Università degli studi di Padova

Dipartimento di Tecnica e Gestione dei Sistemi Industriali

Corso di Laurea Triennale in Ingegneria Gestionale

GESTIONE DI PROGETTI ATTRAVERSO LA TECNICA SCRUM

RELATORE: PROF. Enrico Scarso

LAUREANDO: Florentina Anghel

ANNO ACCADEMICO: 2015-2016

Alla mia famiglia e

soprattutto ai miei genitori.

Indice

Sommario **5**

Introduzione **6**

Capitolo 1: Gestione tradizionale dei progetti **7**

1.Premessa 7

2.Approccio tradizionale: PMBOK 8

3.Approccio Waterfall vs Metodologia Agile 12

Capitolo 2: Metodologia Scrum **14**

1.Cenni storici 14

2.I fondamenti teorici 17

3.Scrum Team 18

3.1. Il Product Owner 19

3.2. Il Team di Sviluppo 19

3.3. Lo Scrum Master 20

4.Gli Eventi di Scrum 21

4.1. Lo Sprint 22

4.2. Sprint Planning 22

4.3.Daily Scrum 24

4.4.Sprint Review 25

4.5.Sprint Retrospective 26

5.Gli Artefatti di Scrum 26

5.1.Product Backlog 27

5.2.Sprint Backlog 28

5.3.Incremento 28

Capitolo 3: Gestire i progetti con Scrum **29**

1.Preparazione del progetto 29

2.Pianificare uno sprint 32

2.1.Scegliere le storie utente 34

3.Eseguire uno sprint 35

4.Tenere traccia del progetto 38

4.1.Preparare lo sprint successivo 38
4.2.Tenere traccia dello stato di avanzamento del rilascio 39
4.3.Finire il rilascio 41

Capitolo 4: Casi Studio 42
1.Intel Corporation 42
1.1.Sommario 42
1.2.Introduzione 42
1.3.Fase 1: Preparazione 43
1.4.Sopravvivenza dei team 44
1.5.Preparazione della Produzione 45
1.6.Retrospettiva 46
1.7.Risultati 49
2.Ferrovie olandesi 50
2.1.Background 50
2.2. Impostazione 51
2.3.Espansione del team 51
2.4.Team per raccogliere informazioni 53
2.5.Documentazione 53
2.6.Esigenze 53
2.7.Testare 53
2.8.Risultati 54
2.9.Conclusione 54

Conclusione 55

Bibliografia 57

Sommario

Pianificare, conoscere e prevedere sono aspetti chiave per l'abbassamento del rischio di fallimento di un'impresa o di un progetto. Per non incorrere in tali rischi una impresa moderna deve pertanto affidarsi al project management, sia esso basato sul modello Waterfall o Agile.

La tesi tratta l'applicazione del modello Scrum nel project management e si presenta divisa in 4 parti. Nella prima e seconda parte viene discusso il concetto di project management in generale, analizzando l'evoluzione delle pratiche a partire dal modello a cascata fino ad arrivare allo Scrum Agile. Nella terza parte si affronta in dettaglio la gestione dei progetti tramite Scrum, per poi presentare nella quarta parte due casi studio di applicazione di questo approccio.

Introduzione

Il Project Management, ovvero l'insieme delle tecniche e degli approcci che consentono di organizzare, pianificare, controllare e migliorare la realizzazione di un progetto, può essere come uno strumento oggi indispensabile.

Nell'abito del Project Management si possono individuare oggi due tipologie di processi di sviluppo di un nuovo progetto: processi di sviluppo classici e processi di sviluppo agili. Quando si considera un processo di sviluppo classico ci si riferisce a modelli come quello a cascata, dove la modalità di lavoro è fortemente orientata all'esecuzione di passi sequenziali.
I processi di sviluppo Agili, come ad esempio Extreme Programming (XP) o SCRUM, si contrappongono a quelli classici. Questi modelli sono composti da una serie di fasi iterative, dove viene presa in considerazione una parte di progetto alla volta, piccola e ben definita. La presente tesi ha inteso analizzare i due approcci allo scopo di valutare i vantaggi offerti dal modello Scrum.

La tesi si presenta divisa in 4 parti:

Nel capitolo 1 si presenta il concetto di Project Management in generale, per poi passare alle caratteristiche dell'approccio Waterfall e al perché c'è bisogno di un approccio maggiormente agile.

Il capitolo 2 tratta in dettaglio le ragioni che hanno portato alla nascita del modello Scrum, analizza le sue fondamenta e i principi di base.

Il capitolo 3 illustra le modalità che si devono seguire per sviluppare un progetto secondo i dettami del modello Scrum, partendo dalla sua preparazione e proseguendo con la pianificazione ed esecuzione.

Nel capitolo 4 si presentano due casi studio relativi all'applicazione del modello Scrum, mettendo in evidenza i pregi dell'approccio e le lezioni apprese dalla sua applicazione.
.

CAPITOLO 1

Gestione tradizionale dei progetti

1.Premessa

Il Project Management è una metodologia gestionale orientato ai risultati. Consiste nella "gestione sistemica di un'impresa complessa, unica e di durata determinata, rivolta al raggiungimento di un obiettivo chiaro e predefinito mediante un processo continuo di pianificazione e controllo di risorse differenziate e con vincoli interdipendenti di costi – tempi - qualità". (Archibald, 2000)

Il Project Management come lo si intende oggi si è affermato circa cinquant'anni fa, quando le aziende iniziarono ad apprezzare i vantaggi del lavoro organizzato per progetti e a comprendere quanto fosse importante riuscire a far lavorare in modo coordinato tra di loro più reparti con diverse competenze e specializzazioni.
Fino alla metà degli anni '60 del Novecento il modello di organizzazione del lavoro prevalente nell'industria meccanica era quello "tayloristico", dal nome dell'ingegnere statunitense Frederick W. Taylor, che alla fine dell'Ottocento teorizza l'organizzazione scientifica del lavoro: al fine di ottenere un incremento della produttività, il suo metodo prevedeva la scomposizione delle mansioni, l'individuazione di compiti elementari ben definiti da affidare a ciascun esecutore, il calcolo della loro misura e la programmazione della produzione.

Un collaboratore di Taylor, Henry Gantt, studiò in dettaglio l'ordine delle operazioni nel lavoro, definendo quello che oggi è conosciuto come Diagramma di Gantt: uno strumento grafico per la rappresentazione sull'asse temporale delle attività che concorrono al completamento di un progetto, permettendo la programmazione e il controllo sull'avanzamento.

In genere i fattori propulsivi del Project management sono riconducibili a:

- la presenza di un progetto importante o la compresenza di più piccoli progetti da gestire contemporaneamente;
- le crescenti aspettative dei clienti.

A sua volta un progetto è uno sforzo articolato e caratterizzato da risultati specifici in termini di prodotti consegnati (deliverables), prodotti/servizi forniti (scope), tempi impiegati (time) e budget necessari a produrre i risultati attesi (cost). Scope, time &

cost definiscono un insieme di vincoli che insieme con la qualità di progetto costituiscono il “perimetro” entro cui il progetto deve essere gestito.
Tipicamente un progetto si articola in una serie di fasi che costituiscono il suo ciclo di vita:

- Definizione e Avvio;
- Pianificazione;
- Realizzazione;
- Chiusura.

La gestione del ciclo di vita di un progetto può essere affrontata con approcci e modelli diversi di Project Management. Gli approcci utilizzati consistono in diversi approcci metodologici adottabili per la gestione delle attività di un progetto, che includono gli approcci agili, interattivi, incrementali e basati sulla successione di fasi predefinite. Indipendentemente dall'approccio utilizzato, una particolare attenzione va dedicata alla definizione chiara degli scopi/obiettivi del progetto e delle loro implicazioni; anche la definizione chiara dei ruoli e delle responsabilità di tutti gli attori coinvolti, inclusi i committenti, riveste un’importanza decisiva per il successo del progetto.

2.Approccio tradizionale: PMBOK

Il Project Management Body of Knowledge (PMBOK) descrive l’insieme delle prassi standard per la gestione di progetti così come definite dal Project Management Institute, che è il principale organismo internazionale di standardizzazione in materia di Project Management.

Il PMBOK prevede un approccio “waterfall” che individua uno sviluppo sequenziale di fasi che descrivono il ciclo di vita di un progetto e, parallelamente, il ciclo di vita del project management che ne governa lo sviluppo.

Il modello a cascata (Waterfall Model), chiamato anche Classic Life Cycle, è un approccio sequenziale e lineare allo sviluppo software. Esso trova le sue origini negli anni ’50, quando l’attività di sviluppo di software iniziò ad affermarsi come una vera e propria attività di produzione industriale. A quel tempo, non essendo presente nessuna metodologia di realizzazione software, gli sviluppatori si ispirarono ai processi di produzione manifatturiera e alle industrie di costruzione, per ottenere una metodologia che potesse essere applicata allo sviluppo di codice in modo ordinato e meno caotico.

Le prime tracce del modello a cascata si possono incontrare in una pubblicazione del 1956 di Herbert D. Benington. In questo articolo egli descrive una struttura sequenziale formata da diverse fasi, che è stata impiegata per lo sviluppo del complesso sistema S.A.G.E. (Semi-Automatic Ground Environment) per la difesa americana.

Successivamente in un articolo del 1970, l'informatico Winston Royce descrive formalmente il modello a cascata. Anche se nell'articolo Royce non cita mai la parola "cascata", tale metodologia è conosciuta in tutto il mondo con questo nome a causa della particolare struttura delle varie attività che la compongono, dove il flusso di sviluppo scorre in maniera lineare da una fase a quella successiva. Ciò significa che l'output prodotto dal primo stadio, sarà l'input per quello che segue. L'esecuzione lineare di tutte le fasi produce in output il prodotto software finale.

Il modello waterfall si basa fortemente sul concetto di Big Design Up Front ovvero su una "grande progettazione a monte": se vengono rilevati difetti o imperfezioni nelle fasi iniziali di produzione del software, tali difetti saranno più facilmente eliminabili; sarà inoltre più economico in termini monetari e temporali correggere un difetto in fase di design piuttosto che in fase di implementazione, in quanto nelle ultime fasi si avrà ormai sprecato tempo e denaro nella realizzare una soluzione difettosa o poco performante.

Un altro aspetto caratterizzante il modello a cascata è il fatto che tale approccio considera di fondamentale importanza la documentazione generata dalle varie fasi, come affermato dallo stesso Royce.

L'idea che sta alla base di questa convinzione risulta essere che nelle situazioni in cui vi siano una povera progettazione e documentazione, la perdita di un componente del team di sviluppo corrisponde alla perdita di conoscenza riguardante il lavoro da svolgere, in quanto gran parte del sapere risiede nei componenti del team. Questo può comportare gravi difficoltà a riprendere il progetto o a inserire nuovi membri. Disponendo invece di una documentazione completa ed esaustiva, risulta relativamente semplice l'aggiunta di nuovo personale o addirittura assegnare il progetto ad un nuovo team di sviluppo.

È sufficiente che chi verrà assegnato ad un nuovo incarico, consulti i vari documenti per familiarizzare con il progetto.

L’approccio tradizionale a cascata prevede almeno le seguenti fasi:

- raccolta dei requisiti;
- analisi dei requisiti;
- progettazione della soluzione;
- codifica;
- test e rilascio dell’applicazione.

