO DELLE ATTIVITÀ INDIVIDUALI

Diagramma di Gantt

Un diagramma di Gantt è un grafico a barre orizzontale, basato sulla linea del tempo, che rappresenta un piano di progetto nel tempo. È stato inventato da Henry Gantt intorno al 1910 ed è stato molto usato durante il 20° secolo per la pianificazione dei progetti. I compiti elencati a sinistra del grafico hanno le loro barre corrispondenti sulla linea del tempo e queste rappresentano il flusso di lavoro in un progetto. Le date di inizio e fine delle attività, le 'milestones', le dipendenze tra le attività e gli assegnatari sono componenti classici dei diagrammi di Gantt. I software moderni si basano sul concetto originale: struttura dei compiti collassabile, percorso critico, barre di progresso e pannelli di gestione delle risorse. I diagrammi di Gantt sono usati dai project manager e dai product manager per suddividere un progetto in pezzi di lavoro gestibili, rimanere organizzati e visualizzare le dipendenze tra i compiti. I grafici di Gantt sono utili per semplificare progetti complessi. Lo strumento presenta tonnellate di dati in un modo incredibilmente visivo e aggregabile. Il grafico a barre mantiene i compiti in pista quando ci sono più parti interessate e grandi o numerosi team, o quando la portata cambia frequentemente. Un altro vantaggio dell'uso del diagramma di Gantt è mantenere una vista totale sull'intero progetto, specialmente su tutte le 'milestones' e le scadenze. Il diagramma di Gantt è efficiente come strumento di primo allarme.

Come usare un diagramma di Gantt

Passiamo rapidamente attraverso un modello di flusso di lavoro del diagramma di Gantt, come verrebbe usato da un project manager waterfall o ibrido.

In primo luogo, determinare una pianificazione del progetto: spezzare i progetti in pezzi di lavoro gestibili; pianificare le epics, le stories, le attività e le sotto-attività risultanti nel tempo (impostando le date di inizio e fine).

Poi stabilire ruoli, responsabilità e risorse: assicurarsi di avere abbastanza risorse per la quantità di lavoro e usare i pannelli di gestione delle risorse per evitare una sotto-/sovra-assegnazione delle risorse. Dopo bisogna monitorare il progresso del progetto e identificare le 'milestones': le 'milestones' sono momenti di verità; realizzazioni che i team dovrebbero raggiungere entro o prima del previsto. Sono opzionali ma raccomandate. Infine, trovare e segnalare i problemi: individuare i problemi reali e minacciosi usando il diagramma di Gantt e usare la funzionalità del percorso critico per identificare le attività che influenzeranno la data di completamento del progetto.

1. Determinare la logistica e le dipendenze dei compiti

I diagrammi di Gantt possono essere impiegati per tenere d'occhio la logistica di un progetto. Le dipendenze dei compiti assicurano che un nuovo compito può iniziare solo quando un altro compito è completato. Se un'attività è ritardata, allora le attività dipendenti sono automaticamente riprogrammate. Questo può essere particolarmente utile quando si pianifica in un ambiente multi-team.

2. Monitorare il progresso di un progetto

Come i team registrano il tempo in base ai problemi nel piano, alla stessa maniera si può monitorare lo stato di salute dei progetti e fare aggiustamenti se necessario. Il diagramma di Gantt può includere date di rilascio, 'milestones' o altre metriche importanti per monitorare il progresso del tuo progetto.

Diagrammi di Gantt nella pianificazione waterfall vs. agile

I diagrammi di Gantt possono essere uno strumento potente per entrambe le metodologie waterfall o agile.

Grafico specifico del progetto: è un piano dettagliato, che utilizza il tracciamento del tempo e le barre di progresso per ogni sotto-attività, così come le frecce che rappresentano le dipendenze tra le attività. I diagrammi di Gantt a livello di progetto tendono ad essere usati a livello di squadra o all'interno di un dipartimento. Questo tipo usa pesantemente le 'milestones', i percorsi critici e le linee di base, e queste aggiunte moderne al grafico a barre centrale motivano collettivamente il team assegnato o il dipartimento a consegnare un prodotto o un rilascio significativo in tempo.

4.2.1 Portfolio level plays

1) Roadmap Planning: è la pietra angolare per un anno di successo e per tutte le successive attività di pianificazione. Pertanto, è il primo passo per raggiungere tutti gli obiettivi proclamati dal consiglio di amministrazione.

"La previsione è molto difficile, specialmente se riguarda il futuro". Niels Bohr

2) La riunione di Roadmap Planning è il piano generale e la spina dorsale dell'avventura lavorativa annuale. Durante la riunione, tutti i team di un dipartimento si riuniscono per pianificare il lavoro per il prossimo anno. La sessione inizia con una presentazione della visione e della tecnologia utilizzata in modo che tutti i partecipanti abbiano una comprensione comune di ciò che pianificheranno in seguito. In seguito, vengono pianificati gli sforzi di lavoro e le dipendenze in modo da ottenere una tabella di marcia annuale con 'milestones' e uscite. Per essere sicuri di non lasciare indietro nessuno, controllare la lista dei ruoli che devono partecipare a questo gioco.

Prima di iniziare la riunione di Roadmap Planning, la visione della business unit deve essere condivisa nel team.

Passo 1 → Apertura + Contesto aziendale (15 min)

Il General Manager inizia la riunione presentando i risultati dell'ultimo anno, prima di nominare gli obiettivi e la strategia dell'organizzazione. Spiegare come la business unit sta contribuendo a questi obiettivi aziendali e dare una rapida panoramica sul

contesto aziendale. Definirà le ‘milestones’ e potrebbe aggiungere altre informazioni aggiuntive per mettere a bordo tutti i dipendenti.

Passo 2 → Visione del prodotto (15 min)

Ora il Product Architect presenta la visione del programma. Spiega le caratteristiche pianificate nelle iniziative ma non ha bisogno di descriverle in pieno dettaglio.

Passo 3 → Breakout del team: Raccolta degli argomenti (1 ora)

Il primo breakout di squadra è inteso come una sessione, dove le squadre (dotate di carte o grandi Post-It) stanno scrivendo tutti gli argomenti su cui si lavorerà durante l'anno. Prendere anche in considerazione gli elementi di lavoro che risiedono ancora nel Backlog. Fare una sessione di brain-storming con la squadra per generare idee innovative e argomenti che sarebbero interessanti dal punto di vista dell'innovazione. Non bisogna perdere nessun punto interessante qui.

Passo 4 → Presentazione dei temi (1 ora)

Dopo il breakout del team tutti i team si riuniscono di nuovo, per visitare le aree di lavoro di ogni team e cercare i temi o gli argomenti mancanti.

Passo 5 → Breakout del team: Magic Estimation (30 Min)

Ora dopo che tutte le squadre hanno annotato i loro temi di lavoro e gli argomenti per il prossimo anno, è il momento di stimarne la complessità con l'aiuto di Magic Estimation. Il metodo viene spiegato a tutte le squadre prima che le squadre vadano di nuovo nel loro spazio di lavoro per eseguire la magic estimation per i loro articoli di lavoro e argomenti. L'obiettivo è quello di ricevere un quadro generale su come i membri del team valutano la complessità di alcuni elementi di lavoro. Annotare la valutazione di complessità sulle carte quando la stima magica è fatta. Questo aiuta a pianificare, coordinare e stimare gli elementi di lavoro nei passi seguenti.

Passo 6 → Allocare le carte sulla timeline (30 min)

Ora che avete raccolto tutte le voci di lavoro per l'anno prossimo, è il momento di assegnarle alla timeline dell'anno. Disegnare una semplice linea temporale con i mesi o le uscite sulla tua lavagna. Ora, cercare di trovare il posto giusto per gli argomenti

rispettando le scadenze e i requisiti. Se un argomento è rilevante per più di una Release, tracciare una linea tra la scheda e il giorno o la scadenza finale.

Passo 7 → Breakout di squadra: Stime approssimative (30 min)

La prossima ora appartiene alle squadre stesse. Tutte le squadre conoscono i loro temi di lavoro e la loro complessità dal giorno prima. Hanno creato un piano approssimativo quando ogni tema di lavoro dovrebbe essere finito o è previsto per il lavoro. Ora è il momento di aggiungere un po' di specificità. Perciò tutte le squadre tornano alla loro area di lavoro. Il loro primo compito qui è quello di annotare la capacità di lavoro delle squadre per ogni periodo di rilascio. Ha senso preparare la capacità del team prima della riunione, perché può richiedere molto tempo. Ora, il team dovrebbe stimare in modo collaborativo il tempo necessario per gli elementi di lavoro. Annotare le stime sulla carta. Si riceve un'idea approssimativa di come gli elementi di lavoro si adatteranno alla capacità.

Passo 8 → Traduzione in epic (1 ora)

Solo ora è il momento di iniziare a pensare in Epic. Tradurre gli elementi di lavoro sulla lavagna in Epic. Dividere in diverse uscite se necessario e dividere anche le stime. Cercare di adottare il tempo originariamente stimato dalla scheda prima di aggiustare le Epics per adattarle alle Release. In questo modo ci si assicura che gli Epics e la loro priorità siano controllati di nuovo. Ora, aggiungere stime più specifiche e fare in modo che le Epics sulla lavagna si adattino alla capacità di rilascio.

Passo 9 → Presentazione delle epics (30 min)

Ogni team presenta la sua Roadmap agli altri team per controllare di nuovo se manca qualcosa o no.

Passo 10 → Da epics a JIRA (1 ora e 30 min)

Dal momento che la pianificazione sulla lavagna è compiuta e tutto è stato controllato, è il momento di trasferire le Epics dalla carta a JIRA. Ogni team è responsabile della propria Roadmap e tutti i membri del team aiutano in questo compito.

Passo 11 → Management review and problem solving (30 min)

Alla fine della giornata, il team di gestione si riunisce per fare aggiustamenti di scopo e aggiustamenti all'obiettivo in base alla pianificazione del giorno. I risultati del Management Review & Problem Solving sono decisivi per i prossimi passi. Se ci sono problemi, rischi o ambiguità riguardanti la Roadmap, che non sono stati ancora affrontati, bisogna ricominciare dal punto 8 e modificare la Roadmap con un altro breakout del team.

Passo 12 →Revisione della pianificazione e 'retrospective' (30 min)

La revisione della gestione presenta i suoi risultati a tutti i partecipanti, compresi i rischi e i colli di bottiglia in modo che tutti ricevano un quadro generale dell'impresa. Infine, si tiene una 'retrospective' di pianificazione, dove tutti possono dichiarare cosa è andato bene e cosa è andato male durante la pianificazione. Potrebbe avere senso raccogliere queste intuizioni nei singoli team in anticipo e definire un portavoce che rappresenti la percezione dei team della riunione.

3) Risultati

- il backlog prioritario con le iniziative del programma stimate e gli epics;
- impegno della Roadmap;
- suggerimenti per migliorare le future riunioni di pianificazione della Roadmap;
- spazi e rischi del team documentati.

4.2.2 Program level plays

Lo Scaled Agile Framework (SAFe) ha fuso i precedenti "Program Level" e "Team Level" nella configurazione "Essential SAFe" e non differenzia più questi livelli. Tuttavia, in questa descrizione si differenziano questi livelli per comprendere più profondamente come la teoria si applica ai modelli Jira proprietari di CGM. Non c'è un confine rigido tra il livello di Team e quello di Programma. I benefici di SAFe possono essere raggiunti solo se tutti su entrambi i livelli lavorano insieme come una squadra di squadre.

1) Il focus di questo livello è di consegnare una soluzione con il veicolo di un Agile Release Train (ART), che è usato per consegnare il valore ad una soluzione che può consistere in uno o più sistemi, servizi o prodotti.

Gli ART hanno le seguenti specifiche: sono composti da 5 a 10 team Scrum; pianificano, si impegnano ed eseguono le soluzioni insieme; sono auto-organizzati e gestiti; usano un approccio incrementale e iterativo per fornire valore.

Con il livello di programma vengono introdotti nuovi ruoli che aiutano ad allineare le persone a una missione condivisa e a coordinare i team agili, oltre a fornire la guida necessaria. La natura duratura, basata sul flusso e auto-organizzante dell'ART è ciò che alimenta SAFe. Gli ART hanno un'auto-organizzazione, una struttura e una missione più persistenti dei programmi tradizionali.

I ruoli principali sono il **Product Architect** e il **Product Manager** che è la voce interna dei clienti e lavora a stretto contatto con il PO. Insieme essi svolgono diversi compiti: forniscono e comprendono le esigenze del cliente, definiscono le caratteristiche del sistema, sono responsabili del Program Backlog.

Poi esiste la figura del System Architect / Engineer. Il concetto di Applied System Thinking è fornito dal System Architect / Engineer, che può essere una singola persona o un intero team. Le sue responsabilità sono: determinare gli elementi principali, definire e condividere la visione architettonica attraverso l'ART, progettare interfacce e collaborazioni.

Dopo c'è il Release Train Engeneer (RTE) che lavora insieme a team agili (dai 5 ai 10). Tra le sue responsabilità rientrano: protezione dell'incremento del programma dalle interruzioni, implementazione dei processi agili e facilitazione delle riunioni agili a livello di programma.

Altre figure importanti sono i proprietari del business. Essi sono stakeholder chiave, che: sono responsabili dei risultati del business, partecipano agli eventi ART e sono responsabili dell'uso, della "governance" e del "Return on Investment" di una soluzione sviluppata dall'ART.

2) Il Program Increment Planning (ex pianificazione del rilascio) è il fratello maggiore della pianificazione dello sprint. Similitudini: si deve pianificare cosa portare con te e rendere visibili le dipendenze.

Prima di iniziare il Program Increment Planning, ci deve essere un allineamento strategico tra le parti interessate e i Business Owner. Inoltre, la visione del prodotto deve essere definita prima di entrare nel workshop di PI Planning. È utile confrontare i team con la visione in anticipo e forse anche eseguire un workshop sulla visione in modo che il team possa già avere dei pensieri iniziali su come implementare la visione prima di andare all'evento di PI Planning. Questo potrebbe anche includere una sessione di rifinitura Epic che aumenterà la comprensione di argomenti complicati in modo massiccio e porterà ad una sessione di pianificazione molto più efficiente.

Passo 1 → Apertura + Contesto del business (30 min)

Il Business Owner inizia la riunione presentando la visione del Portfolio e lo stato attuale del business. Il BO spiega come la business unit sta contribuendo agli obiettivi aziendali e dà una rapida panoramica del Business Context. In CGM questo è molto probabilmente il GM.

Passo 2 → Apertura + Visione del prodotto (30 min)

Ora il Product Management presenta la visione del programma, le prossime 'milestones' ed evidenzia qualsiasi cambiamento dal precedente PI Planning così come.

Passo 3 → Visione dell'architettura e pratiche di sviluppo (45 min)

Il Software Architect introduce l'architettura, gli standard di codifica, i modelli di ramificazione, la strategia di deployment, gli NFR, i framework comuni e i corsi tecnici di azione. Il SA fornisce strategie per l'internazionalizzazione e informazioni per linee guida, wireframes, CSS, integrazione. In generale, il Software Architect evidenzia importanti aspetti dell'architettura e dello sviluppo per il prossimo incremento del programma e dove trovare le descrizioni.

Passo 4 → Pianificazione del contesto (15 min)

Il Release Train Engineer presenta il processo di pianificazione e i risultati attesi dell'evento.

Product owner: ha l'autorità di prendere decisioni a livello di User Story.

Maestri Agile: la sua responsabilità è di gestire il timebox, le dipendenze e le ambiguità.

Team di sviluppo: la loro responsabilità non è solo quella di consegnare Epics e Features, ma anche di incorporare ciò che hanno imparato durante i briefing di questa mattina.

Pianificazione dei requisiti: l'RTE spiega anche quali requisiti ogni team deve soddisfare prima che la pianificazione possa iniziare (figura 11).

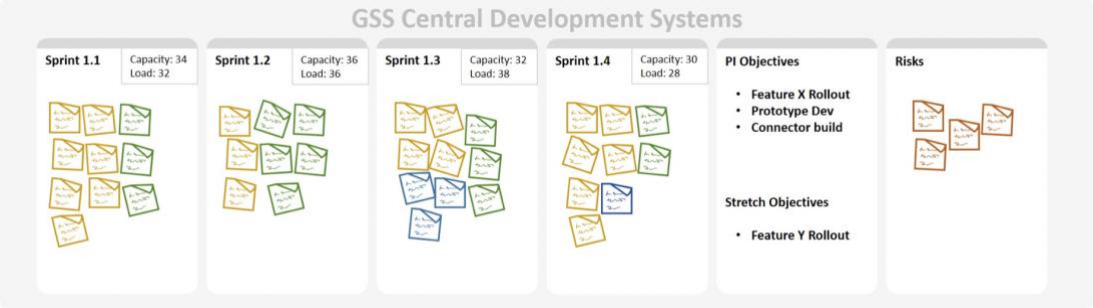

FIGURA 16 - VISIALIZZAZIONE REQUISITI DEL TEAM

Passo 5 → Breakout del team (3 ore)

Le prossime tre ore appartengono ai team Scrum e serviranno loro per calcolare la capacità o la velocità del team. Successivamente per sviluppare un piano approssimativo di PI.

La creazione di User Stories, comprese le stime approssimative e l'identificazione dei rischi e degli ostacoli. Le dipendenze interne o verso altri team sono gestite da Agile Masters e sono rese trasparenti a tutti nel Program Board. Dopo il Team Breakout, il team dovrebbe aver raggiunto quanto segue: Team Velocity definito, Storie stimate e descritte, Sprint (1, 2 & 3 e se possibile di più) riempiti di User Stories secondo la Velocity, Obiettivi PI scritti, Obiettivi Stretch definiti, Rischi del programma definiti.

Passo 6 → Scrum of Scrum (15 min)

Durante le sessioni di breakout del team, l'RTE facilita lo Scrum of Scrums da ora in poi insieme agli Scrum Master, per assicurare che il PI Planning sia sulla buona strada.

Passo 7 → Revisione della bozza del piano (1 ora)

Una volta che tutti i team hanno fatto buoni progressi durante le loro sessioni interne di pianificazione, ogni gruppo presenta un riassunto dei primi tre sprint, i rischi, i potenziali ostacoli e una o più bozze di obiettivi PI.

Passo 8 → Revisione della gestione e risoluzione dei problemi (1 ora)

Alla fine del primo giorno, il team di gestione si riunisce per fare aggiustamenti di portata e aggiustamenti all'obiettivo basati sulla pianificazione del giorno. I risultati di questa riunione saranno il punto di partenza per il giorno successivo (Passo 9).

Passo 9 → Aggiustamento della pianificazione (45 min)

La pianificazione del PI viene regolata in base ai risultati del Management Review e del Problem Solving (Passo 8) del giorno precedente. Gli aggiustamenti sono spiegati alle squadre in modo che tutti conoscano l'ultimo stato delle cose.

I possibili cambiamenti sono: priorità aziendali, aggiustamenti ai piani, modifiche all'ambito, spostamento di persone o storie.

Passo 10 → Breakout di squadra (2 ore)

Seguirà il Team Breakout #2 (come descritto nel Passo 5) per realizzare gli aggiustamenti annunciati nella pianificazione del PI. Durante questo breakout vengono pianificati i restanti sprint.

Passo 11 → Revisione finale del piano e pranzo (2 ore)

In seguito, ha luogo la revisione finale del piano. Questo slot include gli stessi passi del passo 6.

Passo 12 → Rischi del programma (1 ora)

Una delle parti finali del PI Planning è la discussione e il ROAMing (Resolving, Owning, Accepting, Mitigating) dei rischi che rimangono. Durante questa parte, i

rischi sono resi trasparenti a tutti e le responsabilità sono assegnate ai rischi per poterli affrontare durante il PI.

Passo 13 → Voto di fiducia (15 min)

Alla fine della pianificazione del PI, i vostri team devono fare due voti di fiducia. Uno riguarda la valutazione della pianificazione a livello di programma e l'altro riguarda la pianificazione a livello di squadra. Così ogni squadra ha un voto di fiducia sulla propria pianificazione interna al team. Per il voto di fiducia, tutti i partecipanti alla riunione votano sull'incremento del programma in arrivo.

Voto di fiducia: I votanti alzano le mani contemporaneamente con 1, 2, 3, 4 o 5 dita distese. 1 dito è il voto più basso e 5 dita il voto più alto. Si svolge una breve 'retrospective' sulla pianificazione del PI. Dà l'opportunità di implementare miglioramenti per la prossima Pianificazione PI.

3) Risultati

- backlogs prioritari con iniziative di programma stimate ed epics;
- spazi documentati del team e schede dei rischi;
- piano di incremento del programma e impegno;
- suggerimenti per migliorare le future riunioni di pianificazione.

Scrum of Scrums (SoS)

"I buoni leader devono prima diventare buoni servitori". Robert K. Greenleaf

1) Lo Scrum of Scrums (SoS) è un incontro regolare con i rappresentanti di altri team per ricevere una panoramica sullo stato delle cose negli altri team. Il suo scopo principale è quello di coordinare la collaborazione tra i team.

Come il Daily Scrum, questo evento SAFe si suppone abbia luogo nello stesso luogo e alla stessa ora ogni settimana. Uno dei partecipanti viene eletto come responsabile dello Scrum of Scrum (molto probabilmente il Release Train Engineer). Il responsabile organizza e facilita l'evento, invita tutti gli Agile Master dell'unità organizzativa e tiene traccia del timebox.

2) La riunione dovrebbe aver luogo almeno una volta alla settimana (più frequentemente se necessario) ed è simile al Daily Scrum ad un livello superiore. Il SoS aiuta a coordinare le dipendenze degli ART e fornisce visibilità sui progressi e sugli impedimenti attraverso i rappresentanti del team (spesso l'Agile Master) e l'RTE. L'evento è cruciale per mantenere l'Agile Release Train in pista e dovrebbe concentrarsi sulla revisione dei progressi verso le milestone, gli obiettivi PI e le dipendenze tra i team. L'RTE dovrebbe durare circa 30 minuti.

Sincronizzazione PO/PA

"Gli uomini d'affari e gli sviluppatori devono lavorare insieme quotidianamente durante tutto il progetto". Manifesto Agile

1) Il regolare PO/PA Sync è una riunione di stand-up tra un PA e i suoi PO subordinati. Ogni PO indica cosa è stato raggiunto dall'ultimo incontro e cosa sarà all'ordine del giorno fino al prossimo PO/PA Sync.

Il PO/PA Sync dovrebbe avvenire alla stessa ora e nello stesso posto in un ciclo settimanale (o più frequentemente se necessario). È una riunione a tempo in cui i Product Owner e i Product Architect si riuniscono in modo simile agli RTE e agli Agile Master nello Scrum of Scrums. Normalmente, un PA responsabile o uno dei partecipanti è l'organizzatore e responsabile dell'esecuzione e della facilitazione dell'evento.

2) La riunione è paragonabile alla riunione di Scrum of Scrums, ma si svolge con i Product Owner (PO) e i Product Architects (PA). Ogni PO rappresenta il suo o i suoi team.

Rispetto allo Scrum of Scrums il PO/PA Sync ha un focus leggermente diverso. Lo scopo della riunione è quello di ottenere visibilità su come l'Agile Release Train sta procedendo verso il raggiungimento dei suoi Obiettivi PI, per discutere e valutare i problemi o le opportunità e per valutare gli aggiustamenti dell'ambito. L'incontro può anche includere il Backlog Refinement o il Kick-Off per nuove caratteristiche ed è limitato a 30 minuti, ma può essere programmato un meet-after.