Le fasi si concretizzano nelle seguenti attività, come viene mostrato in Fig.1:

- raccogliere tutti i requisiti e le funzionalità desiderate dalle singole direzioni;
- analizzare i requisiti per comprenderne la logica e la consistenza della soluzione;
- progettare una soluzione e sottoporla all’accettazione dell’utente sulla carta;
- realizzare la codifica dell’applicazione;
- effettuare i collaudi con l’aiuto nuovamente dell’utente per l’accettazione;
- implementare la soluzione e avviarle l’esercizio e la manutenzione.

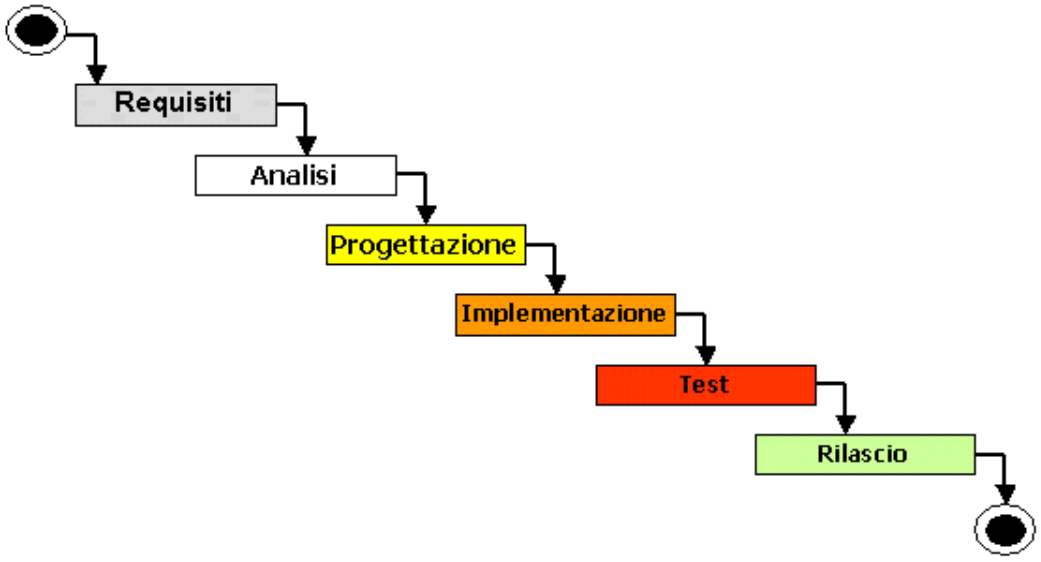

Fig.1 – Fasi dell’approccio Waterfall.

Con il passare degli anni, gli stessi sostenitori del modello a cascata si resero presto conto di alcune importanti problematiche che nascevano dall’utilizzo di tale modello. Nell’articolo in cui presentò tale modello, lo stesso Royce descrisse che pur essendo un approccio molto semplice, nella sua esperienza il modello a cascata non risultò adatto alla realizzazione di grandi sistemi software e che i costi di produzione utilizzando tale metodo eccedettero di gran lunga le stime inizialmente previste.

Alcune delle maggiori cause di fallimento del modello waterfall si possono riconoscere in quelle che seguono:

- il tempo necessario all'attività di analisi e pianificazione può ritardare l'implementazione;
- i requisiti, una volta formalizzati, possono essere modificati solo attraverso specifiche procedure di escalation (ingrandimento);
- il committente del progetto, anche a fronte di cambiamenti dello scenario del mercato, ha difficoltà ad influire su quanto richiesto, perché la fase di progettazione è tutta all'inizio e non è previsto un suo coinvolgimento;
- il cliente prende visione dei deliverable(consegnabili) solo al momento del loro completamento;
- possono emergere esigenze di nuove funzionalità in corso d'opera che necessitano di un approccio più flessibile ;
- quello che appare come un processo lineare ed efficiente diventa spesso una serie di cicli turbolenti che si traducono in una perdita di tempo e soldi. Questo perché le fasi sono legate tra loro da artefatti che non sono definibili in modo rigoroso e questo scatena tutta una fase di cicli di revisione all'indietro che riducono drasticamente l'efficienza del metodo.

In un mercato sempre più volatile come quello attuale, basato su rapidi cambiamenti, bisogna essere agili, maggiormente produttivi, veloci e innovativi. Il cambiamento come regola mette in discussione i principi teorici base del management "tradizionale", specialmente quelli legati alla stabilità dei requisiti o dei presupposti iniziali di progetto. Sempre più aziende prendono la decisione di passare da un approccio tradizionale di project management a un approccio iterativo che parte dal presupposto di ritornare più volte sulle varie fasi ridefinendo di volta in volta i vincoli di progetto finché non si raggiunge un sufficiente grado di maturità del prodotto.

Con un approccio allo sviluppo software più incline ad « abbracciare il cambiamento » piuttosto che a controllarlo, è possibile rispondere rapidamente alle opportunità di mercato e/o alle richieste dei committenti. In molti casi il time-to-market rimane il requisito fondamentale per il cliente, in quanto la perdita di una finestra di mercato, può rendere il software inutile.

Per questo motivo questi metodi si basano su processi iterativi che possano fornire piccoli rilasci di software in brevi cicli di sviluppo. Questo permette di ottenere un rapido feedback da parte del committente sin dalle prime settimane di lavoro.

3.Approccio Waterfall vs Metodologia Agile

	Metodologie a cascata (Waterfall)	**Agile (Scrum)**
Definizione dei requisiti	Requisiti dettagliati definiti prima dell'avvio di Disegno e Sviluppo	Definizione di alto livello del prodotto Requisiti di dettaglio definiti gradualmente in base all'avanzamento del disegno
Pianificazione	Forte enfasi sulla pianificazione preventiva	Pianificazione a finestra mobile Rinvio delle decisioni finché possibile
Controllo del contenuto	Il controllo del contenuto è essenziale per controllare costi e tempi (schedulazione)	Il contenuto può cambiare ed aumentare per soddisfare le esigenze del cliente
Approccio al Project Management	Enfasi sul controllo di costi e tempi	Enfasi sulla flessibilità e l"adattabilità per soddisfare le esigenze di business

Tab.1 – Tabella riassuntiva delle differenze tra l'approccio Agile e Waterfall.

Come si nota in Tab.1, l'approccio a cascata cerca di acquisire ed analizzare tutti i requisiti del progetto prima di avviare il disegno dell'applicazione. L'approccio Agile, invece, pianifica per fasi successive (sprint) rinviando le decisioni quanto più possibile.

In Fig.2 si nota che l'approccio tradizionale a cascata prevede tempi e costi in anticipo controllando il contenuto per rispettarli mentre l'approccio agile è molto più flessibile, consente di modificare i requisiti per soddisfare tutte le esigenze di business.

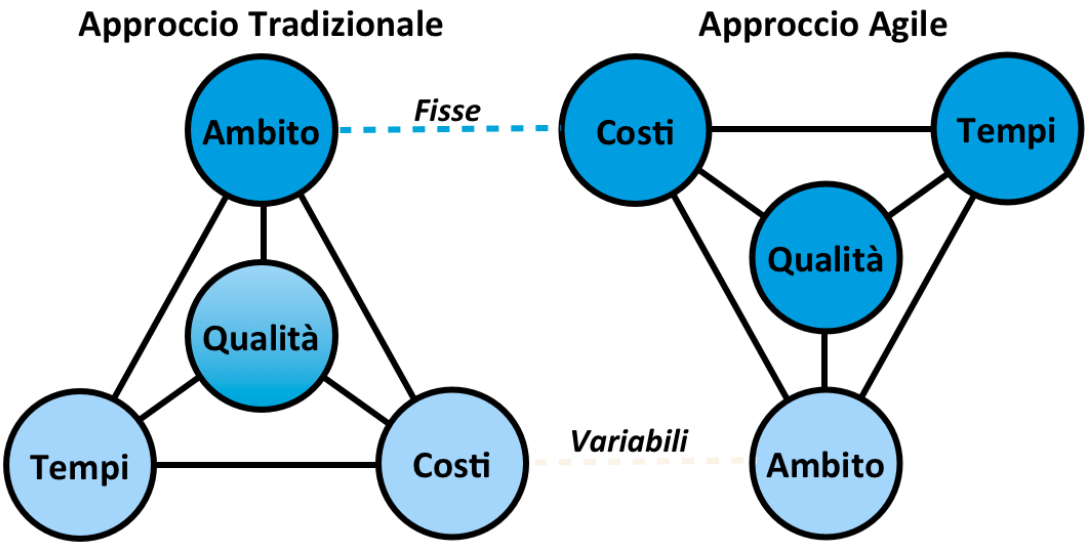

Fig.2 – Flessibilità dei requisiti nei due approcci.

CAPITOLO 2

Metodologia SCRUM

1.Cenni storici

Scrum è un framework di processo utilizzato dai primi anni novanta per gestire lo sviluppo di prodotti complessi. Non è un processo o una tecnica per costruire prodotti ma piuttosto è un framework all'interno del quale è possibile utilizzare vari processi e tecniche. Scrum rende chiara l'efficacia relativa del proprio product management e delle proprie pratiche di sviluppo così da poterle migliorare (Schwaber e Sutherland, 2011).

La metodologia Scrum affonda le radici delle sue origini in Giappone : il termine «Scrum» è stato introdotto da Takeuchi e Nonaka che nel 1986 descrissero un nuovo approccio allo sviluppo di prodotti commerciali che avrebbe aumentato la velocità e la flessibilità. I due autori giapponesi la definiscono come una "strategia flessibile e olistica allo sviluppo di un prodotto, dove il team di sviluppo lavora come un'unica entità per raggiungere un obiettivo comune, in opposizione all'approccio sequenziale delle metodologie tradizionali".

Il termine Scrum descrive la fase di mischia nel gioco del rugby, usata per riprendere il gioco dopo un'infrazione. L'analogia con la metodologia di Agile Software Development risiede nel fatto che, come spiega Takeuchi, il team di sviluppo deve lavorare come una squadra che "cerca di raggiungere l'obiettivo come unità, passando la palla avanti e indietro".

Nel 1995 Jeff Sutherland e Ken Schwaber (che sei anni più tardi furono due dei diciassette sottoscriventi del "Manifesto Agile di Sviluppo Software") presentarono per la prima volta un articolo che descriveva la metodologia Scrum. Negli anni successivi Sutherland e Schwaber continuarono lo sviluppo di Scrum integrandolo con le loro esperienze lavorative e alcune best-practice dell'industria. Il risultato è la metodologia Scrum che conosciamo al giorno d'oggi.

Nel febbraio del 2001 diciassette esperti dello sviluppo software (gli autori di Scrum, Extreme Programming, Dynamic Systems Development Method e Crystal) alla ricerca di alternative alle metodologie di sviluppo tradizionale si incontrarono in una località sciistica nelle Wasatch Mountains nello Utah. Tutti i partecipanti erano

concordi nel ritenere che le metodologie derivate dal modello a cascata non rappresentassero la soluzione più efficace in ambienti moderni, dove era importante possedere una certa “agilità” nell’affrontare la costruzione di sistemi software: nella maggior parte delle situazioni i clienti non avevano chiaro ciò che desideravano, e per questo andavano seguiti con particolare attenzione per cercare di far loro capire i requisiti intrinseci della soluzione desiderata e costruire con loro un forte legame di collaborazione. Uno dei punti principali, condiviso da tutti i partecipanti, risultava essere che tali requisiti non rimanevano immutati e subivano indubbiamente modifiche lungo il processo di realizzazione. Era necessario quindi prevedere questo cambiamento invece di controllarlo, e sfruttarlo per permettere al cliente di ottenere una soluzione più compatibile alle nuove esigenze emerse, siano esse di tipo tecnologico, manageriale, progettuale o temporale.

Essere agili significa inoltre alleggerire i processi: se è vero che una documentazione formale e approfondita del sistema può descriverlo in ogni aspetto e permette ad un nuovo membro la completa familiarizzazione, è altrettanto vero che proprio la grande mole di tale documentazione può essere un ostacolo in situazioni turbolente, dove anche pochi giorni di ritardo fanno la differenza. Un componente del team che necessità di essere istruito o aggiornato sul progetto in corso, può necessitare di troppo tempo per essere informato sull’intero progetto.

Il risultato fu la redazione del Manifesto Agile, che ha definito un set comune di 12 valori e principi generali per tutte le singole metodologie Agile dell'epoca :

1. “La nostra massima priorità è soddisfare il cliente per mezzo di tempestivi e continui rilasci di software di valore.”
2. “Siano benvenuti i cambiamenti nelle specifiche, anche a stadi avanzati dello sviluppo. I processi agili sfruttano il cambiamento a favore del vantaggio competitivo del cliente.”
3. “Rilascia software funzionante frequentemente, da un paio di settimane a un paio di mesi, con preferenza per i periodi brevi.”
4. “Manager e sviluppatori devono lavorare insieme quotidianamente durante tutto il progetto.”
5. “Basa i progetti su individui motivati. Dai loro l’ambiente e il supporto di cui necessitano e confida nella loro capacità di portare il lavoro a termine.”
6. “Il metodo più efficiente ed efficace di trasmettere informazione verso e all’interno di un team di sviluppo è la conversazione faccia a faccia.”
7. “Il software funzionante è la misura primaria di progresso.”

8. “I processi agili promuovono uno sviluppo sostenibile. Gli sponsor, gli sviluppatori e gli utenti dovrebbero essere in grado di mantenere un ritmo costante.”
9. “L’attenzione continua per l’eccellenza tecnica e la buona progettazione esaltano l’agilità.”
10. “La semplicità - l’arte di massimizzare l’ammontare di lavoro non svolto - è essenziale.”
11. “Le migliori architetture, specifiche e design emergono da team auto-organizzati.”
12. “A intervalli regolari il team riflette su come diventare più efficace, dopodiché mette a punto e aggiusta il suo comportamento di conseguenza.”

Fig.3 – I quattro principi base della metodologia Agile.

I principi su cui si basa una metodologia leggera che segua i punti indicati dall'Agile Manifesto, sono quattro (Fig.3):

- **Individui e reciproche interazioni:** le relazioni e la comunicazione tra gli attori di un progetto software sono la miglior risorsa del progetto;
- **Distribuzione di software funzionante**: bisogna rilasciare nuove versioni del software ad intervalli frequenti, e bisogna mantenere il codice semplice e avanzato tecnicamente, riducendo la documentazione al minimo indispensabile;
- **Collaborazione del cliente**: la collaborazione diretta offre risultati migliori dei rapporti contrattuali;

- **Risposta al cambiamento**: bisogna essere pronti a rispondere ai cambiamenti più che aderire al progetto, il team di sviluppo dovrebbe essere autorizzato a suggerire modifiche al progetto in ogni momento.

I firmatari del “Manifesto Agile” si auto-definirono in quella occasione la Agile Alliance il cui scopo è diffondere le idee per nuovi e più realistici approcci alla realizzazione di prodotti software.

2.I fondamenti teorici

Scrum si basa sulla teoria del controllo empirico dei processi, o empirismo. L’empirismo afferma che la conoscenza deriva dall’esperienza e che le decisioni si basano su ciò che si conosce. Scrum utilizza un metodo iterativo ed un approccio incrementale per ottimizzare la prevedibilità ed il controllo del rischio.

I pilastri che sostengono ogni implementazione del controllo empirico di processo sono:

- **Trasparenza** : Gli aspetti significativi del processo devono essere visibili ai responsabili del risultato finale (outcome). La trasparenza richiede che quegli aspetti siano definiti da uno standard comune in modo tale che gli osservatori condividano una comune comprensione di ciò che viene visto.
- **Ispezione** : Chi utilizza Scrum deve ispezionare frequentemente gli artefatti (modi di documentare il lavoro) di Scrum e l’avanzamento verso un obiettivo con lo scopo di rilevare le eventuali deviazioni indesiderate. Tali ispezioni non dovrebbero essere tanto frequenti da intralciare il lavoro stesso. Le ispezioni sono più utili quando eseguite diligentemente da chi ha l’abilità e la competenza necessaria a effettuarle rispetto ad un particolare stadio del lavoro.
- **Adattamento** : Se chi ispeziona verifica che uno o più aspetti del processo sono al di fuori dei limiti accettabili e che il prodotto finale non potrà essere accettato, deve adattare il processo o il materiale ad esso relativo. L’adattamento deve essere portato a termine il più rapidamente possibile per ridurre al minimo l’ulteriore deviazione.

Il framework Scrum è costituito dai Team Scrum e dai ruoli, eventi, artefatti e regole a essi associati. Ogni parte del framework serve a uno specifico scopo ed è essenziale per il successo e l’utilizzo di Scrum. Le regole di Scrum legano insieme gli eventi, i ruoli e gli artefatti governando le relazioni e le interazioni tra essi.

In sintesi Scrum definisce un set di attività che consentono al team di offrire più valore ai clienti in meno tempo. Queste attività offrono ai clienti la possibilità di esaminare, guidare e influire sul lavoro del team durante lo sviluppo. L' approccio non tenta di definire tutto all'inizio di un progetto. Al contrario, il team lavora in brevi iterazioni (noti anche come sprint) perfezionando il piano man mano che il lavoro procede (Fig.4).

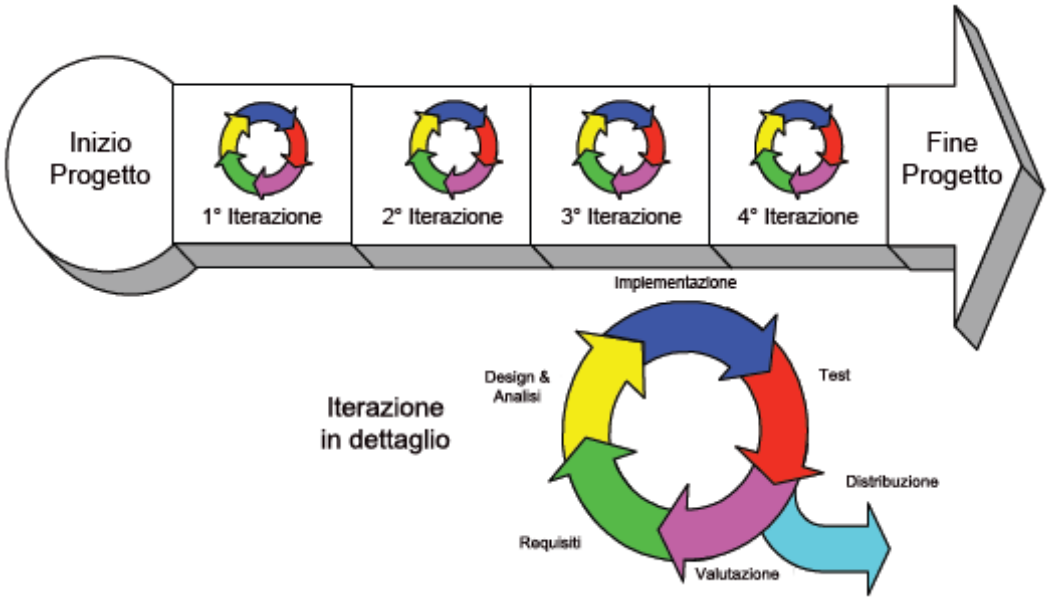

Fig.4 – Successione di iterazioni, sprint, del progetto.

3.Scrum Team

I ruoli principali nel processo Scrum costituiscono il Team Scrum e sono coloro che realizzano il prodotto. Il Team Scrum è formato dal Product Owner, Il Team di sviluppo (Development Team) e uno Scrum Master, come si nota in Fig.5.

I Team Scrum sono auto-organizzati e cross-funzionali: scelgono come meglio compiere il lavoro organizzandosi e coordinandosi al proprio interno e hanno tutte le competenze necessarie per realizzare il lavoro senza dover dipendere da nessuno al di fuori del team. Il modello di team in Scrum è progettato per ottimizzare la flessibilità, la creatività e la produttività. I Team Scrum rilasciano i prodotti in modo iterativo e incrementale, massimizzando le opportunità di feedback.

3.1. Il Product Owner

Il Product Owner rappresenta gli stakeholders ed è la voce del cliente. Ha la responsabilità di massimizzare il valore del prodotto e del lavoro svolto dal Team di

Sviluppo. Come questo è fatto può variare di molto secondo l'organizzazione, gli Scrum Team e gli individui.

Il Product Owner ha la responsabilità esclusiva di gestione del Product Backlog. Tale gestione prevede che:

● Gli elementi del Product Backlog siano espressi in modo chiaro;

● Gli elementi del Product Backlog siano ordinati per raggiungere meglio gli obiettivi e le missioni;

● Il valore del lavoro svolto dal Team sia ottimizzato;

● Il Product Backlog sia visibile, trasparente e chiaro a tutti e mostri su cosa lo Scrum Team lavorerà in seguito;

● Gli elementi del Product Backlog siano compresi al livello necessario dal Team di Sviluppo.

Il product Owner è un'unica persona, non un comitato. Può esprimere la volontà di un comitato nel Product Backlog, ma chiunque voglia cambiare l'ordine di un elemento deve rivolgersi al Product Owner.

3.2. Il Team di Sviluppo

Il Team di Sviluppo è costituito da professionisti che lavorano per produrre un incremento "Fatto" di prodotto potenzialmente rilasciabile alla fine di ogni Sprint. . Lo sprint è un'unità di base dello sviluppo in Scrum ed è di durata fissa, generalmente da una a quattro settimane. Ogni Sprint è preceduto da una riunione di pianificazione in cui vengono identificati gli obbiettivi e vengono stimati i tempi.

I Team di Sviluppo hanno le seguenti caratteristiche:

- Sono auto-organizzati. Nessuno (neanche lo Scrum Master) dice al Team di Sviluppo come trasformare il Product Backlog in Incrementi di prodotto potenzialmente rilasciabili;
- Sono cross-funzionali, con tutte le competenze come team necessarie a creare un incremento di prodotto;
- Scrum non riconosce alcun titolo ai membri del Team di Sviluppo al di fuori di Sviluppatore, indipendentemente dal lavoro eseguito dalla persona ;
- Non contengono sotto-team dedicati a particolari domini come il testing o la Business Analysis; non ci sono eccezioni a questa regola;

- I singoli membri hanno competenze specialistiche e aree di focus, ma è il Team di Sviluppo nel suo complesso ad avere la responsabilità finale.

La dimensione ottimale del Team di Sviluppo è abbastanza piccola da rimanere agile e abbastanza grande da completare un lavoro significativo all'interno dello Sprint, di solito è composta da 3 a 9 persone. Avere meno di tre persone nel Team di Sviluppo diminuisce l'interazione e comporta un minore guadagno in termini di produttività. Avere più di nove persone nel Team di Sviluppo richiede un eccessivo lavoro di coordinamento. I ruoli del Product Owner e dello Scrum Master non sono inclusi nel conteggio, a meno che non stiano eseguendo anche loro il lavoro contenuto nello Sprint Backlog.

3.3. Lo Scrum Master

Lo Scrum Master è responsabile della rimozione degli ostacoli che limitano la capacità del team di raggiungere l'obiettivo dello Sprint e i deliverable previsti. Gli Scrum Master fanno questo assicurandosi che lo Scrum Team aderisca ai valori, alle pratiche e alle regole di Scrum.