Riunione di gestione del rilascio

"Se ti concentri sui risultati, non cambierai mai. Se ti concentri sul cambiamento, otterrai risultati" Jack Dixon

1) Il Release Management Meeting è la pianificazione e il monitoraggio della Release e serve a mantenere pulito il backlog in modo che la Release sia allineata con la visione comune. Ha luogo su base regolare per valutare il contenuto, il progresso e la qualità. Il Release Management è attivamente coinvolto nel Release Scope Management e si occupa anche di altri argomenti come l'internazionalizzazione, il packing e il deployment, i requisiti di formazione, le comunicazioni interne ed esterne, ecc. Così le parti interessate interni ed esterni sono inclusi nella pianificazione di tutte le attività che portano al raggiungimento degli obiettivi.

2) I ruoli tipici che sono presenti durante la riunione sono il direttore generale, l'ingegnere del Release Train e il Product Architect. Ma oltre a loro, i seguenti ruoli potrebbero far parte della riunione: rappresentanti delle vendite e del marketing, Team-Leaders, proprietari di prodotto, IT interno, Senior Management, Architetti del software, Capo Tecnico ecc.

Passo 1→ Presentazione dello stato del rilascio

Il Product Architect disponibile presenta lo stato attuale delle loro release al GM. Presentano il loro lavoro di grooming mostrando gli aggiustamenti del backlog che hanno fatto. È possibile che questi aggiustamenti abbiano un effetto sull'Agile Release Train che sarà discusso nel prossimo passo.

Passo 2 → Soluzione

Tutti i partecipanti al Release Management Meeting discutono soluzioni o reazioni ai cambiamenti a livello di backlog che hanno un impatto sul Release Train. Una soluzione comune deve essere trovata in modo che tutti i team siano ancora allineati lungo il Release Train. Spesso tali cambiamenti porteranno i PA partecipanti all'azione perché molto probabilmente avranno bisogno di comunicare i cambiamenti ai loro Product Owner subordinati.

Passo 3 → Aggiustamenti

Infine, le soluzioni e gli aggiustamenti discussi sono applicati in modo che la release sia riallineata alla visione.

3) Risultati

- release Train/Roadmap aggiustato;
- backlog aggiustato.

Perfezionamento del backlog

"Il miglioramento continuo è meglio della perfezione ritardata". Mark Twain

1) Il Backlog Refinement è il compito continuo di un Product Architect e dei Product Owner. Riguarda la prioritizzazione di mantenimento e la preparazione del backlog del programma. È un processo continuo.

Il Product Architect / Product Architect Professional deve assicurare che lo sviluppo del prodotto è in corso ed è responsabile di fornire le informazioni richieste nel backlog, che rappresenta il lavoro futuro in modo prioritario. Questa è un'attività continua dei ruoli menzionati e dovrebbe anche includere un lavoro collaborativo in modo da creare e mantenere una comprensione condivisa del Program Backlog. Non c'è una regola generica, quanto spesso, quando e con chi incontrarsi. Questo dipende dai bisogni e se si necessitano esperti o di conoscenza del dominio o di altre persone specifiche.

2) Il processo di Refinement include la revisione e l'aggiornamento delle ipotesi di beneficio, la descrizione degli elementi del backlog, la creazione di criteri di accettazione, l'aggiunta di casi d'uso e la rifinitura delle User Story Maps. Se necessario gli elementi del backlog sono divisi in pezzi più piccoli che possono fornire valore in modo incrementale. Per gli elementi in cima al Backlog (quelli con la priorità più alta), vengono create le User Stories e vengono aggiunte le informazioni necessarie. Le stime approssimative sono fatte preferibilmente in modo relativo. Si possono usare diversi metodi di stima.

3) Risultati

- la parte superiore del backlog è raffinata;

- le issue contengono tutte le informazioni rilevanti;
- ipotesi di beneficio;
- criteri di accettazione;
- priorità;
- il backlog si sta preparando per il prossimo PI Planning.

Revisione dell'incremento del programma

"È una buona idea rivedere gli errori del passato prima di commetterne di nuovi". Warren Buffet

1) Se si è sviluppato un bel prodotto o rilasciato una nuova versione in un progetto promettente, questa è la riunione per mostrarlo. Presentare i risultati dell'ultimo incremento di prodotto del team e usare l'occasione per raccogliere un feedback diretto, che può essere usato per migliorare il prodotto.

2) L'evento Program Increment Review può essere paragonato al Review Meeting a livello di team, ma l'attenzione è posta sullo scambio di conoscenze presentando i risultati del team agli altri team della Business Unit invece di ricevere il feedback dei clienti

Il General Manager di una Business Unit organizza l'incontro con tutti i suoi team subordinati e dovrebbe avere luogo immediatamente dopo ogni Program Iteration.

Passo 1 → Presentazione

Il Direttore Generale apre la riunione spiegando l'ordine del giorno e presentando i team. In seguito ogni Team-Leader o rappresentante del team presenta i risultati del suo team nell'ultimo PI.

Passo 2 → Dimostrazione dal vivo

Durante una dimostrazione dal vivo ogni rappresentante del team presenta brevemente i prodotti sviluppati o le caratteristiche e le funzionalità più importanti. Prima si decide se le domande possono essere fatte subito o dopo la dimostrazione dal vivo in una sessione di domande e risposte. Si raccomanda di avere qualcuno che prenda appunti durante il feedback per essere in grado di usarli in seguito.

Passo 3 → Discussione e feedback

Lo scopo dell'Iteration Review è quello di ricevere informazioni su ciò che gli altri team stanno lavorando e di ricevere un feedback da persone esterne. Questo potrebbe portare a nuove idee che potrebbero essere rilevanti per l'ulteriore sviluppo del prodotto. La soddisfazione del cliente è di grande importanza, quindi è utile ricevere un feedback su base regolare e avere sessioni di discussione durante l'incontro.

Passo 4 → Aggiustamenti

L'input ricevuto porta a nuove richieste, miglioramenti, idee o requisiti che dovrebbero essere documentati e realizzati nei futuri incrementi del prodotto. In questo modo ci si può assicurare di avere un prodotto di successo e si minimizza il rischio di essere fuori bersaglio rispetto alle richieste dei clienti.

Foglio di lavoro

Questo foglio di lavoro dovrebbe essere usato durante la riunione di Iteration Review una volta al trimestre per organizzare e documentare tutti i risultati. Lo scopo della riunione è di avere una dimostrazione di ciò che è stato pianificato. L'Iteration Review è la migliore opportunità per i team di sviluppo di mostrare i loro risultati e per il GM di capire il risultato del trimestre precedente.

Action ICons

Se ci sono dei problemi emersi durante la riunione che devono essere risolti o su cui si deve ancora lavorare, bisogna definire un'Action. Un elemento di azione può essere un compito documentato, un evento, un'attività o un'azione. Si possono valutare i punti d'azione usando i "5 perché" o l'"analisi di Fishbone". Questi elementi possono essere pianificati nel prossimo Sprint. Naturalmente, si può tracciare e gestire le Action Item retrospettive in un altro modo.

3) Risultati

- scambio di conoscenze sui prodotti in tutta la business unit;
- dimostrazione dal vivo dell'incremento del prodotto;
- raccolta di nuove idee.

Retrospective di release

"A intervalli regolari, il team riflette su come diventare più efficace, poi sintonizza e regola il suo comportamento di conseguenza." Manifesto Agile

1) La Retrospective PI è la riflessione interna dell'Agile Release Train sull'ultimo Incremento di Programma e parte dell'Inspect & Adapt Workshop definito nello Scaled Agile Framework. Durante l'evento, l'Agile Release Train con tutti i suoi membri valuta e discute ciò che deve essere migliorato per fornire frequentemente valore, operare con successo e rendere il lavoro collaborativo ancora migliore. Questi miglioramenti saranno adattati all'interno del prossimo incremento di programma e renderanno il lavoro futuro più efficiente, miglioreranno il prodotto e renderanno i dipendenti più felici o più impegnati. Inoltre, la riunione può essere usata per risolvere problemi interni.

2) L' "Iteration Retrospective" è corrispondente alla "Sprint Retrospective" ma riflette l'ultimo incremento di programma e il lavoro dell'Agile Release Train invece che di un solo team. Pertanto ogni membro dell'ART dovrebbe essere presente. Una buona preparazione è la chiave. Se si stanno raccogliendo attentamente i miglioramenti, le sfide e gli impedimenti durante un incremento di programma, si hanno già i punti principali per l'agenda.

L'obiettivo è la creazione di elementi di miglioramento su cui l'Agile Release Train si impegna e che saranno implementati nei futuri PI. Idealmente ha luogo dopo la revisione del PI e prima del prossimo Program Increment Planning.

L'RTE dovrebbe preparare questo evento insieme agli Agile Masters. Questo potrebbe includere la preparazione di misure raccolte, esperienze dai team o problemi trovati dal punto di vista dell'Agile Master o del Release Train Engineer. Mettetevi anche in contatto con il vostro PA se sono stati rilevati problemi importanti nell'area del Product Management che potrebbero essere inseriti nell'agenda.

Il formato della sessione è abbastanza aperto e può essere basato sui risultati menzionati sopra o può avere un'agenda aperta come il workshop Inspect & Adapt definito dallo Scaled Agile Framework.

Passo 1→ Ispezionare

Misure quantitative Il Product Architect / Development evidenzia i KPI chiave dell'ultimo incremento di prodotto. Questo potrebbe includere il prodotto o lo sviluppo relativo ricevuto dal feedback dei clienti o dai rapporti. Se necessario, i miglioramenti vengono discussi e documentati.

Misure qualitative Come RTE dovreste includere anche misure qualitative, potreste averle raccolte prima ma naturalmente potete anche eseguire una sessione per raccogliere feedback qualitativi dai partecipanti.

Passo 2 → Raccogliere

Identificare e raccogliere i problemi o gli impedimenti che il vostro Agile Release Train deve affrontare. Create una lista che possa essere compresa da ogni partecipante.

Passo 3 → Dare priorità

Sulla base del principio di Pareto o della regola 80/20, che afferma che si può risolvere l'80% del problema con il 20% dello sforzo, c'è bisogno di dare priorità ai problemi o alle questioni identificate. Questo aiuta a identificare i più importanti e ad affrontarli per primi. Eseguire una sessione di voto con tutti i partecipanti.

Passo 4 → Adattare

Cercare di trovare soluzioni per le voci più votate. L'obiettivo è quello di identificare i successi rapidi per i quali potrebbe essere necessario capire prima la causa principale. Usare i 5 perché o un diagramma a lisca di pesce per aiutarsi. Poi cercare di ricavare le azioni che devono essere implementate nel prossimo incremento del programma. Evitare di farsi prendere da discussioni dettagliate. È possibile che le azioni dettagliate debbano essere discusse nei team di sviluppo o a livello di esperti. Tuttavia, evidenziare ciò che si deve adottare nel prossimo incremento del programma e assegnare i compiti a squadre o esperti se necessario.

3) Risultati

- miglioramenti impegnati;

- "assegnazioni di "compiti a casa;
- colleghi soddisfatti;
- documentazione della riunione.

4.2.3 Team level plays

L'unità fondamentale di Scrum e SAFe è una piccola squadra di persone, lo Scrum Team. Lo Scrum Team consiste di un Agile Master, un Product Owner e degli sviluppatori, non ha sotto-team o gerarchie all'interno del team, si concentra su un obiettivo alla volta, il Product Goal, è inter-funzionale nel senso che i membri hanno tutte le competenze necessarie per creare valore ad ogni iterazione, è composto tipicamente da 3-9 persone più Agile Master e Product Owner, per una migliore comunicazione e produttività, è responsabile delle attività relative al prodotto, dalla collaborazione con le parti interessate, alla verifica, alla manutenzione, al funzionamento, alla sperimentazione, alla ricerca e sviluppo, e qualsiasi altra cosa che potrebbe essere richiesta e infine è responsabile della creazione di un incremento utile e di valore ogni Sprint.