Sebbene sia un ruolo manageriale, lo Scrum Master non è il team leader, ma piuttosto colui che facilita una corretta esecuzione del processo. Lo Scrum Master aiuta coloro al di fuori dello Scrum Team a capire se le loro interazioni con lo Scrum Team sono utili oppure no.

Lo Scrum Master rende un servizio al Product Owner in vari modi, tra cui:

- Trovare le tecniche per una gestione efficace del Product Backlog;
- Aiutare lo Scrum Team a comprendere come creare gli elementi del Product Backlog in modo chiaro e conciso;
- Comprendere la pianificazione del prodotto in un ambiente empirico;
- Assicurare che il Product Owner capisca come ordinare gli elementi del Product Backlog per massimizzare il valore;
- Capire e praticare l'agilità;
- Facilitare gli eventi Scrum come richiesto e necessario.

Lo Scrum Master è di supporto anche al Team di Sviluppo:

- Supporta il Team di Sviluppo per l'autogestione e la cross-funzionalità;
- Aiuta a creare prodotti di alto valore;

- Elimina gli ostacoli ai progressi del Team di Sviluppo;
- Facilita gli eventi Scrum se necessario.

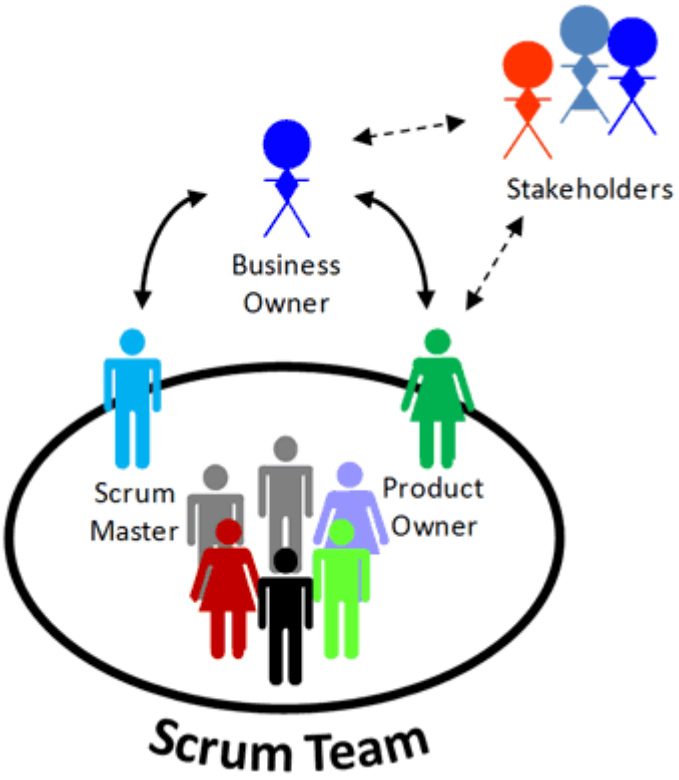

Fig.5 – Esempio di Scrum Team.

4.Gli Eventi di Scrum

Gli eventi previsti sono utilizzati in Scrum per creare regolarità e ridurre al minimo la necessità di riunioni non definite da Scrum stesso. Scrum utilizza eventi time-box, in modo che ogni evento abbia una durata massima. Questo assicura che una quantità appropriata di tempo sia trascorsa pianificando senza permettere l'introduzione di sprechi nel processo di pianificazione.

Oltre allo stesso Sprint, che è un contenitore di tutti gli altri eventi, ogni evento in Scrum è una occasione formale per ispezionare e adattare qualcosa. Questi eventi sono specificamente progettati per consentire trasparenza critica e ispezione. La mancata inclusione di uno qualsiasi dei risultati di questi eventi riduce la trasparenza ed è un'occasione persa per ispezionare e adattarsi.

Quando uno Sprint inizia, la sua durata è prefissata e non può essere né accorciata né allungata. Gli altri eventi possono invece terminare quando è raggiunto il loro scopo, assicurando che sia spesa l'appropriata quantità di tempo senza permettere sprechi all'interno del processo.

4.1. Lo Sprint

Lo Sprint è un'unità di base dello sviluppo in Scrum ed è di durata fissa, generalmente da una a quattro settimane. Ogni Sprint è preceduto da una riunione di pianificazione in cui vengono identificati gli obbiettivi e vengono stimati i tempi. Durante uno Sprint non è permesso cambiare gli obiettivi, quindi le modifiche sono sospese fino alla successiva riunione di pianificazione, in cui potranno essere inserite nel prossimo sprint. Ogni Sprint può essere considerato un progetto con un orizzonte non più lungo di un mese. Come i progetti, gli Sprint sono utilizzabili per realizzare qualcosa. Ogni Sprint ha una definizione di ciò che si va a costruire, un progetto e un piano flessibile che guideranno la costruzione, il lavoro svolto e il prodotto risultante.

Al termine di ogni Sprint il team di sviluppo consegna una versione potenzialmente completa e funzionante del prodotto, contenente gli avanzamenti decisi nella riunione di pianificazione dello sprint. Gli Sprint migliorano la prevedibilità, assicurando che l'ispezione e l'adattamento del progresso verso un obiettivo sono effettuati almeno una volta al mese. Gli Sprint inoltre limitano il rischio ad un mese di costo.

Lo Sprint consiste e contiene lo Sprint Planning, il Daily Scrum, il lavoro di sviluppo, la Sprint Review e la Sprint Retrospective.

Uno Sprint può essere cancellato prima della scadenza del tempo massimo stabilito se lo Sprint Goal diventa obsoleto: questo potrebbe verificarsi se l'organizzazione cambia direzione o se le condizioni di mercato o della tecnologia cambiano. Solo il Product Owner ha l'autorità di annullare lo Sprint, anche se può farlo anche sotto l'influenza degli stakeholder, del Team di Sviluppo o dello Scrum Master. Le cancellazioni degli Sprint consumano risorse, poiché tutti devono partecipare di nuovo ad un altro Sprint Planning Meeting per poterne cominciare un altro ; per questo motivo sono molto rare.

4.2. Sprint Planning

Il meeting di Sprint Planning è un incontro della durata massima di otto ore per uno Sprint di un mese. Durante questo meeting lo Scrum Master, il Product Owner e il

Team si riuniscono per determinare su quali funzionalità del prodotto il team dovrà lavorare (Fig.6).

L'input per questo incontro sono il Product Backlog, l'ultimo incremento del prodotto (somma di tutti gli elementi del Product Backlog completati durante uno Sprint), la previsione di capacità del Team di Sviluppo durante lo Sprint e le performance registrate in passato del Team di Sviluppo. Il numero di elementi selezionati dal Product Backlog per lo Sprint è definito esclusivamente dal Team di Sviluppo. Soltanto il Team di Sviluppo è in grado di valutare cosa può compiere durante il prossimo Sprint. Dopo che il Team di Sviluppo ha previsto gli elementi del Product Backlog che consegnerà con lo Sprint, lo Scrum Team modella lo Sprint Goal.

Lo Sprint Goal è l'obiettivo selezionato per lo Sprint che può essere raggiunto attraverso l'implementazione del Product Backlog. Esso fornisce una guida al Team di Sviluppo sul perché stia costruendo l'incremento. Lo Sprint Goal può creare quella coerenza nel lavoro del Team di Sviluppo che non sarebbe invece presente attraverso iniziative individuali senza un obiettivo comune.

Dopo aver creato lo Sprint Goal e selezionato gli elementi del Product Backlog per lo Sprint, il Team di Sviluppo decide come costruirà queste funzionalità in un Incremento "Fatto" del prodotto all'interno dello Sprint stesso. Le voci del Product Backlog selezionate per lo Sprint più il piano per la consegna definiscono lo Sprint Backlog. Il Team di Sviluppo di solito inizia con la progettazione del sistema e il lavoro necessario per convertire il Product Backlog in un Incremento del prodotto. Il Team di Sviluppo pianifica sempre avendo in mente lo Sprint Goal. Può capitare che durante lo Sprint il lavoro necessario sia diverso da quanto il Team di Sviluppo aveva pianificato. In questo caso il Team di Sviluppo collaborerà con il Product Owner per determinare come rivedere al meglio il piano senza perdere di vista lo Sprint Goal.

Alla fine del meeting di Sprint Planning, il Team di Sviluppo dovrebbe essere in grado di spiegare al Product Owner e allo Scrum Master come intende lavorare in quanto team autogestito, al fine di raggiungere l'Obiettivo di Sprint e creare l'Incremento previsto.

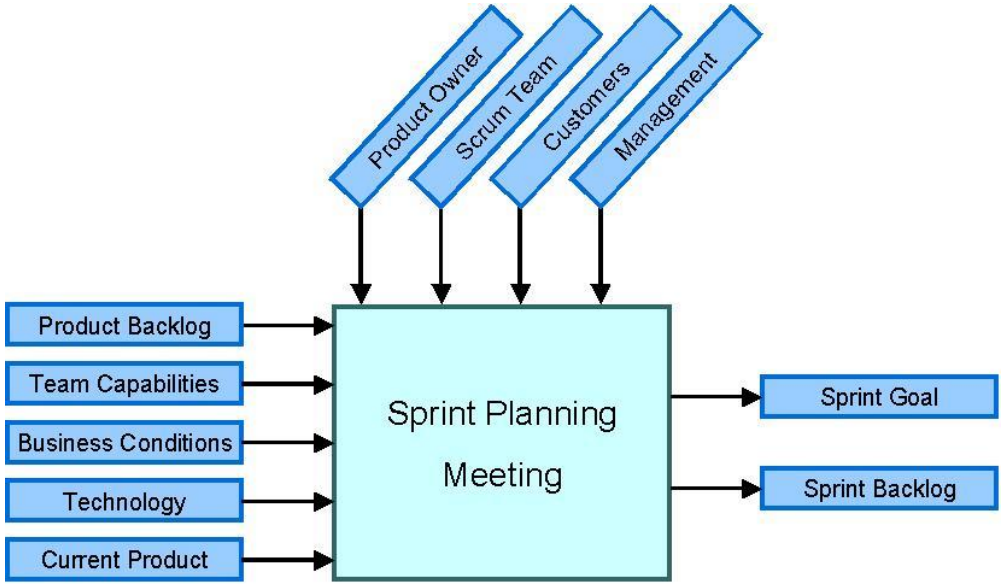

Fig.6 – Esempio di come funziona uno Sprint.

4.3.Daily Scrum

Ogni giorno durante lo Sprint, viene tenuta una riunione di comunicazione del team di progetto. Questo meeting viene chiamato "daily scrum" e ed ha un insieme di regole specifiche:

- Tutti i membri del Team di sviluppo vengono preparati con gli aggiornamenti per la riunione ;
- L'incontro inizia puntualmente anche se qualche membro del team è assente ;
- Il meeting dovrebbe avvenire ogni giorno nello stesso luogo e allo stesso tempo per ridurre la complessità ;
- La durata del meeting è fissata al tempo massimo di 15 minuti ;
- Si partecipa rimanendo in piedi, per non dare modo ai partecipanti di distrarsi ed isolarsi come accade nelle riunioni "tradizionali" ;
- Tutti sono benvenuti, ma normalmente solo i ruoli principali possono parlare .

Durante l'incontro ogni persona del Team di Sviluppo spiega:

- Cosa ho fatto ieri che ha aiutato il Team di Sviluppo a procedere verso lo Sprint Goal?
- Cosa farò oggi per aiutare il Team di Sviluppo a procedere verso lo Sprint Goal?
- Vedo degli ostacoli tra me o il Team di Sviluppo e lo Spring Goal?