Ruoli

Sviluppatori: sono responsabili della creazione di aspetti di un incremento utilizzabile ad ogni sprint. Il loro ampio set di abilità varia a seconda dell'area di lavoro. Tuttavia, gli sviluppatori sono responsabili per il Backlog di Sprint, assicurare la qualità aderendo ad una Definizione di Fatto, personalizzare il loro lavoro ogni giorno verso l'obiettivo dello sprint, ritenersi reciprocamente responsabili come professionisti

Product owner: essendo responsabile di massimizzare il valore del prodotto, il PO rappresenta i bisogni delle parti interessate nel Product Backlog ed è responsabile di delegare il lavoro da completare. Il PO è responsabile di una gestione efficace del Product Backlog, che include: comunicare e sviluppare l'obiettivo del prodotto; ordinare, creare e comunicare gli elementi del Product Backlog; mantenere il Product Backlog trasparente e comprensibile.

Scrum master: è responsabile di stabilire Scrum come definito nella Guida Scrum, o in qualsiasi altro quadro scelto. L'AM (Agile Master) assicura che la teoria e la pratica siano comprese dal team e dall'organizzazione. L'AM è responsabile dell'efficacia del team e serve il team allenandoli nell'autogestione e nella inter-funzionalità, assicurando che tutti gli eventi abbiano luogo e aiutando il team a concentrarsi sullo sviluppo di incrementi di alto valore. L'AM serve il PO in termini di aiutare a stabilire una pianificazione empirica del prodotto per un ambiente complesso, facilitare la collaborazione delle parti interessate come richiesto, fornendo tecniche per un'efficace definizione degli obiettivi di prodotto e la gestione del Product Backlog. L'AM serve l'organizzazione in termini di pianificare e consigliare l'implementazione di Scrum all'interno dell'organizzazione, rimuovendo gli ostacoli tra il team e le parti interessate, guidando e allenando l'organizzazione nella sua adozione di Scrum

Pianificazione dello sprint

1) La pianificazione dello sprint è paragonabile alla preparazione della valigia prima del viaggio. Si deve pianificare la vacanza nel dettaglio in modo da essere preparati per ogni situazione. “La bellezza dell'agile risiede nella sua natura incrementale e nell'uso dell'empirismo per concentrarsi su interazione, iterazione e miglioramento”. Perla Zhu

2) Lo Sprint Planning dà inizio allo Sprint e consiste nel pianificare il lavoro, creare uno Sprint Backlog insieme ad uno Sprint Goal e come Team impegnarsi nel piano. L’Agile Master dovrebbe assicurarsi che ogni membro del team possa prendere parte allo Sprint Planning e sia invitato. Per essere sicuri di non lasciare indietro nessuno, controllare la lista delle persone che devono partecipare a questo play. È cruciale avere l'intero team Scrum disponibile perché la creazione di uno Sprint Backlog è un compito collaborativo. Lo Scrum team può invitare altri per un consiglio.

Passo 1 → Presentare il Product Backlog - Il Cosa

Il Product Owner inizia la riunione presentando gli elementi ad alta priorità del Product Backlog (Top 20 User Stories) e come sono mappati al Product Goal. Il PO spiega come lo Sprint potrebbe aumentare il valore del prodotto. Se necessario il

team fa domande per capire gli elementi di lavoro per definire come implementarli nel corso successivo dello Sprint Planning.

Passo 2 → Pianificazione della capacità - Previsione

Il team Scrum seleziona gli elementi dal Product Backlog per portarli nello Sprint Backlog secondo la capacità del team. Questo può essere difficile all'inizio, ma più la squadra ha esperienza con le performance passate, la velocità e la comprensione della Definition of Done, più fiducia si avrà durante la pianificazione. Un ottimo strumento per aiutarvi a pianificare la capacità è il Capacity Plan in SPOCK.

Passo 3 → Pianificazione e scomposizione - Il come

A seconda della capacità calcolata del team, lo Sprint Backlog viene riempito con le User Stories. Per ogni elemento del Product Backlog gli sviluppatori pianificano il lavoro che è necessario per creare un incremento che soddisfa la definizione di Fatto. Questo potrebbe comportare di spezzare gli elementi del Product Backlog (per esempio le User Stories) in elementi di lavoro più piccoli o lo scopo di questi elementi è definire come l'implementazione o il lavoro può essere fatto. Alla fine, lo Sprint Goal, gli elementi del Product Backlog selezionati per la realizzazione nello Sprint e il piano per consegnarli sono chiamati Sprint Backlog.

Passo 4 → Definire e documentare lo Sprint Goal

Rendere lo Sprint Goal (almeno) disponibile allo Scrum Team, incluso il Product Owner e le parti interessate. Questo aiuta il team a sapere cosa fare e su cosa concentrarsi, inoltre definisce il risultato atteso dello Sprint.

Passo 5 → Voto di fiducia

Alla fine della riunione, l'Agile Master può iniziare un voto di fiducia. Qui, tutti i membri del team possono esprimere le loro aspettative riguardo al raggiungimento degli obiettivi dello Sprint. Tutti i membri del team alzano la mano allo stesso tempo con 1, 2, 3, 4 o 5 dita distese. "1" è la valutazione più bassa; "5" la più alta. Viene calcolata una media e sia i membri del team con il voto più alto che quello più basso devono spiegare le ragioni del loro voto. In questo modo i problemi che possono sorgere possono essere rappresentati e discussi prima dell'inizio dello Sprint.

Passo 6 → Iniziare lo Sprint

3) Risultati

- ogni membro del team sa cosa fare;
- l'obiettivo dello sprint e lo sprint backlog sono definiti;
- il team si impegna a raggiungere l'obiettivo;
- Sprint Goal, Sprint Backlog e piano di consegna sono documentati.

Lo sprint è iniziato

1) Il Daily Scrum è paragonabile a scrivere un blog sul vostro viaggio. Su base giornaliera, raccontare agli altri ciò che si è vissuto ieri e ciò che è previsto per oggi per raggiungere l'obiettivo di viaggio.

"… gli stand-up servono a mettere in evidenza i problemi - o quelli che noi chiamiamo 'impedimenti' ". Jeff Sutherland

2) Il Daily Scrum dovrebbe essere parte della l'attività quotidiana in CGM. È una semplice, breve riunione per tenere il team aggiornato sui progressi verso gli obiettivi dello sprint e sui potenziali impedimenti. Lo scopo del Daily Scrum è quello di controllare i progressi fatti verso gli Sprint Goals e come procedere ulteriormente per raggiungerli. È un evento di routine per gli sviluppatori di uno Scrum Team. Lo Scrum quotidiano si trasformerà in un'abitudine per gli sviluppatori, poiché si svolge nello stesso luogo e alla stessa ora ogni giorno, per ridurre la complessità. Tenere la riunione di Daily Scrum al mattino ha dei vantaggi poiché il piano per il lavoro del giorno può essere realizzato e direttamente eseguito.

Passo 1 → Burndown & Scrum (Scrumban) Board

Secondo lo Scrum degli sviluppatori è possibile scegliere qualsiasi struttura e tecnica disponibile, purché il loro Daily Scrum si concentri sui progressi verso gli Sprint Goals e produca un piano d'azione per il prossimo giorno di lavoro. Tuttavia, dare un'occhiata al grafico di Burndown aiuta enormemente a capire i progressi del team. Mostrare la Scrum Board (o Kanban/Scrumban Board) durante il Daily Scrum ha dei vantaggi per la capacità di attenzione dei partecipanti e fornisce una guida mentre presenta ciò che è stato fatto. I membri del team possono spiegare il loro lavoro

realizzato con l'aiuto della lavagna e non sono distratti dal capire cosa dire mentre aspettano il loro turno.

Passo 2 → Pianificare la giornata

L'obiettivo è quello di ricevere un piano d'azione per la giornata su come procedere per raggiungere gli obiettivi dello sprint. Il Daily Scrum crea concentrazione e migliora l'autogestione, poiché aiuta ad aumentare la comunicazione, identifica gli impedimenti, promuove un rapido processo decisionale, e di conseguenza elimina la necessità di altre riunioni. Naturalmente il Daily Scrum non è l'unico momento in cui gli sviluppatori possono modificare il loro piano. Potrebbero anche incontrarsi nel corso della giornata per aggiustare il loro fare al fine di raggiungere gli Sprint Goals.

Perfezionamento del backlog

"La perfezione si raggiunge non quando non c'è più niente da aggiungere, ma quando non c'è più niente da togliere". Antoine de Saint-Exupery

1) Il "Backlog Refinement" è l'attività quotidiana di un Product Owner. Per aumentare la trasparenza ha senso coinvolgere il team Scrum in un evento del genere per ricevere feedback dagli sviluppatori, chiarire le domande e preparare gli sviluppatori per il prossimo Sprint Planning.

2) Potrebbe avere senso invitare le parti interessate tecnici per ottenere gli approfondimenti necessari. I partecipanti alla riunione aggiungono ulteriori informazioni alle voci del Product Backlog. Raffinano le stime, le dipendenze, i concetti, i criteri di accettazione, le priorità e potrebbero essere in grado di dividere gli elementi del Backlog in pezzi di lavoro più piccoli.

Passo 1 → Presentazione del Product Backlog

Il Product Owner presenta il Product Backlog agli sviluppatori. Gli elementi del backlog dovrebbero essere stati prioritizzati dal PO in anticipo.

Passo 2 → Gli sviluppatori fanno domande

Durante o dopo la presentazione del backlog prioritizzato, ogni membro del team può fare domande per rendere l'argomento e la prioritizzazione tangibile per lui/lei.

Questo è importante, perché tutti dovrebbero essere coinvolti nel processo di rifinitura della riunione e quindi hanno bisogno di capire l'obiettivo del prodotto e come la priorità e il contenuto degli elementi del Backlog aiutano a raggiungere l'obiettivo.

Passo 3 → Grooming

Riassumendo, il processo di grooming o refinement può contenere le seguenti azioni: mostra e spiega l'obiettivo del prodotto e la priorità degli elementi del Backlog, è pronto a rispondere alle domande, aggiunge dettagli agli elementi del Backlog. Il raffinamento inizia dalla cima del backlog e continua fino a quando la riunione a tempo non raggiunge la sua fine. Tutti possono aggiungere informazioni alle JIRA Issues. Non è un compito specifico degli OP.

Passo 4→ Aggiornamento dello stato dei ticket

Poco prima che il time-box sia consumato, potrebbe avere senso che il Product Owner dia un'altra occhiata al Backlog insieme al team per evidenziare gli elementi importanti per il prossimo futuro. Questo è tutto, avete finito!

3) Risultati

- storie utente raffinate e criteri di accettazione;
- stime, descrizioni e informazioni aggiuntive (Mockup, modelli, dati, ecc.);
- backlog con priorità;
- comprensione condivisa del lavoro imminente e degli obiettivi del prodotto;
- allineamento.

Sprint Review

1) La Sprint Review è il fratello minore della Release Review. Si tratta di condividere i risultati e le esperienze con gli amici. Riflettere sul recente viaggio in dettaglio e forse ricevere qualche utile consiglio e feedback per avere un viaggio ancora migliore la prossima volta. Prospettive diverse spesso possono portare ad alcune grandi idee.

"Ascolta i tuoi clienti, non i tuoi concorrenti". Joel Spolsky

2) Lo Scrum team si riunisce per presentare i risultati dell'ultimo Sprint a parti interessate, clienti e colleghi. L'incontro offre spazio per discussioni e feedback che possono portare a nuove idee. Il feedback dato può essere implementato nelle future sessioni di pianificazione e influenzerà il Product Backlog.

Per eseguire la Sprint Review, è importante prendersi cura di un ambiente preferibilmente neutrale. La Sprint Review mira alla valutazione del prodotto e quindi ispeziona i progressi verso lo Sprint e il Product Goal. Se non fossero stati raggiunti, il team o le singole persone non dovrebbero essere incolpati per questo. Il fallimento è considerato un punto di apprendimento. Si noti che l'ispezione del perché non è stato raggiunto fa parte della “Sprint Retrospective”. La Sprint Review dovrebbe essere focalizzata sui risultati dello Sprint e su cosa fare dopo.

Passo 1 → Obiettivo dello sprint

Molto probabilmente il Product Owner inizia la riunione con una breve introduzione degli Sprint Goals definiti nell'ultimo Sprint Planning. Questo aiuta ad allineare tutti ed è un buon kick-off per l'evento.

Passo 2 → Presentazione

Il team di sviluppo presenta i propri risultati ale parti interessate e agli altri partecipanti con l'aiuto di una demo dal vivo. Si desidera un feedback riguardante i risultati effettivi e lo stato attuale del prodotto. È meglio preparare un caso di test, che si può eseguire durante la presentazione per renderla più visiva e comprensibile agli altri.

Passo 3 → Rivedere

Lo Scrum Team e le parti interessate rivedono i risultati e i cambiamenti dell'ambiente. Sulla base di questo, i partecipanti discutono e collaborano su cosa fare dopo. Questo potrebbe portare a cambiamenti nel Product Backlog ma potrebbe anche risultare in nuove idee ed opportunità. La Sprint Review è una sessione di lavoro e come detto sopra non dovrebbe essere limitata ad una presentazione.

Passo 4 → Prevedere

Ha senso preparare l'evento in anticipo ed evidenziare lo stato attuale del Product Backlog, per mettere tutti sulla stessa pagina e assicurare che il lavoro imminente sia allineato alle necessità del cliente. L'evento non dovrebbe durare più di 2 ore per uno Sprint di 2 settimane.

3) Risultati

- valutazione se il team è sulla buona strada per raggiungere i requisiti e gli obiettivi;
- dimostrazione dei risultati dello sprint;
- documentazione dell'incontro che include i risultati del feedback e della discussione;
- backlog aggiustato e sllineamento tra Scrum Team e clienti.

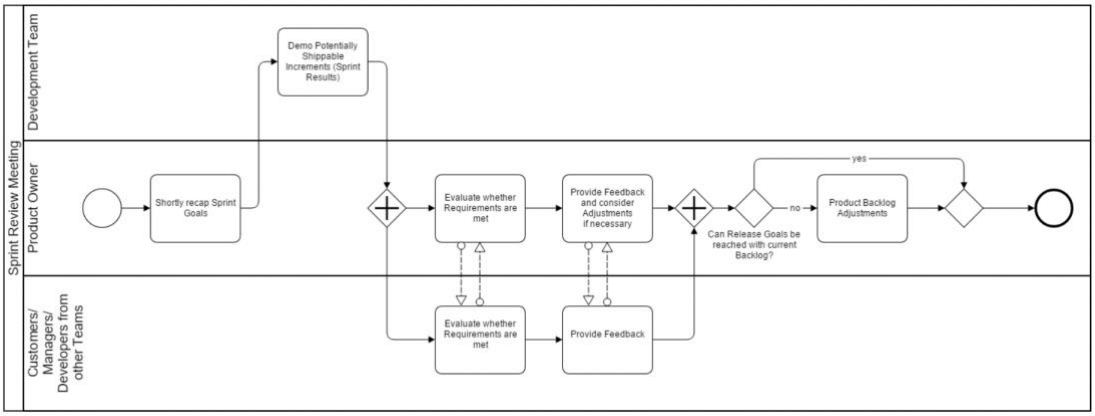

FIGURA 17 - ESEMPIO DI UNO SPRINT REVIEW MEETING

Sprint retrospective

"Se adottate una sola pratica agile, che sia ‘retrospective’. Tutto il resto seguirà" Antoine de Saint-exupery

1) Non importa quanto sia bravo il tuo team Scrum, c'è sempre un'opportunità per migliorare i processi interni. Questa riunione serve a riflettere sull'ultimo Sprint. Ogni membro del team prende una parola e dichiara le esperienze durante l'ultima iterazione. Questo porta a miglioramenti nell'attività quotidiana e nella collaborazione e rafforza lo spirito del team.

2) L'obiettivo è quello di pianificare i modi per aumentare la qualità e l'efficacia. L'evento è programmato da e per il team Scrum. Tutti dovrebbero essere coinvolti attivamente. Ha senso preparare i KPI che forniscono una comprensione della qualità e dell'efficacia del tuo team.

Passo 1 → Ispezionare

Molto probabilmente l'Agile Master facilita l'incontro (ma naturalmente questo può essere fatto anche da chiunque altro nello Scrum Team) e l'intero team ispeziona l'ultimo Sprint per quanto riguarda gli individui, le interazioni, i processi o gli strumenti, e la loro Definizione di Fatto. Si discutono le ipotesi e si esplorano le loro origini o le cause profonde. Lo Scrum Team discute i problemi incontrati nell'ultimo Sprint, e come sono stati risolti o devono essere risolti in futuro.

Passo 2 →Identificare i cambiamenti

Il team identifica i cambiamenti più utili per migliorare l'efficacia. I miglioramenti più impattanti devono essere affrontati il prima possibile. Si potrebbe differenziare tra Quick Wins e Big problems se si vuole. Comunque, i problemi che si possono affrontare presto potrebbero essere messi direttamente nel prossimo Sprint Backlog.

Passo 3 → Quick Wins

I Quick Wins sono problemi che possono essere risolti rapidamente. Il team discute le soluzioni, le impegna e le integra nel prossimo Sprint Backlog.

Passo 4 → Big problems

I Big problems, che non possono essere risolti facilmente o supportati da altre parti interessate sono più difficili da affrontare subito. Per risolvere questi problemi, potrebbe essere necessaria un'analisi dettagliata. Dato che la riunione è a tempo, la tua squadra ha bisogno di dare priorità ai Big problems in modo che quelli con la priorità più alta siano discussi e analizzati per primi. La prioritizzazione è fatta dalla squadra.

3) Risultati

- elementi di azione ‘retrospective’ prioritari e miglioramenti impegnati;

- elementi di miglioramento più impattanti affrontati il prima possibile;
- rafforzamento dello spirito di squadra e miglioramento dell'autogestione.

Sprint reporting

Lo Sprint Reporting è usato per valutare se le risorse dell'ultimo viaggio sono state usate in modo sufficiente ed efficiente.

Lavorare con i report la prima volta non è facile. Due cose importanti sono: il team deve registrare i tempi di lavoro su User Stories e Sub-tasks in JIRA e i team funzionali e i prodotti devono essere correttamente mantenuti in CGM SPOCK per ricevere report validi.

FIGURA 18 - ESEMPIO DI MEETING RILEVANTI PER UNO SPRINT A LIVELLO DI TEAM

4.2.4 Agile development

"La visione senza azione è solo un sogno, l'azione senza visione fa solo passare il tempo, e la visione con l'azione può cambiare il mondo" Nelson Mandela

1) La Vision è la forza motrice e il senso comune per lavorare su un prodotto. Il workshop aiuta a trovare la Vision e a raggiungere una comprensione comune tra colleghi del perché si sta facendo quello che si fa.

2) La visione di un progetto può avere un grande impatto sull'atteggiamento di un team verso un nuovo progetto. Se la visione del prodotto presentata può elettrizzare i colleghi, lavoreranno con determinazione ed entusiasmo. Inoltre, aiuta ogni membro del team a capire come i suoi compiti contribuiscono al quadro generale.

Passo 1 → Visione

Pensa alla motivazione per creare il prodotto. Cosa ci si aspetta dal nuovo prodotto? Quale cambiamento positivo porterà? Annotare un riassunto conciso dell'idea che descriva l'intenzione e la motivazione. Limitare la dichiarazione di Vision a due frasi - è solo una prima bozza di idee.

Passo 2 → Gruppo target

Nei passi seguenti, si deve riempire la Visione con informazioni dettagliate. Iniziare con il gruppo target. Chi sono i clienti target del nuovo prodotto? A quale mercato o segmento di mercato si rivolge il prodotto? È una buona idea descrivere il gruppo target nel modo più conciso possibile. Così il prodotto e il marketing possono essere più specifici e fornire maggiori benefici al cliente.

Passo 3 → Bisogni

Il compito chiave della visione è probabilmente la definizione dei bisogni. Quale problema risolve il prodotto? Quale beneficio fornisce? Se non si riesce a trasmettere il valore di un prodotto al team, come farà il cliente a riconoscerne il senso? Mettere in chiaro che il prodotto fornisce un beneficio distinto per ogni utente, altrimenti non spenderà soldi per esso.

Passo 4 → Prodotto

Per rendere tangibili i bisogni definiti, è una buona idea riassumere da tre a cinque caratteristiche principali del prodotto. Queste caratteristiche dovrebbero dimostrare chiaramente la proposta unica di vendita del tuo prodotto.

Passo 5 → Obiettivi di business

L'ultimo passo è quello di abbozzare i benefici del tuo prodotto per l'azienda. Spiegare perché vale la pena per l'azienda investire nel nuovo prodotto e come può

aiutare a raggiungere gli obiettivi di business. Di solito, si raggiungono gli stessi obiettivi, ad esempio aumentare le entrate, entrare in un nuovo mercato, ridurre i costi, sviluppare il marchio o ottenere un vantaggio tecnologico.

3) Risultati

- visione comune per il nuovo prodotto;
- colleghi motivati;
- entusiasmo nel lavoro.

Defininition of Done

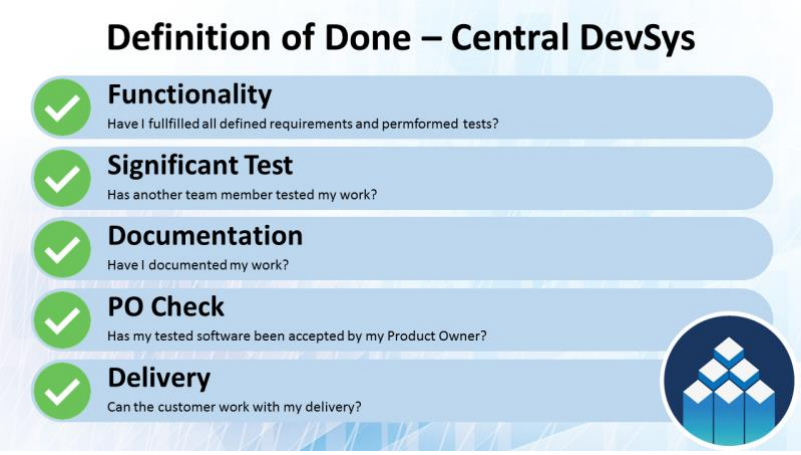

FIGURA 19 - DEFINIZIONE DI 'DONE'

1) Ogni membro del team ha compiti specifici durante il processo di sviluppo. Una comprensione comune di "Done" aiuta il team a non avere sorprese alla fine dello sprint.

"Senza obiettivi, e piani per raggiungerli, sei come una nave che è salpata senza direzione" Fitzhugh Dodson

2) Ogni membro del team ha compiti diversi, ma ha senso definire una comprensione comune di ciò che si intende per dire che un compito è "Done". Il risultato di questo workshop è una lista di controllo completa dei criteri necessari che devono essere soddisfatti per etichettare un compito con "Done". Questa lista assicura che solo le caratteristiche assolutamente completate siano consegnate al cliente.

Passo 1 → Definire le attività

La definizione di Done consiste in singoli criteri che si riferiscono ad attività recenti. In un primo passo, il team si accorda sulle attività principali che devono essere fatte per ogni caratteristica (es. test, integrazione continua, ecc.) e le annota sul Flipchart.

Passo 2 → Definire i criteri

Convertire le attività in criteri specifici e concisi. Il team deve decidere se l'aggiunta di metriche e criteri di realizzazione rende più facile rispettare la Definizione di Fatto o se la limitano troppo. Si deve anche pensare a diverse Definizioni di Fatto per diversi livelli. Ci sono criteri che devono essere soddisfatti in ogni sprint? Altri in ogni release? Altri per ogni compito? Dopo tutto, il team si impegna in un massimo di tre diverse checklist. Questa lista di controllo sarà onnipresente e ogni membro deve garantire che il suo lavoro soddisfi i criteri.