Lo Scrum Master fa rispettare la regola che soltanto i membri del Development Team possono partecipare al Daily Scrum. Il Daily Scrum migliora la comunicazione, elimina altri incontri, identifica e rimuove gli ostacoli allo sviluppo, evidenzia e promuove il rapido processo decisionale e migliora il livello di conoscenza del progetto da parte del Team di Sviluppo. Rappresenta un incontro chiave d'ispezione e adattamento.

4.4.Sprint Review

Alla fine dello Sprint si tiene l'incontro di Sprint Review, della durata massima di 4 ore, per ispezionare l'incremento e adattare, se necessario, il Product Backlog. Durante la riunione di Sprint Review lo Scrum Team e gli stakeholder collaborano su ciò che è stato fatto durante lo Sprint.

A partire da questo e dai cambiamenti apportati al Product Backlog durante lo Sprint, i partecipanti collaborano sulle prossime cose che potrebbero essere fatte per ottimizzare il valore.

La Sprint Review include i seguenti elementi:

- I partecipanti includono lo Scrum Team e i principali stakeholder invitati dal Product Owner;
- Il Product Owner identifica ciò che è stato "Fatto" e ciò che non è stato "Fatto";
- Il Team di Sviluppo discute su cosa è andato bene durante lo Sprint, quali problemi si sono incontrati e come questi problemi sono stati risolti;
- Il Team di Sviluppo mostra il lavoro "Fatto" e risponde alle domande relative all'Incremento;
- Il Product Owner discute lo stato attuale del Product Backlog. Fa una previsione della possibile data di completamento in base alla misura dell'avanzamento attuale se necessario;
- L'intero gruppo collabora su cosa fare dopo, così la Sprint Review fornisce un prezioso contributo alle successive riunioni di Sprint Planning.
- Passare in rassegna come il marketplace o il potenziale utilizzo del prodotto possa avere cambiato l'elemento di maggior valore da implementare prossimamente;
- Passare in rassegna la timeline, il budget, le funzionalità potenziali e il marketplace per la prossima prevista release del prodotto.

4.5.Sprint Retrospective

La Sprint Retrospective è un incontro della durata massima di tre ore ed è un'occasione per lo Scrum Team per ispezionare se stesso e creare un piano di miglioramento da attuare durante il prossimo Sprint. Lo Scrum Team si riunisce per la Sprint Retrospective dopo la Sprint Review e prima del successivo Sprint Planning.

Lo scopo della Sprint Retrospective è di:

- Esaminare come l'ultimo Sprint è andato in merito a persone, relazioni, processi e strumenti;
- Identificare e ordinare gli elementi principali che sono andati bene e le migliorie potenziali;
- Creare un piano per attuare i miglioramenti al modo di lavorare dello Scrum Team.

Entro la fine della Sprint Retrospective, lo Scrum Team dovrebbe aver individuato i miglioramenti che saranno implementati nel prossimo Sprint. Attuare tali miglioramenti durante il prossimo Sprint è l'adattamento all'ispezione dello Scrum Team stesso. Anche se i miglioramenti possono essere implementati in ogni momento, la Sprint Retrospective fornisce un'opportunità formale per focalizzare l'ispezione e l'adattamento.

5.Gli Artefatti di Scrum

Gli artefatti di Scrum rappresentano il lavoro e il valore in diversi modi utili al fine di fornire trasparenza e opportunità di ispezione e adattamento. Gli artefatti definiti da Scrum sono progettati - nello specifico - per massimizzare la trasparenza delle informazioni chiave, di modo che ognuno abbia lo stesso livello di comprensione dell'artefatto.

Le decisioni per ottimizzare il valore e controllare il rischio sono prese in base allo stato attuale percepito degli artefatti. Nella misura in cui la trasparenza sia completa, tali decisioni hanno una base solida. Nella misura in cui gli artefatti non siano completamente trasparenti, tali decisioni possono essere imperfette, il valore può diminuire e il rischio può aumentare.

Compito dello Scrum Master è quello di lavorare con il Team di Scrum e con l'organizzazione per aumentare la trasparenza degli artefatti. Questo lavoro di solito

comporta una fase di apprendimento, una di persuasione ed una di cambiamento. La trasparenza non si ottiene da un giorno all'altro, ma piuttosto attraverso un percorso.

5.1.Product Backlog

Il product backlog è un elenco ordinato di idee per il prodotto, mantenuta nell'ordine ed è l'unica fonte di requisiti per le modifiche da apportare al prodotto. Il Product Owner è il responsabile del Product Backlog, compreso il suo contenuto, la sua disponibilità e l'ordinamento dei suoi elementi.
Un Product Backlog può nascere come una lunga lista oppure con pochi elementi. Non è mai completo, evolve come il prodotto al quale si riferisce, è dinamico e cambia continuamente per identificare ciò che serve al prodotto per essere appropriato, competitivo e utile.

Il Product Backlog elenca tutte le caratteristiche, le funzioni, i requisiti, le migliorie e le correzioni che costituiscono le modifiche da apportare al prodotto nelle future versioni. I suoi elementi hanno i seguenti attributi: descrizione, ordine, stima e valore. I requisiti non smettono mai di cambiare e perciò il Product Backlog è un artefatto vivente.

Il product backlog contiene delle stime approssimative sia del valore di business che dello sforzo necessario a svilupparle; questi ultimi valori sono spesso espressi mediante story point utilizzando una successione Fibonacci arrotondata. Le stime aiutano il Product Owner a calcolare la timeline e possono influenzare l'ordine dei backlog item.

Il raffinamento del Product Backlog è l'atto di aggiungere dettagli, stime e ordine agli elementi. Lo Scrum Team decide come e quando il raffinamento è completato. Il raffinamento solitamente occupa non più del 10% della capacità del Team di Sviluppo.

Questa attività comprende ma non è limitata a :

- Mantenere ordinato il Product Backlog ;
- Eliminare o riposizionare verso il basso gli elementi che non sono più ritenuti importanti ;
- Riposizionare gli elementi, nuovi o già esistenti, che assumono un'importanza maggiore ;
- Dividere gli elementi in parti più piccole ;
- Stimare gli elementi del Product Backlog.

Il Team di Sviluppo è responsabile di tutte le stime. Il Product Owner può influenzare il Team di Sviluppo aiutando a capire e selezionare i compromessi ma, coloro che eseguiranno il lavoro sono gli stessi che effettueranno la stima finale.

5.2.Sprint Backlog

Lo Sprint backlog è la lista del lavoro che il team di sviluppo deve effettuare nel corso dello sprint successivo. Questa lista viene generata selezionando una quantità di storie/funzionalità a partire dalla cima del product backlog determinata da quanto il Team di sviluppo ritiene possa realizzare durante lo sprint: ovvero avere una quantità di lavoro tale da riempire lo sprint.

Il Team di Sviluppo modifica lo Sprint Backlog durante tutto lo Sprint e lo Sprint Backlog emerge durante lo Sprint. Ciò si verifica quando il Team di Sviluppo opera attraverso il piano e viene a conoscenza di più dettagli sul lavoro necessario a raggiungere lo Sprint Goal.

Durante lo Sprint, il team di Sviluppo si auto-organizza per produrre un incremento di Prodotto corrispondente allo Sprint Backlog, come concordato durante lo Sprint Planning. Questo significa che il team si prende la responsabilità di produrre un incremento di prodotto in accordo con gli standard aziendali.

5.3.Incremento

Ogni Sprint genera un Incremento di Prodotto, che deve avere una qualità sufficientemente elevata da consentirne il rilascio agli utenti. L'Incremento è la somma di tutti gli elementi del Product Backlog completati durante uno Sprint e durante tutti gli Sprint precedenti.

Alla fine di uno Sprint, il nuovo Incremento deve risultare “Fatto”, il che significa che deve essere utilizzabile e deve incontrare la definizione di “Fatto” data dallo Scrum Team. Deve essere utilizzabile indipendentemente dal fatto che il Product Owner decida di rilasciarlo realmente o meno.

CAPITOLO 3

Gestire i progetti con Scrum

1.Preparazione del progetto

Prima di iniziare il progetto, è necessario effettuare le seguenti attività:

- stabilire il business case;
- assemblare un team;
- configurare l'infrastruttura del team.

In un progetto Scrum, il team impiegherà la maggior parte del tempo a sviluppare il prodotto in una serie di sprint. Tuttavia, il team deve innanzitutto creare un piano di alto livello per il progetto. Questo piano rappresenta una guida di orientamento per le decisioni più dettagliate che il team prenderà durante il corso del progetto, che cambierà man mano che viene implementato. Al termine della pianificazione del progetto, il team avrà creato un backlog del prodotto e, se necessario, un piano di rilascio.

Compilare il backlog del prodotto

E' un elenco di storie utente che descrivono le necessità e le stime degli utenti. Le storie utente vengono classificate in ordine di priorità in base al valore e al rischio aziendale e il team valuta le storie utente in unità astratte definite punti della storia.

Creare storie utente

Secondo la definizione "una storia utente descrive funzionalità utili sia per l'utente che per l'acquirente di un sistema o di un software" ("User Stories Applied" di Mike Cohn; 2004), le storie utente sono scritte dal punto di vista dell'utente finale.

Classificare le storie utente in ordine di priorità

Il proprietario del prodotto classifica le storie utente in ordine di priorità nel backlog del prodotto collaborando con i clienti per comprendere ciò che loro ritengono importante e con il team per comprendere rischi e dipendenze. Il proprietario del prodotto specifica le priorità assegnando una classificazione a ogni storia utente per indicare l'ordine in cui il team deve implementarle. Il proprietario del prodotto può utilizzare molte tecniche per analizzare e confrontare il valore delle storie utente.

Alcune tecniche di definizione delle priorità sono strettamente associate alla community Agile, per esempio il modello Kano della soddisfazione del cliente e la

ponderazione relativa di Karl Wiegers. Altre tecniche di definizione delle priorità, come la definizione delle priorità dei costi, il valore attuale netto e il tasso di rendimento interno, sono ben definite al di fuori della community Agile.

Stimare le storie utente

Il team stima in modo collaborativo ogni storia utente in punti della storia (Fig.7): "i punti della storia sono un'unità di misura per esprimere le dimensioni complessive di una storia utente, una funzionalità o un altro elemento di lavoro" ("Agile Estimation and Planning", Mike Cohn, 2005). I punti della storia sono valori relativi che non vengono tradotti direttamente in un numero specifico di ore. I punti della storia consentono invece a un team di quantificare le dimensioni generali della storia utente. Queste stime relative sono meno precise in modo da richiedere meno sforzo in fase di determinazione ma perdurano meglio nel tempo. Realizzando stime in punti della storia, il team fornirà inizialmente le dimensioni generali delle storie utente e svilupperà successivamente una stima più dettagliata delle ore di lavoro necessarie, quando i membri del team saranno pronti per iniziare l'implementazione delle storie utente.

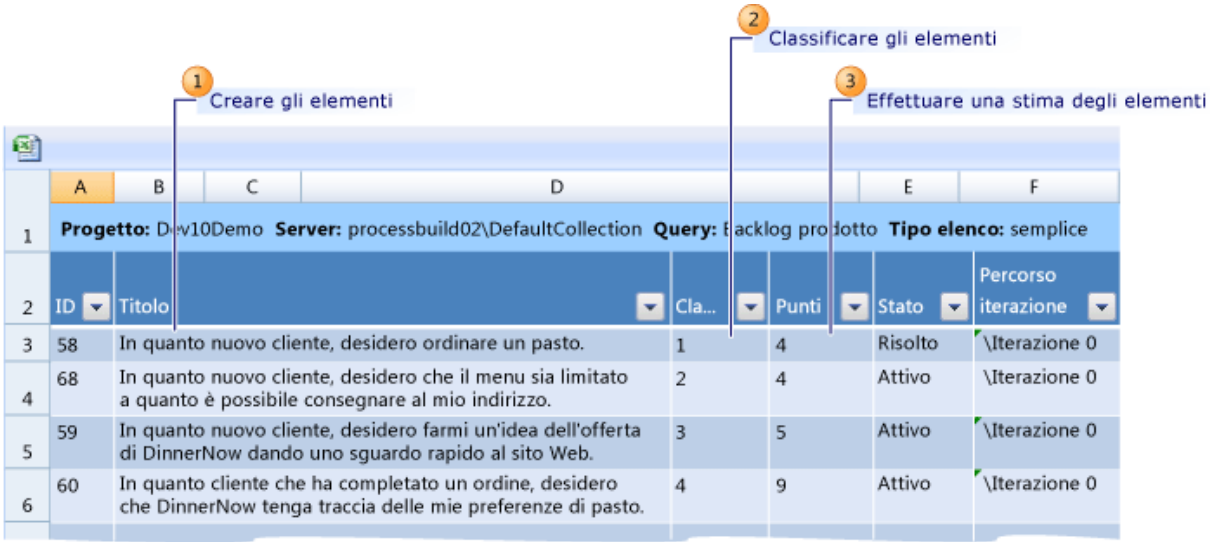

Fig.7 – Definizione storie utenti tramite software; Fonte: www.microsoft.com

Determinare la velocità del team

Prima di creare il piano di rilascio e pianificare ogni sprint, il team deve determinare la velocità. La velocità del team è rappresentata dal numero di punti della storia che possono essere completati in uno sprint.

Se il team ha calcolato la velocità raccogliendo i dati che indicano il numero di storie utente completate in un determinato periodo di tempo, è consigliato utilizzare tali

dati in quanto forniscono la visione migliore della velocità del team. Se non si dispone già di tali dati, dovranno essere raccolti nel corso del progetto.

Se i dati cronologici non sono disponibili, il team può effettuare una stima approssimativa del numero di punti della storia che possono essere completati nel primo sprint. E' necessario esaminare le storie utente valutate in cima allo stack di priorità ed eseguire una rapida valutazione di quante storie potrebbero essere completate in uno sprint. Dopo bisogna aggiungere i punti della storia per ognuna di tali storie utente per ottenere una stima iniziale.

Dopo il primo sprint, è possibile utilizzare dati cronologici per determinare la velocità del team. Nei primi sprint è possibile aspettarsi una variazione sostanziale in quanto il team sta imparando come effettuare le stime in modo coerente. Dopo diversi sprint la velocità calcolata del team dovrebbe diventare più stabile. Non appena la velocità calcolata del team diventa stabile, è necessario valutare nuovamente il piano di rilascio.

La stima della velocità del team fornisce un punto iniziale da utilizzare per determinare quante storie utente devono essere implementate nello sprint, ma la stima non costituisce la base dell'impegno del team. L'impegno del team dovrà basarsi su stime più dettagliate delle attività richieste per completare le storie utente.

Determinare il piano di rilascio

A ogni sprint il team completerà un incremento nel prodotto da rilasciare. Sebbene le storie utente che il team implementa siano pronte per essere distribuite alla fine dello sprint, è possibile che non apportino sufficiente valore aziendale da giustificare effettivamente il rilascio del prodotto.

Per pianificare le versioni assegnandole a iterazioni si devono rispettare i passi seguenti:

- Identificare gruppi di storie utente che, insieme, garantiscano valore aziendale sufficiente da offrire ai clienti;
- Determinare gli sprint in cui il team prevede di completare tali gruppi di storie utente.

Man mano che il team procede con il lavoro, verranno aggiunte storie utente al backlog del prodotto, mentre altre potranno essere rimosse. Inoltre, la priorità di alcune storie utente potrebbe cambiare ed è possibile che determinate storie utente

vengano implementate in anticipo o in ritardo rispetto a quanto originariamente previsto. Il proprietario del prodotto gestirà il piano di rilascio insieme al backlog del prodotto per tutta la durata del progetto.

Preparare il primo sprint

Prima che il team inizi il primo sprint, il proprietario del prodotto prepara il backlog del prodotto. Le storie utente con una priorità sufficientemente elevata da essere inserite nel primo sprint devono essere pronte affinché vengano implementate dal team.

Il proprietario del prodotto deve preparare il backlog del prodotto eseguendo le attività seguenti:

- Suddividere le storie utente in storie minori.
- Fornire informazioni dettagliate sulle storie utente che il team dovrà suddividere in attività.

Il proprietario del prodotto riterrà eccessive le dimensioni di una storia utente se essa rappresenta una quantità significativa della velocità totale del team. Ad esempio, una storia utente costituita da 15 punti della storia è considerata troppo grande da rientrare nella riunione di pianificazione dello sprint se la velocità del team è costituita da 30 punti della storia. Il team accetterà solo metà dello sprint per completare la storia utente.

Il team chiederà inoltre i dettagli sulle storie utente per essere in grado di suddividerle in attività e di stimare tali attività. Quando, ad esempio, il team esamina la storia utente "Come cliente desidero poter cercare un tipo di pasto", il team potrebbe porre i tipi di domande seguenti:

- "Deve trattarsi di una ricerca a testo libero o può essere una scelta da un elenco di tipi disponibili?"
- "Se più fornitori forniscono un pasto che corrisponde alla ricerca, come devono essere ordinati i risultati?"

2.Pianificare uno sprint

Una volta che il team ha sviluppato il backlog del prodotto e ha stabilito un piano di rilascio, può iniziare a lavorare in sprint. Il team inizia lo sprint con una riunione di pianificazione dello sprint in cui si impegna a completare un set di storie utente dal backlog del prodotto. Il set di storie utente e le attività di supporto costituiscono insieme il backlog dello sprint.

La riunione di pianificazione si svolge in due parti, ciascuna delle quali deve durare metà del tempo complessivo previsto per la riunione. Nella prima parte della riunione il proprietario del prodotto incontra il team per discutere delle storie utente che potrebbero essere incluse nello sprint. Il proprietario del prodotto condividerà le informazioni e risponderà alle domande poste dal team in merito alle storie. Durante questa conversazione potrebbero emergere dettagli quali le origini dati, il layout dell'interfaccia utente, le aspettative in termini di tempi di risposta e considerazioni per la sicurezza e l'usabilità. Il team aggiungerà questi dettagli alle storie utente. Durante questa parte della riunione, il team otterrà informazioni sulla natura della compilazione.

Nella seconda parte della riunione, il team determina come svilupperà e testerà tali storie utente. Il team suddivide quindi le storie utente in attività e stima il lavoro necessario per completarle. La tecnica denominata planning poker consente di stimare le ore dell'attività. Tramite questo strumento, ogni membro del team può partecipare alla stima, anziché dipendere dagli esperti in materia per la stima delle proprie attività. Sia che si utilizzi questa tecnica o un'altra, sarà necessario coinvolgere tutto il team nella determinazione del numero di ore richieste per ogni attività.

Per completare le attività non deve essere necessario più di un giorno. Se il team determina che non è in grado di completare una o più delle storie utente richieste dal proprietario del prodotto perché le ore stimate per le attività superano il numero di ore disponibili, sarà necessario avvisare il proprietario del prodotto. Il proprietario del prodotto potrebbe sostituire una storia con una più piccola, suddividere la storia o mantenere la storia nel backlog del prodotto per uno sprint futuro.

Al termine di entrambe le parti della riunione di pianificazione, il team avrà:

- creato il backlog sprint, con le attività e le ore per ogni storia utente
- confermato che le storie utente verranno recapitate nello sprint
- inteso, come team auto-organizzato, il modo in cui lavorerà unitamente per soddisfare gli impegni

Dopo l'inizio dello sprint, le storie utente presenti nel backlog dello sprint non vengono modificate. Pertanto, il piano deve essere sufficientemente dettagliato affinché il team possa tranquillamente adempiere all'impegno.

2.1.Scegliere le storie utente

Il team sceglie le storie utente che devono essere implementate nello sprint e identifica le storie utente con la massima priorità e i cui punti della storia non superano la velocità stimata. È possibile utilizzare la cartella di lavoro Backlog iterazione per pianificare e tenere traccia dello stato di avanzamento del lavoro per ciascuna iterazione, anche nota come sprint. Questa cartella di lavoro consente di calcolare la capacità del team in base al lavoro richiesto stimato e rimanente definito per le attività. In questa cartella di lavoro sono presenti cinque fogli di lavoro, come mostrato nell'illustrazione seguente.

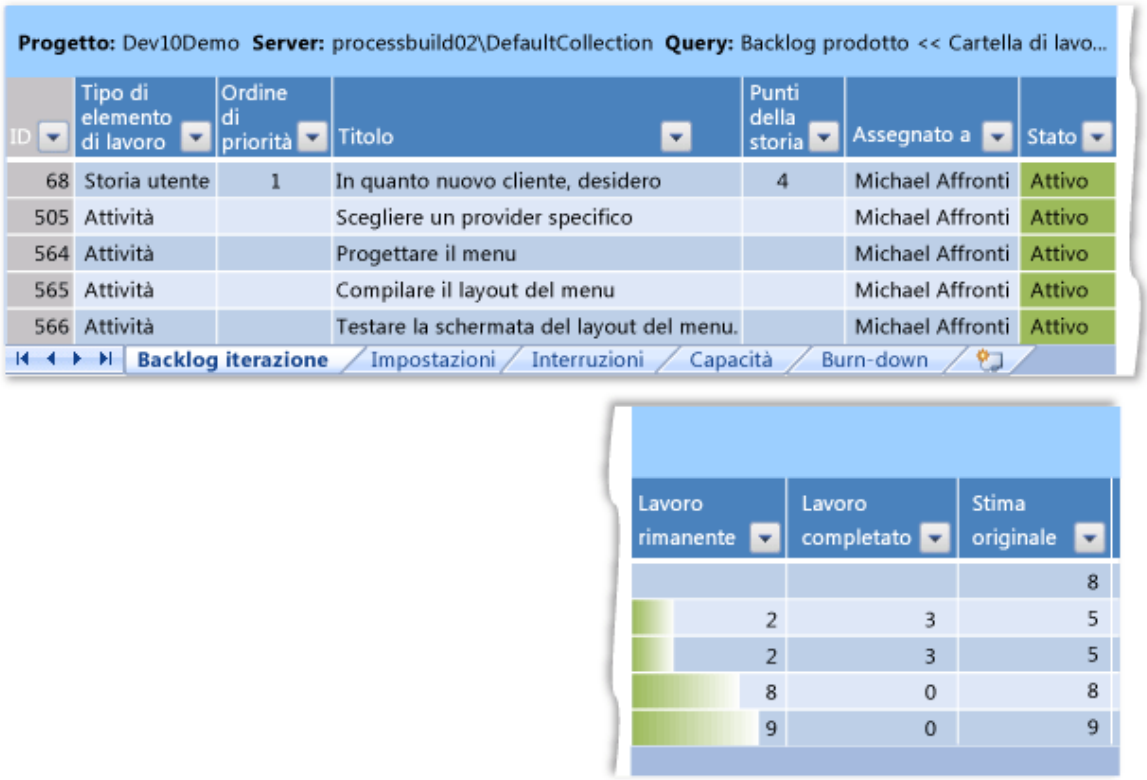

Fig.8 – Cartella di lavoro Backlog iterazione; Fonte: www.microsoft.com

I fogli di lavoro,fig.8, vengono utilizzati nei modi seguenti:

- Backlog iterazione: consente di verificare che tutte le attività siano assegnate a storie utente; rivedere e assegnare i livelli di lavoro richiesto a ogni attività; assegnare attività alle iterazioni.
- Impostazioni: consente di programmare l'iterazione.
- Interruzioni: consente di specificare le festività e altri giorni di astensione dal lavoro per il team e per i singoli membri.
- Capacità: consente di bilanciare il carico di lavoro tra il team.
- Burn-down: consente di stimare la fine dell'iterazione in base alle date di inizio per l'iterazione.