3) Risultati

- comprensione comune di “Done”;
- lista di controllo utile per ogni membro del team;
- riduzione dei costi di rielaborazione;
- ridotto rischio di incomprensioni e conflitti tra il team, il Product Owner e i clienti.

1. Ruoli e responsabilità: Livello di programma

General Manager (GM): ha la responsabilità generale di tutta la Business Unit. È il supervisore disciplinare di tutti i dipendenti assegnati alla Business Unit. Questo include le aree di lavoro Amministrazione e Vendite & Marketing così come Prodotto e Servizio & Supporto

Software Architect (SA): è il "maestro costruttore" dell'architettura del software. È responsabile di prodotti complessi e altamente collegati in rete. Manutenibilità, performance di runtime, qualità del codice, efficienza di sviluppo e affidabilità sono le priorità principali.

Release-Train-Engineer (RTE): supervisiona un Release Train che di solito è composto da 5 a 10 team agili.

Ux-Architect (UX): crea le specifiche generali, le guide di stile e le regole per le interfacce utente dei nostri prodotti. L'UXA crea la migliore filosofia utente possibile e la rispettiva navigazione, gli elementi utente (controlli) e la loro disposizione nell'UI dalle specifiche del SA e del PA. L'UXA assicura che i prodotti abbiano un look and feel uniforme, che CI/CD siano ancorati. La UXA assicura anche che i controlli siano uniformi e standardizzati.

Product Architect (PA): Il Product Architect trasforma una buona idea (visione del valore) nel concetto (l'architettura del prodotto) per un "prodotto cool". Si muove in modo ottimale all'interno del quadro stabilito dal business plan e dalle possibilità tecniche (architettura del software). A seconda delle dimensioni dell'area di sviluppo, il PA guida un team più o meno grande di dipendenti con i possibili ruoli: Product Owner, Business Analyst, Requirements Manager e PA Professional.

Quality Assurance Manager (QM): testa il software e segnala gli errori trovati nello strumento di tracciamento dei problemi ("ticket-system").

Team Leader Sviluppo (TL Dev): Nelle grandi organizzazioni può essere utile raggruppare diversi team di sviluppo sotto un TL DEV che, come supervisore responsabile dei Team Leaders, porta le loro responsabilità combinate ad un livello superiore.

2. Ruoli e responsabilità: Livello di team

Team-Leader (TL): guida un team al livello gerarchico più basso.

Product Owner (PO): rappresenta gli interessi e gli obiettivi dei PA in uno o più team agili. Il suo obiettivo è quello di massimizzare il valore commerciale del prodotto e il rilascio anticipato delle funzioni chiave, in modo da ottenere un rapido ritorno dell'investimento.

Agile Master (AM): supervisiona uno o più processi agili e supporta i team associati secondo i principi di Scrum o altri processi agili.

Sviluppatore di software (SWD): sviluppa il software richiesto, impiegando principi di codice pulito, assicurando così una programmazione efficiente e di alta qualità. Questi principi dovrebbero essere delineati nelle linee guida di programmazione del suo dipartimento di prodotto. Le priorità dei problemi (backlog items, bug dei clienti) sono stabilite dal PA/PO e definite nel processo di manutenzione (SLAs → service level agreements) per i bug provenienti dai clienti (o dai centri di assistenza per il prodotto).

Scrittore tecnico (TW): In consultazione con il Product Architect o il Product Owner e i team di sviluppo, il Technical Writer (TW) è responsabile della produzione della documentazione tecnica per i prodotti e i sistemi, compresi gli aiuti online, gli aiuti scritti e i manuali utente.

4.2.5 JIRA

Jira è il framework di Athlassian usato da CGM per gestire i task di sviluppo e il flusso di lavoro di tutti i progetti.

Stima dei problemi JIRA

"Un principio chiave della stima agile è che stimiamo la dimensione ma deriviamo la durata" Mike Cohn

Questo play contiene una breve panoramica della gerarchia di collegamento, delle stime di tempo e a quale punto un problema di Jira dovrebbe avere una certa maturità.

1) Gerarchia dei ticket

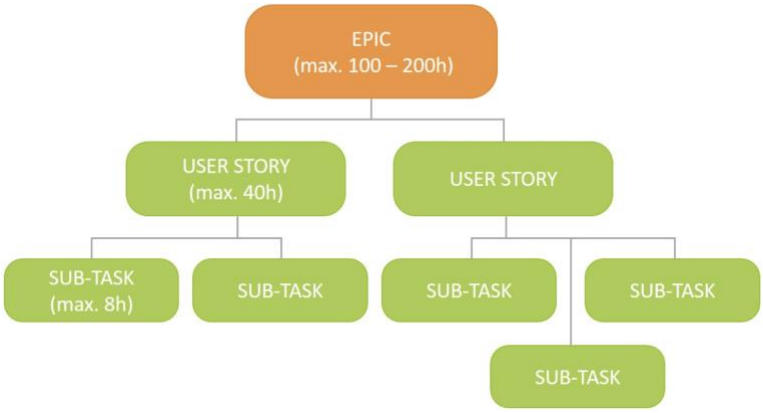

FIGURA 20 - GERARCHIA DI COLLEGAMENTO TICKET

2) Processo di stima

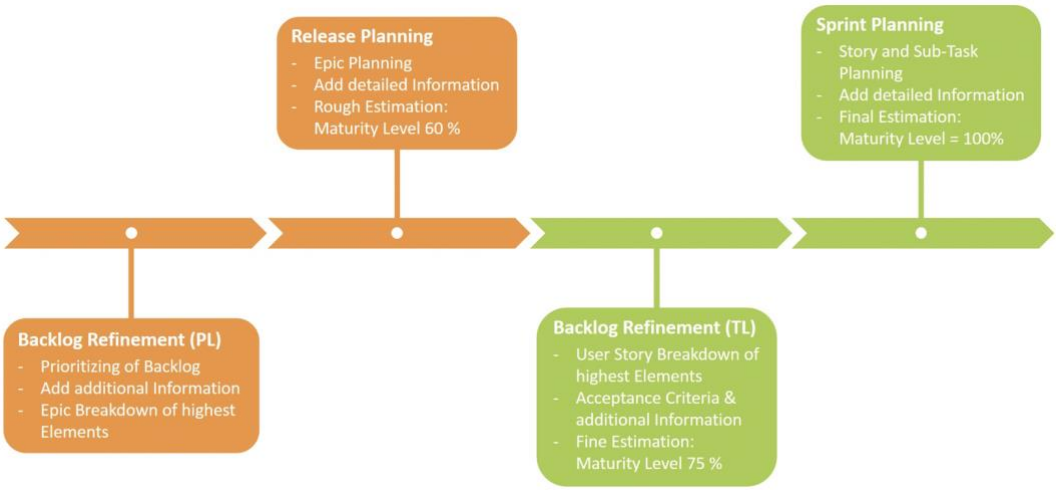

FIGURA 21 - STIMA DI TEMPO

3) Livello di confidenza della stima

A seconda dello stato del problema è necessario raggiungere un certo livello di fiducia nella stima. Il livello di fiducia dovrebbe aiutarti a migliorare la stima in uno stato iniziale (figura 6).

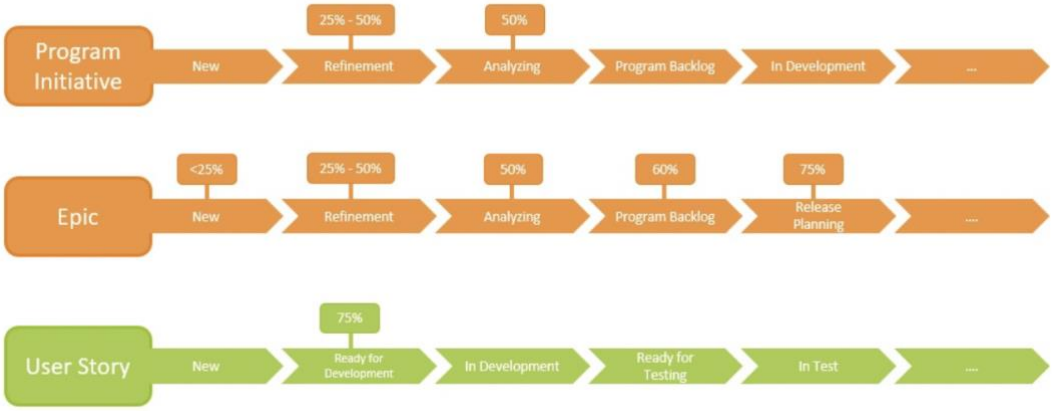

FIGURA 22 - ANALISI LIVELLI DI CONFIDENZA DELLA STIMA

5) Perfezionamento del backlog (livello di programma): il team si riunisce per preparare il backlog del prodotto. Danno priorità al backlog, aggiungono nuove informazioni e suddividono le initiatives in epics più concise. Le singole epics dovrebbero essere di circa 120-200 ore.

6) Pianificazione del rilascio: nella riunione, il team pianifica il lavoro per il prossimo release. Quindi, si scompone l'epics esistente in User Stories che non dovrebbero essere più grandi di 40 ore. Una prima stima per ogni User Story deve essere fatta. Il livello di maturità dovrebbe essere superiore al 60% per assicurare un'alta efficienza nelle prossime riunioni di pianificazione.

7) Perfezionamento del backlog (a livello di team): il team si riunisce per perfezionare le User Story create nel backlog del prodotto. Questo include specificarle aggiungendo criteri di accettazione, dipendenze e informazioni aggiuntive (come allegati, ecc.). Inoltre, la prima stima dal Release Planning viene aggiornata con maggiore cura e le User Stories vengono suddivise in Sotto-attività che devono essere più piccole o uguali a 8 h. Lo stato dovrebbe essere cambiato in "Pronto per lo sviluppo".

8) Pianificazione dello sprint: in questa fase finale del processo di stima, il team controlla le User Stories e i Sub-Tasks esistenti e li pianifica nel prossimo Sprint. Bisogna controllare se i temi pianificati sono conformi a tutti i requisiti. Le User Stories e i Sub-Tasks non devono superare le ore di lavoro disponibili per lo sprint successivo e devono contenere i criteri di accettazione e tutti i campi richiesti (come

"Fix Version", ecc.). Questa è l'ultima possibilità di aggiustare la stima. Assicurarsi che la stima sia realistica. Il livello di maturità alla fine della riunione deve essere del 100%.

9) Stima e prenotazione del tempo

Ci sono due possibilità per stimare gli sforzi e tracciare le ore di lavoro su questioni a livello di team. Questi problemi sono Bug, User Stories e Sub-Tasks e ToDo's.

Tutti gli sforzi sono stimati e tracciati su User Stories / Bug / ToDo. La stima totale è inserita nel campo "Stima originale" e "Stima rimanente" e contiene il tempo stimato per completare la questione con tutti i compiti necessari (per esempio includendo il test, il deploy, ecc.). Tutti i worklogs di tutti i colleghi che lavorano su questo problema specifico sono registrati sulla User Story / Bug / ToDo. Tutti gli sforzi sono divisi in Sotto-attività. Tutti i worklogs devono essere tracciati sul corrispondente Sub-Task

Gerarchia e collegamenti in JIRA

"Le storie agiscono come una 'lingua pidgin' dove entrambe le parti (utenti e sviluppatori) possono essere abbastanza d'accordo da lavorare insieme in modo efficace" Bill Wake

Questa descrizione vi mostrerà come i problemi di Jira sono collegati attraverso i diversi livelli dello Scaled Agile Framework (figura 7).