Il team utilizza la cartella di lavoro Backlog iterazione per assicurarsi di avere a disposizione un numero di ore di lavoro sufficiente per completare tutte le attività (Fig.9). Se lo sprint implica più lavoro di quanto possa essere gestito dal team nello sprint, le storie utente con la classificazione più bassa devono essere rimosse affinché il lavoro rientri nella capacità del team per lo sprint. È possibile sostituire le storie più grandi che non si adattano allo sprint con storie più piccole con una priorità più bassa.

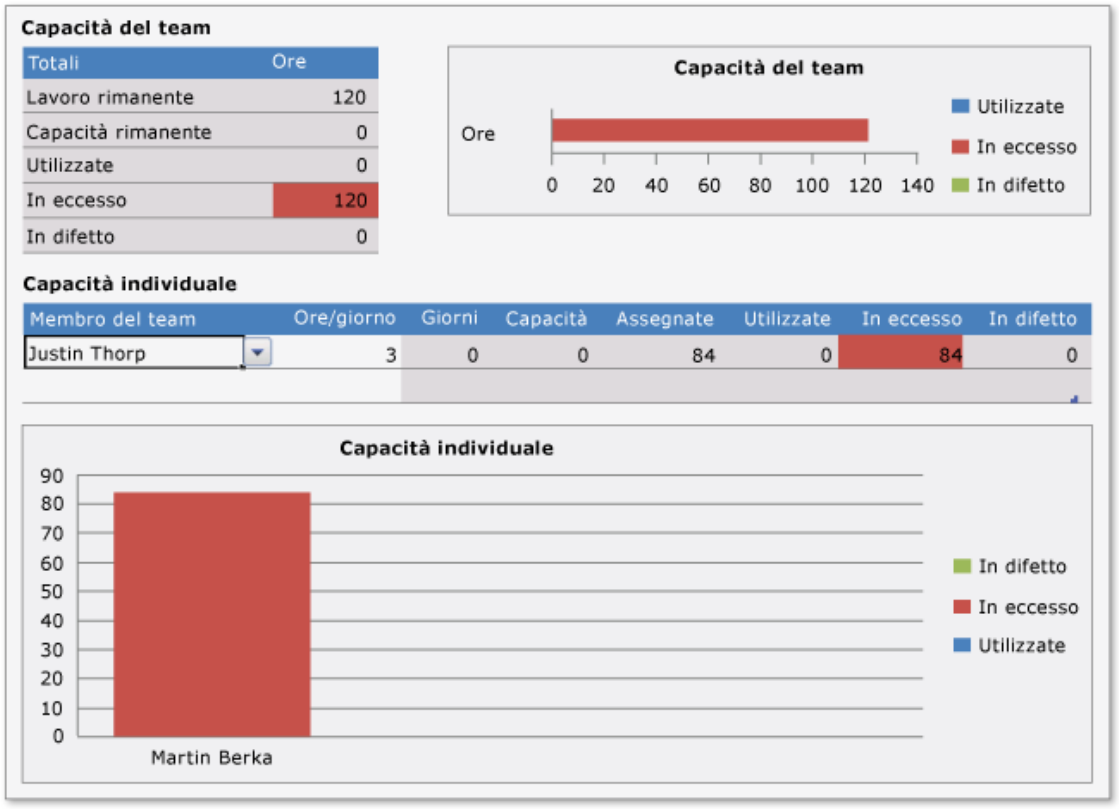

Fig.9 – Misurazione capacità del team tramite la cartella di lavoro Backlog iterazione; Fonte: www.microsoft.com

3.Eseguire uno sprint

Dopo aver pianificato lo sprint, il team avrà un elenco di storie utente che deve completare durante lo sprint. Queste storie utente sono state suddivise in attività. Ogni membro del team si iscrive a un'attività quando inizia lo sprint. Dopo avere completato tale attività, il membro del team aggiorna lo stato e si iscrive a un'altra attività. In questo modo, il team lavora rispettando l'elenco delle attività, completando le storie utente nel backlog dello sprint. Un membro del team può indicare le attività che vengono completate durante l'archiviazione del codice.

Scrum si basa sulla comunicazione tra le persone più che sui processi formali per garantire che vengano comprese le dipendenze, venga condivisa l'esperienza e le modifiche ai piani vengano apportate in modo efficace. Il team deve tenere riunioni Scrum giornaliere. Ogni membro del team descrive ciò che ha realizzato dall'ultima riunione, il lavoro che prevede di completare nel giorno attuale e qualsiasi problema o difficoltà che potrebbe influire sugli altri membri del team o richiederne il supporto. Lo Scrum Master applica rigorosamente la struttura della riunione e si assicura che inizi puntualmente e si concluda nell'arco di 15 minuti circa. In questa riunione ogni membro del team risponde a tre domande:

- Che cosa ho portato a termine rispetto all'ultima riunione Scrum?
- Cosa porterò a termine prima della successiva riunione Scrum?
- Quali problemi o difficoltà potrebbero influire sul lavoro?

La riunione deve iniziare e finire con puntualità ed essere organizzata alla stessa ora e nello stesso luogo tutti i giorni. Questa coerenza è di aiuto al team perché viene creato un modello a cui conformarsi.
Il software prodotto da un team Scrum non dovrebbe contenere errori. Tuttavia, è probabile che il team rileverà alcuni bug nei progetti. Gestire i bug tenendo presente il fatto che è più conveniente e più rapido correggerli appena trovati anziché rimandarli a un momento successivo. Quando il team rileva dei bug, deve correggerli immediatamente.

Una volta completata una o più iterazioni o sprint, è possibile analizzare il rapporto burn-down e velocità per determinare la velocità con cui il team ha proceduto con il lavoro. Come si nota in Fig.10, il burn-down mostra la tendenza del lavoro completato e rimanente in un periodo di tempo specificato. La velocità fornisce dei calcoli della frequenza di lavoro completata e richiesta in base al periodo specificato. Inoltre, nel rapporto è presente un grafico che mostra la quantità di lavoro completato e rimanente assegnato ai membri del team. È possibile visualizzare il rapporto burn-down e velocità basato sulle ore lavorate o sul numero di elementi di lavoro che sono stati risolti e chiusi.

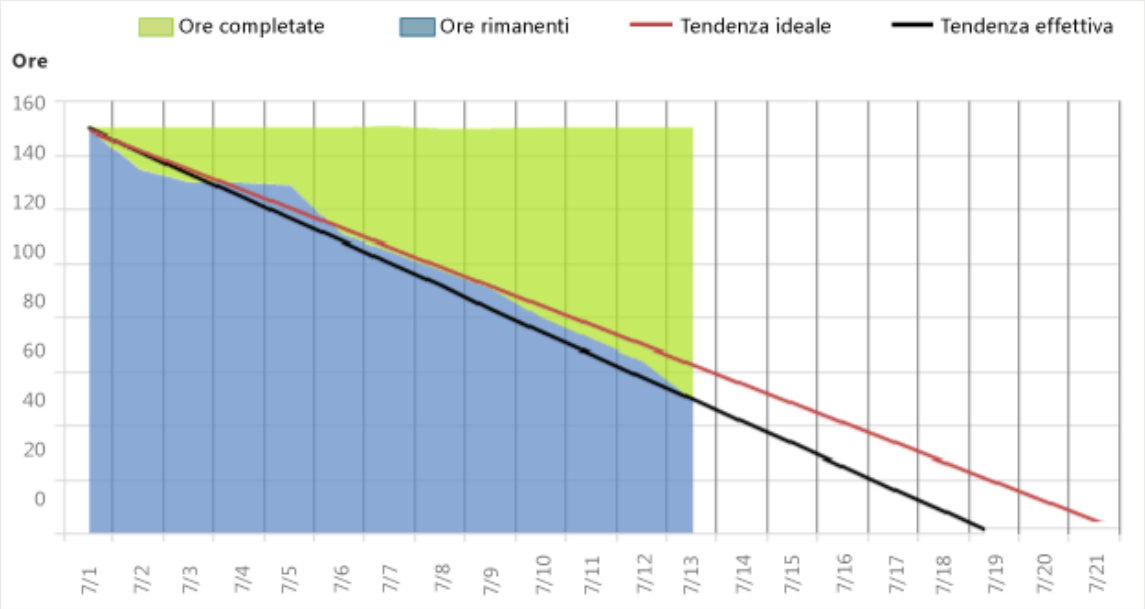

Fig.10 – Rapporto Burn-down; Fonte: www.microsoft.com

Durante l'avanzamento dello sprint il team potrebbe identificare la necessità di dover svolgere un lavoro non previsto che però si è rivelato necessario al fine del completamento di una storia utente. In questo caso, è necessario creare un'attività, stimarla e determinare quindi se il team può completarla nelle ore rimanenti dello sprint. Se il team è in grado di completare l'attività, si prosegue con lo sprint. Se invece non può completare l'attività nello sprint, il team incontra il proprietario del prodotto per stabilire come procedere.

Il proprietario del prodotto e il team possono risolvere il problema apportando i tipi di modifiche seguenti:

- Ridurre i criteri di accettazione per la storia utente, in modo da rendere l'attività non necessaria;
- Rimuovere la storia utente dal backlog dello sprint;
- Annullare lo sprint.

Nell'ultimo giorno dello sprint, il team incontra il proprietario del prodotto, i clienti e le parti interessate per accettare il lavoro completato e identificare nuovi requisiti. Durante lo sprint, il team deve avere già raccolto e incorporato i commenti e i suggerimenti. Il team deve inoltre avere già eseguito i test di accettazione per ogni storia utente completata. In questa riunione il team dimostrerà tutte le storie utente che ha completato nello sprint. Il proprietario del prodotto, i clienti e le parti interessate accettano le storie utente che soddisfano le aspettative. In base a questa riunione, alcune storie utente verranno accettate come complete. Nel backlog del prodotto

verranno mantenute le storie utente incomplete e verranno aggiunte nuove storie utente.

Entrambi i set di storie utente verranno classificati e stimati o stimati di nuovo nella successiva riunione di pianificazione dello sprint. Dopo questa riunione e la riunione retrospettiva, il team pianificherà lo sprint successivo. Poiché le esigenze aziendali cambiano rapidamente, è possibile sfruttare questa riunione con il proprietario del prodotto, i clienti e le parti interessate per rivedere nuovamente le priorità del backlog del prodotto.

4.Tenere traccia del progetto

Come il team lavora in sprint per produrre incrementi nel progetto, così i clienti sviluppano una migliore comprensione delle necessità rimanenti, il che può implicare eventuali modifiche all'ambiente aziendale. Il proprietario del prodotto lavora insieme ai clienti per capire tali modifiche, gestisce il backlog del prodotto e il piano di rilascio per riflettere le modifiche e assicurarsi che il team disponga di tutto il necessario all'inizio di ogni sprint. Il team tiene traccia dello stato di avanzamento del prodotto nel suo complesso per assicurarsi che i progressi verso il completamento del progetto procedano correttamente.