Gerarchia

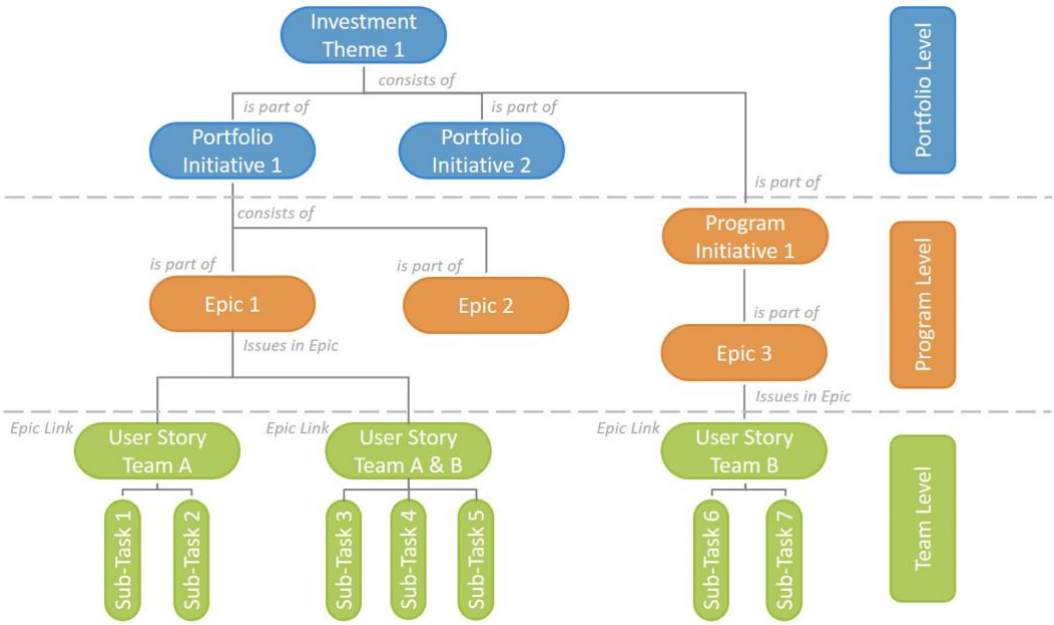

FIGURA 23 - COLLEGAMENTI LIVELLI DI SCALED AGILE FRAMEWORK

1) Livello di portafoglio

1.1 Tema di investimento:

Rappresenta un budget (di sviluppo) che viene fornito per scopi definiti all'interno del Tema. Investimento a lungo termine che normalmente copre diversi trimestri. Mostra la strategia di prodotto dell'azienda. Consiste in iniziative di portafoglio (queste iniziative sono collegate in JIRA da "fa parte di" o "consiste di"). È sempre assegnato a un Senior Manager che è responsabile ed è gestito da un General Manager.

Collegamento: Tema di investimento → consiste in → Iniziativa di portafoglio → consiste in → Epic (a livello di programma)

1.2 Iniziativa di portafoglio:

Diviso in “Portfolio Business Initiatives” e “Portfolio Architectural Initiatives”.

“Portfolio Business Initiatives” descrive un ampio sviluppo del prodotto con l'obiettivo di acquisire più clienti, aumentare la soddisfazione del cliente e aumentare le vendite. Sono create e documentate da un “Product Architect”.

Le “Portfolio Architectural Initiatives” si concentrano sull'ulteriore sviluppo di un prodotto al fine di aumentare la stabilità, la scalabilità e la manutenibilità. Sono create e documentate da un “Software Architect”.

Non solo descrive i requisiti per gli sviluppi del prodotto, ma contiene anche dati economici per la considerazione di una realizzazione. Non deve essere realizzata entro una release, ma può durare per diversi trimestri. Un'iniziativa è sempre collegata a un tema di investimento

Collegamento: Tema d'investimento → consiste in → Iniziativa di portafoglio → consiste in → Epic (a livello di programma)

2) Livello di programma

2.1 Iniziativa di programma:

Si divide in “Program Business Initiative” e “Program Architectural Initiatives”.

“Program Business Initiative” descrive uno sviluppo completo del prodotto con l'obiettivo di acquisire più clienti, aumentare la soddisfazione del cliente e incrementare le vendite. Sono normalmente create e documentate da un “Product Architect”.

Le “Program Architectural Initiatives” si concentrano sul miglioramento di un prodotto al fine di migliorarne la stabilità, la scalabilità o la manutenzione. Sono normalmente create e documentate da un “Software Architect”.

Descrive i requisiti per lo sviluppo del prodotto, ma contiene anche dati economici per la considerazione di una realizzazione. Non deve essere realizzato entro una release, ma può durare per diversi trimestri. È sempre collegato ad un Tema di Investimento.

Collegamento: Tema d'investimento → consiste in → Iniziativa di programma → consiste in → Epic (a livello di programma)

2.2 Epic:

Rappresenta sforzi di 100-200 ore di carico di lavoro e dovrebbe essere finito entro una release. I requisiti sono gestiti dal “Product Architect” o dal “Product Owner”. È

un insieme di requisiti o un singolo requisito che può essere realizzato all'interno di una release e potenzialmente spedito. Appartiene sempre esattamente a un Portafoglio / Iniziativa di programma. Contiene una dichiarazione di valore Epic come pagina Confluence collegata.

Collegamenti al livello di Portfolio/Programma: Ogni Epic deve essere collegata a un Portfolio / Programma Iniziativa o Tema di investimento. Iniziativa / Tema → consiste di → Epic

Collegamenti a livello di team: Ogni Epic deve essere collegato ad una o più storie. Le storie che appartengono ad un Epic possono esistere in diversi Backlogs di Team. Queste assegnazioni sono espresse con un campo separato e non con collegamenti espliciti: Ogni Epic ha un campo chiamato "Epic Name". All'interno di ogni storia, puoi inserire questo alias nel campo "Epic Link" per ricevere un'assegnazione tra storia ed Epic.

3) Livello di team

3.1 Storia utente:

Rappresenta un carico di lavoro fino a 40h. È una funzionalità o un requisito del prodotto. È creata e gestita dal “Product Architect” o dal “Product Owner”. Contiene almeno una "User Story Statement" nel modello "Come <ruolo> voglio <obiettivo/desiderio> per <beneficio>". Contiene criteri di accettazione che definiscono chiaramente quando la User Story è valutata come "done". Dovrebbe essere finita entro uno Sprint.

Collegamento: Le User Stories → hanno un collegamento a → Epic

3.2 Sub-Task:

Rappresenta uno sforzo di 8 ore al massimo. Collegato ad una specifica User Story. Descrive un carico di lavoro che può essere gestito da uno sviluppatore. Supporta il soddisfacimento dei criteri di accettazione della User Story.

Collegamento: Sub-Tasks → fa parte di → User Story

Name	When to use
SAFe Team Level	Older template which contains extensive fields and is to be used if your product is a medical device.
Team Level 2.0	Teams which contribute to several Program Level Projects and Products.
SAFe Program Level	Older template which contains extensive fields and is to be used if your product is a medical device.
Program Level 2.0	A simplified configuration of the Program Level. To be used if the program feeds multiple team levels or acts as a cross-BU level.
Combined PL/TL 2.0	Development organisation which primarily work on one product which is not often influenced by other programs. Within the project you can handle multiple teams with the help of different boards.
SAFe Portfolio Level	This template provides guidance in working with Investment Themes and Portfolio Initiatives on the Portfolio Level.
SAFe Request	This template is designed for managing ideas and requests before they are taken into the Program / Team Level.
SAFe Management	This template is used for lightweight task management intended for management groups or teams.
Management 2.0	COMING SOON ... This template is used for lightweight task management intended for management groups or teams and is an improved version of "SAFe Management".

FIGURA 24 - JIRA TEMPLATE OVERVIEW

4.2 Esperienza in azienda

Ho partecipato alla gestione del progetto che CGM ha sviluppato per introdurre in Francia il software XDent, che è un Dental Information System (sistema informatico per la gestione di uno studio dentistico), per gestire la verticalizzazione del software usato in Italia ed adattarlo alle esigenze del mercato francese.

Ho analizzato il flusso di lavoro dei team, durante l'esecuzione del progetto, che è stato strutturato utilizzando la metodologia sopra descritta e le "best practices" individuate da CGM, sopra descritte.

Queste le principali fasi del progetto:

1. sono state analizzate le esigenze specifiche dei dentisti francesi e questo è avvenuto instaurando una collaborazione e condividendo le conoscenze fra team Italiano e team Francese;
2. sono stati analizzati i prodotti dei competior francesi di CGM;
3. è stato fatto il "business plan" per capire se lo sforzo di sviluppo poteva essere compatibile con i possibili ricavi e per poter individuare il valore che lo sviluppo del software poteva avere per il business dell'azienda;
4. è stata fatta la progettazione delle attività di sviluppo utilizzando il framework Jira con la metodologia Agile;
5. sono state pianificate le attività di marketing sia sui social sia su Internet sia individuando possibili distributori del prodotto;
6. è stato avviato il programma di training delle risorse sales per poter vendere il prodotto.

Il tutto è stato condotto utilizzando i meeting previsti e l'organizzazione che hanno permesso ai team, che avrebbero dovuto portare avanti il lavoro di sviluppo necessario, di avere una chiara visione dell'obiettivo finale e condividere il valore del progetto e la responsabilità per l'importanza di ciò che si portava avanti.

Ho visto applicato anche lo scaled Agile, visto che i team che hanno portato avanti lo sviluppo del software erano distribuiti in Italia e in Francia e come questi meeting potessero aiutare a convogliare il lavoro delle squadre verso il successo.

I team sono stati motivati e ho potuto vedere all'opera Scrum Master e Product Owner svolgere le proprie funzioni nel motivare i team e nel condividere il valore dello sviluppo, nel preparare il backlog di prodotto con le user story, nel prioritizzare le medesime; vedere anche il team preparare il backlog degli sprint, fare le stime e avanzare nel lavoro programmat.

Sono stata protagonista anche nel compilare il verbale dei 'Daily Scrum', degli 'Sprint Planning' e dei 'Retrospective meeting' in cui ho avuto modo di apprendere come la metodologia ti aiuta a reagire nel momento in cui si ha una deviazione nella pianificazione delle attività stabilite dopo la fase di analisi, nel momento in cui ci si è accorti che il mercato francese richiedeva lo sviluppo di alcune features non previste in fase di analisi.

Soprattutto ho potuto osservare come i ‘Retrospective meeting’ sono, secondo me, il momento cruciale di questa metodologia visto che in questo evento si celebrano i successi della squadra, di ogni singolo componente del team, ma soprattutto ci si rende conto degli errori commessi e si trascrivono tutte le difficoltà che hanno condotto a questi risultati per evitare che questo possa risuccedere.

5. Conclusione

Le ragioni per cui adottare Agile e i benefici derivanti dalla sua adozione

Secondo quanto riportato da un report del 2018 (*La Realizzazione Dei Benefici Secondo Le Metodologie Di Project Management Tradizionale e Agile*, 2018) circa il 52% delle persone intervistate hanno dischiarato che nelle proprie aziende più della metà dei team adotta la metodologia Agile. Alla domanda riguardo le ragioni per cui adottare Agile, gli intervistati, che avevano la possibilità di indicare più di un'opzione, hanno riconosciuto come principali motivazioni:

- il fatto che Agile permette di velocizzare la release del software (75%),
- migliori le abilità di gestire le priorità che cambiano (64%),
- aumenti la produttività (55%),
- accresca l'allineamento tra business e IT (49%),
- incrementi la qualità del software (49%) e la visibilità del progetto (49%).

Gli intervistati riferiscono anche che le imprese stanno attribuendo il successo di Agile nei progetti e il 61% riporta che la maggioranza o tutti i loro progetti Agile hanno avuto successo.

Parlando ora dei benefici dedotti dall'adozione di Agile gli intervistati, che potevano indicare più di un'opzione, hanno dato risalto:

- al fatto che incrementi l'abilità di gestire cambiamenti di priorità (71%),
- nella visibilità del progetto (66%),
- nell'allineamento tra business e IT (65%),
- nella velocità di consegna (e conseguente riduzione del time to market) (62%),
- nella produttività del team (61%) e
- nel morale del team (61%).

In termini di maturità organizzativa in ambienti Agile, l'analisi mostra che ci sono ancora molti margini di miglioramento. Infatti, solo il 12% degli intervistati crede che le loro aziende abbiano un alto livello di competenza nelle pratiche Agile, e solo

il 4% crede che queste pratiche stiano permettendo loro di essere più adattabili alle condizioni del mercato. Tuttavia, sembra che il 59% degli intervistati stia usando Agile ma stia ancora lavorando per sviluppare le competenze necessarie, mentre solo il 2% non ha iniziato a usare Agile.

Individuando dei KPI chiave per misurare la performance di Agile rispetto ad una metodologia tradizione, elenchiamo:

1. **la soddisfazione del cliente**: concentrandosi sui bisogni del cliente, gli approcci iterativi sono in grado di valutare rapidamente le esigenze dei clienti e, nel peggiore dei casi, cambiare rapidamente le attività del progetto. Infatti, il 54% degli intervistati ritiene che Agile permetta loro di soddisfare meglio le esigenze dei clienti e di aumentare la loro soddisfazione;
2. **la soddisfazione del team**: Agile, secondo il 78% degli intervistati, permette migliori relazioni interpersonali all'interno dei team di lavoro rispetto a Waterfall. Questo è probabilmente dovuto alla continua interazione tra tutte le parti coinvolte, così come gli eventi e i ruoli ben definiti da metodologie come Scrum;
3. **la qualità del prodotto**: i rispondenti hanno dato prevalentemente (76%) una valutazione molto positiva. I fattori collegati alla qualità sembrano essere uno dei punti chiave del passaggio dai metodi tradizionali Waterfall all'Agile. Come espresso nella prima parte di questa tesi l'implementazione di metodi iterativi permette una gestione differente della qualità e gli intervistati sembrano riconoscere che i prodotti ottenuti con progetti Agile abbiano una qualità superiore;
4. **efficienza del processo**: circa l'80% degli intervistati reputano la produttività di agile superiore a quella, ad esempio, delle metodologie Waterfall.

Confronto dei tassi di successo dei progetti Waterfall e Agile

I risultati del più recente Standish Group Chaos Study (Mersino, 2018) mostrano i tassi di successo e fallimento dei progetti Waterfall e Agile. Non dovrebbe sorprendere che i progetti Agile hanno statisticamente 2 volte più probabilità di successo e 1/3 di probabilità in meno di fallire rispetto ai progetti Waterfall. Lo Standish Group ha condotto indagini sui tassi di successo e fallimento dei progetti IT

ogni 2 anni dal 1994. I dati qui inclusi sono del rapporto più recente pubblicato nel 2018 (figura 25).

Inizialmente, le statistiche di successo e fallimento erano piuttosto scarse, con tassi di successo dei progetti IT misurati a meno del 20%. Per fortuna le cose sono un po' migliorate. Il rapporto più recente (2018) dello Standish Group mostra tassi di successo dei progetti tecnologici al 36% con progetti agili che hanno successo più frequentemente. Credo che sia il successo dei progetti agili che sta sollevando tutto il successo dei progetti IT.

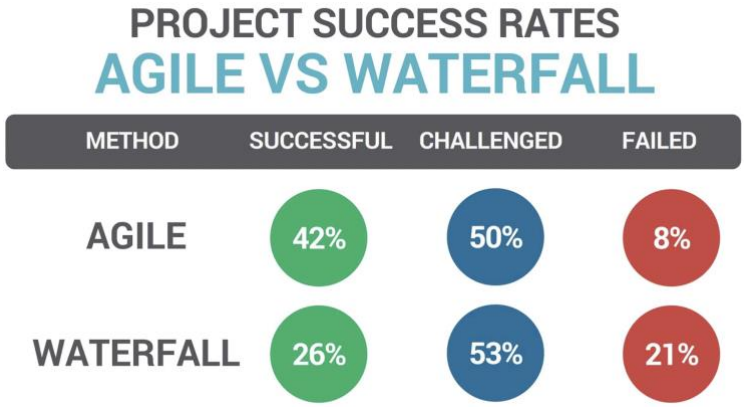

FIGURA 25 - CONFRONTO SUCCESSI E FALLIMENTI

Un'altra scoperta chiave dello Standish Group è che i progetti più grandi hanno tassi di fallimento più alti. Questo non dovrebbe essere una sorpresa per nessuno. Quello che mi ha sorpreso è stata la misura in cui i progetti più piccoli hanno ridotto il rischio (vedi il grafico dei tassi di fallimento dei progetti qui sotto). Il risultato chiave qui è che i progetti più piccoli massimizzeranno il successo dei progetti agili.

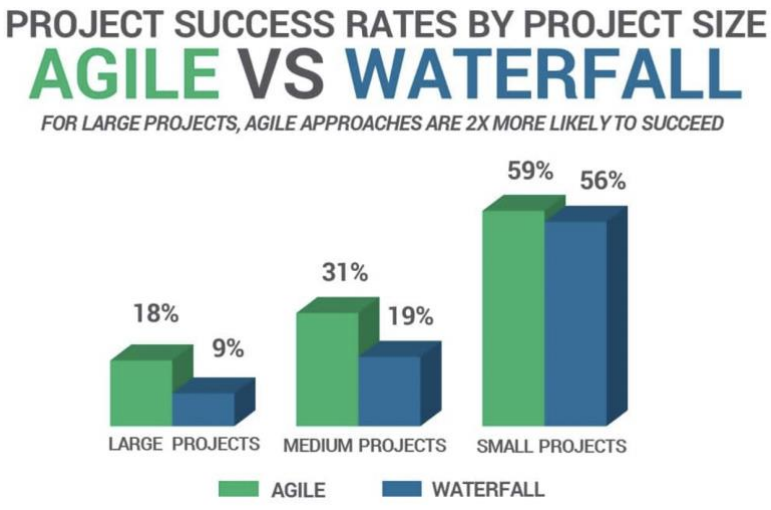

FIGURA 26 - CONFRONTO GRANDEZZE PROGETTI

Guardando il grafico, si può vedere che i progetti Agile hanno un vantaggio significativo sui progetti Waterfall per ogni dimensione del progetto. Dei due fattori del progetto, la dimensione e l'approccio, lo Standish Group dice che la dimensione ha un impatto maggiore sui tassi di fallimento del progetto rispetto all'agilità. Entrambi hanno l'impatto maggiore.

La diffusione dell'Agile a livello internazionale e in Italia

Se negli anni passati l'Agile veniva visto come una metodologia applicabile in contesti limitati e a basso rischio, oggi questi vincoli sono stati superati e i CIO concordano che di questo approccio possano beneficiarne tutti i progetti. Si parlava qualche anno fa di IT-bimodale: una direzione IT capace di operare dividendo i progetti in due categorie, quelli che necessitavano velocità e innovazione e quelli che richiedevano sicurezza e continuità. Due tipologie di progetti, con due modalità organizzative, processi e metodi differenti. Questo approccio, alla prova dei fatti, si è rivelato miope e poco applicabile. Oggi questo concetto è del tutto superato e le aziende vedono nell'Agile una modalità per guidare una trasformazione complessiva del proprio modo di operare.

Secondo il 12° State of Agile Report (survey globale sull'adozione dell'Agile) del 2018, nel 52% delle organizzazioni più di metà dei team adotta l'Agile, e solo il 2% non ha team Agile.

In Italia, secondo il Tavolo di lavoro Agile degli Osservatori Digital Innovation, la situazione è diversa ma in forte crescita: se nel 2017 il 20% delle imprese dichiarava di non utilizzare in alcun modo l'Agile, nel 2018 questa percentuale era scesa al 10%, mentre era salita dal 2% al 10% la percentuale di aziende che è completamente Agile (figura 6). Sembra quindi che anche in Italia si siano vinte le iniziali ritrosie e diffidenze e oggi le aziende cerchino di recuperare velocemente terreno.

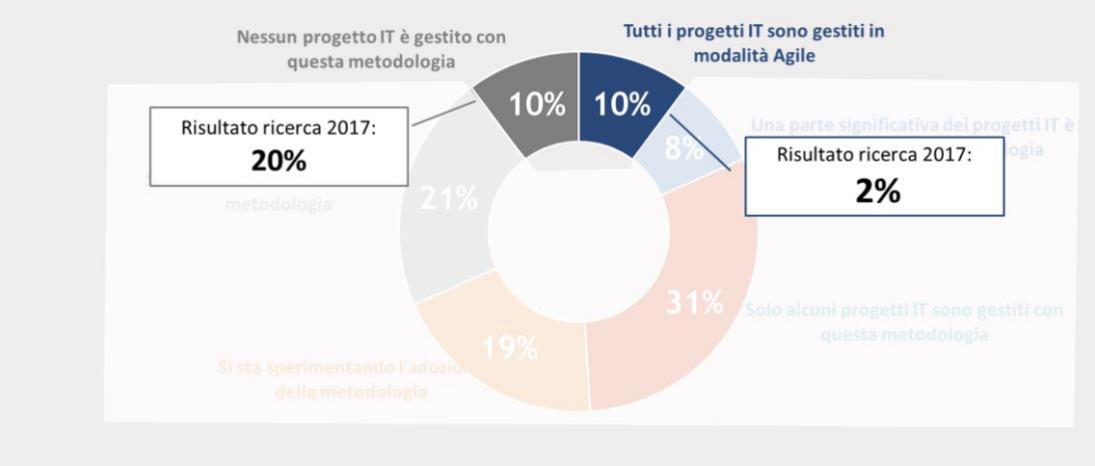

FIGURA 27 - ADOZIONE DELLA METODOLOGIA AGILE IN ITALIA

Allegati

Allegato 1 - Tabella confronto tra struttura organizzativa Tradizione e Autonomia condivisa

	Predizione e controllo	**Autonomia condivisa**
Struttura organizzativa	Top-down, gerarchica, autoreferenziale	Sistema, orizzontale, focalizzata sul cliente
Come vengono prese le decisioni	Dal management separato dal lavoro produttivo	Integrandole nel lavoro produttivo e distribuite
Sistemi di misura	Budget, target, standard	Capacità e variabilità correlate allo scopo
Disegno organizzativo	Funzionale	Basato sulla domanda, il valore e il flusso
Attitudine nei confronti dei clienti	Contrattuale	Quello che è importante
Attidudine nei confronti dei fornitori	Contrattuale	Cooperativa
Ruolo del management	Gestire persone e budget, controllo	Agire sul sistema, apprendimento continuo
Cambiamenti	Reattiva per progetti	Adattivi su tutto il sistema
Approccio	Quello che è conveniente per l'organizzazione dal punto di vista della gerarchia funzionale	Quello che vogliono i clienti dal punto di vista della risposta di tutto il sistema
Strategia produttiva	Economie di scala	Economie di flusso

Bibliografia

1. *Metodologia Agile: definizione, principi e obiettivi*. (2019). AGILE SCHOOL. https://www.agile-school.com/blog/metodologia-agile-definizione-principi-e-obiettivi
2. *Scrum - what it is, how it works, and why it's awesome*. (2018). Atlassian. https://www.atlassian.com/it/agile/scrum
3. Lisca, F. (2017). *Il quinto paradigma: Come trasformare la propria azienda in un'organizzazione agile (Italian Edition)*. Franco Angeli Edizioni.
4. A. (2009). *Cambiare. Come progettare prodotti e servizi di successo in un mondo di incertezza*. Tecniche Nuove.
5. *Search agile*. (2020). Harvard Business Review. https://hbr.org/search?search_type=&term=agile+
6. Salati, M. E., & Leoni, A. (2021). *Neuroscienze e sviluppo (del) personale (Italian Edition)*. goWare & Guerini Next.
7. *Manifesto for Agile Software Development*. (2005). Manifesto for Agile Software Development. https://agilemanifesto.org
8. *Planning poker, stima agile dei requisiti*. (2021, May 2). Agile Way. https://www.agileway.it/planning-poker-stima-agile-requisiti/
9. *La realizzazione dei benefici secondo le metodologie di Project Management tradizionale e agile*. (2018). Alessio Perri.
10. Mersino, A. (2018, April 1). *Post author: Anthony Mersino*. Vitality Chicago Inc. https://vitalitychicago.com/blog/agile-projects-are-more-successful-traditional-projects/