4.1.Preparare lo sprint successivo

La validità del backlog del prodotto ha una relazione diretta con la qualità complessiva e la completezza del progetto. Il backlog deve essere regolarmente aggiornato, modificato e riformulato per assicurarsi che sia pronto ogni volta che il team sta per iniziare uno sprint.

Il proprietario del prodotto prepara il backlog del prodotto per lo sprint successivo eseguendo le attività seguenti, come si vede anche in Fig.11:

- Aggiornamento delle storie utente e delle relative priorità in base alle nuove esigenze dei clienti.
- Suddivisione delle storie utente la cui implementazione è prevista nello sprint successivo.

Quando il team finisce uno sprint, le altre storie utente si avvicinano all'inizio del backlog del prodotto. Il proprietario del prodotto deve analizzare e suddividere le storie che si trovano all'inizio, in modo tale che il team possa implementarle nello sprint successivo.

Cohn definisce spesso questo processo come iceberg del backlog del prodotto. Man mano che il team lavora su un set di funzionalità, l'iceberg si scioglie lasciando affiorare nuovo lavoro e riducendo le proprie dimensioni. In questo processo, emergono ulteriori dettagli aggiuntivi, senza vincoli di spazio e tempo.
Mentre il team è impegnato nell'esecuzione di uno sprint, il proprietario del prodotto non può aspettarsi di avere dal team lo stesso livello di coinvolgimento nella gestione del backlog del prodotto avuto durante la preparazione del primo sprint. Per aiutare il proprietario del prodotto a preparare il backlog con un'interruzione minima dello sprint, il team e il proprietario del prodotto discuteranno i problemi aperti relativi al backlog del prodotto nel corso dello sprint.

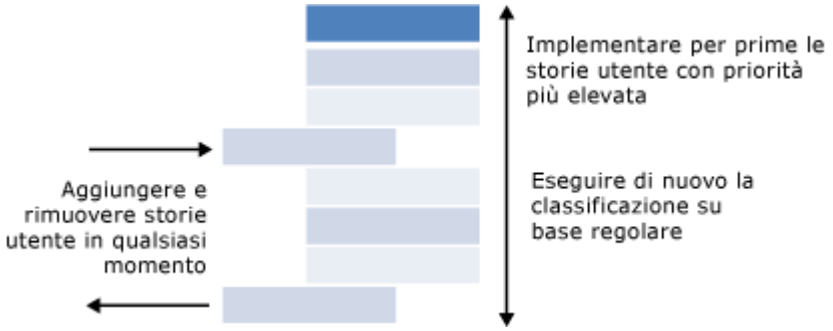

Fig.11 – Aggiornamento e suddivisione delle storie utente; Fonte: www.microsoft.com

4.2.Tenere traccia dello stato di avanzamento del rilascio

Con l'avanzamento del progetto sprint dopo sprint, il team tiene traccia dello stato di avanzamento complessivo verso il rilascio successivo. Il team terrà traccia dello stato di avanzamento anche per stimare e migliorare la velocità.

Man mano che il team tiene traccia dello stato di avanzamento, deve tentare di rispondere a domande del seguente tipo:

- Stiamo lavorando sulle storie utente più appropriate? Il backlog del prodotto viene ridefinito con nuove storie utente con l'avanzamento del progetto. Tuttavia, se il numero totale di storie nel backlog non diminuisce, sebbene vengano completate storie a ogni sprint, il team deve esaminare la causa. Le storie completate potrebbero non rappresentare le scelte migliori. Il team deve prefissarsi una visione e un obiettivo per ogni rilascio e deve assicurarsi che le storie siano collegate direttamente a ciò che viene richiesto dal cliente.

- Siamo in presenza di un debito tecnico? Alcuni team considerano una storia utente finita anche se rimane ancora del lavoro da completare come, ad esempio, correggere alcuni bug. Tali team si assumono un debito tecnico che devono pagare in un secondo momento, generalmente a un costo maggiore.

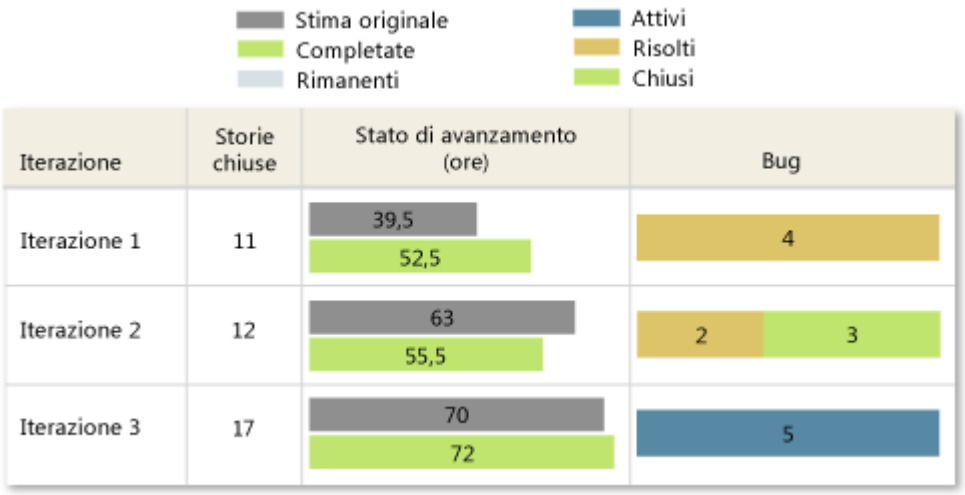

Fig.12 – Stato di avanzamento delle iterazioni; Fonte: www.microsoft.com

Per ogni iterazione definita per le aree del prodotto specificate, questo rapporto (Fig.12) visualizza le informazioni seguenti:

- **Storie chiuse**: il numero di storie utente chiuse. Questi valori sono derivati dai valori correnti specificati per l'iterazione e lo stato di ogni storia utente.
- **Stato di avanzamento (Ore)**: una rappresentazione numerica e visiva a due barre che rappresenta i valori per **Stima Originale** (grigio)**, Completato** (verde), **Rimanente** (celeste), basati sul rollup delle ore definite per tutte le attività. Questi valori sono derivati dai valori correnti specificati per l'iterazione e le ore di ogni attività.
- **Bug**: un valore numerico e una rappresentazione visiva per tutti i bug, raggruppati in base agli stati correnti di **Attivo** (blu), **Risolto** (oro) e **Chiuso** (verde).Questi valori sono derivati dai valori correnti specificati per l'iterazione e lo stato di ogni bug.

Inoltre, è possibile fare clic su un'iterazione per accedere al rapporto burn-down e velocità per quell'iterazione.

4.3.Finire il rilascio

Se il team non accumula alcun debito tecnico, può rilasciare il prodotto non appena ha completato gli sprint del rilascio, senza alcun lavoro aggiuntivo. Il team e il proprietario del prodotto tengono riunioni retrospettive e per la revisione del cliente al fine di esaminare il rilascio nel suo complesso.

Tuttavia, il debito tecnico è un problema difficile che i team non possono eliminare facilmente. Se il proprio team, come molti altri, continua ad accumulare debito tecnico, sarà necessario dedicare tempo allo svolgimento di tutto il lavoro rimanente per finire le storie utente prima di rilasciare il prodotto. Nella retrospettiva per il rilascio, considerare le azioni che il team deve eseguire negli sprint successivi al fine di evitare di assumere altro debito.

CAPITOLO 4

Casi studio

Scrum fornisce una base per l'esecuzione dei progetti. Tuttavia, il processo Scrum deve essere adattato a ogni progetto per rispondere a specifiche esigenze. Il modo in cui viene implementato è il fattore chiave che porta al successo o al fallimento del progetto. Di seguito vengono illustrati alcuni Casi Studio che spiegano i diversi modi di applicare la tecnica SCRUM.

1.Intel Corporation

Il seguente caso studio è stato redatto da Pat Elwer, in collaborazione con Tim Gallagher, Intel Corporation; Dan Rawsthorne, Danube Technologies, Inc.; Michael James, Danube Technologies, Inc.

1.1.Sommario

Nel settore dei microprocessori, il gruppo di ingegnerizzazione di sviluppo del prodotto (PDE) è formato da più gruppi e garantisce il funzionamento del prodotto. Trovandosi in mezzo tra i team di progettazione e quelli di fabbricazione, PDE è spesso messo sotto pressione senza avere controllo su scadenze, obiettivi, requisiti e output del team.

Per meglio coordinare il lavoro dei gruppi all'interno del PDE, sette squadre composte da circa 50 persone si offrirono di usare la tecnica Scrum. Di seguito viene descritto il percorso fatto dall'organizzazione, le lezioni apprese ed i risultati dell'investimento in Scrum.

1.2.Introduzione

Tante best practice esistono circa l'implementazione della tecnica Scrum nel campo dell'ingegneria informatica. Alcune coinvolgono piccole-medie imprese che sviluppano software riguardanti i linguaggi orientati agli oggetti. Il PDE team, chiamato Oregon e Pacific (OAP), implementa lo Scrum tra diverse squadre, culture e ambienti.

L'esperienza nella fabbricazione e produzione ha portato in Intel una forte cultura «a cascata», ritenuto il miglior modo per raggiungere il successo. Ciascun team era composto da esperti di settore le cui competenze raramente si sovrapponevano con quelle degli altri membri. Il risultato fu che alcuni team avevano troppa responsabilità nelle fasi finali del progetto e altri team nessuna.

Per unire i team e migliorare la consegna dei prodotti, si scelse di introdurre Scrum all'inizio del progetto, quando la maggior parte del lavoro si basava sullo sviluppo della struttura. Se Scrum avesse funzionato in questa prima fase, allora avrebbe funzionato anche nelle fasi più complesse.

1.3.Fase 1: Preparazione

Si scelse di partire con un gruppo di prova, che comprendeva sei team e numerosi sotto-team. Come primo passo, Danube Technologies Inc.(azienda specializzata nell'applicazione dei processi Scrum) realizzò per i team leader dei corsi introduttivi ai principi e pratiche Scrum, al quale parteciparono circa 20 manager.

Dopo la formazione, i manager parteciparono a una riunione per discutere le loro impressioni sull' approccio Scrum, senza la presenza dei rappresentanti del Danube. I team leader accettarono di impegnarsi per 3 mesi nell'applicazione dei principi e delle pratiche prima di mettere in discussione l'efficacia del nuovo processo o di provare ad adattarlo alle esigenze di Intel. Un team di processo di azione (PAT) è stato formato dagli esperti del Danube per monitorare lo sviluppo di Scrum all'interno dei team e sostenerli durante il processo.

Insieme al team PAT si decise che avere dei Scrum Master sarebbe stato importante per il successo. In primo luogo, si lavorò insieme al management team di Intel per cambiare il ruolo di Scrum Master da semplice carico amministrativo a un "lavoro di ingegneria". In secondo luogo, chi si offrì per il ruolo dei Scrum Master lo diventò nel team che aveva bisogno di una persona con una certa preparazione tecnica. Non era prevista una ricompensa per questo ruolo, tuttavia chi si era offerto fu facilitato con un cambio delle proprie responsabilità all'interno del team.

Alla fine dei 3 mesi di prova ci furono altri tre Scrum Master a condurre 7 team. Inoltre un ottavo team si offrì di iniziare a usare Scrum.

Dopo approssimativamente 5 mesi la dimensione di ogni team diventò la sfida più grande, c'era bisogno di più conoscenze su come gestire la collaborazione tra vari team e su come facilitare la comunicazione tra di loro. Danube decise di tenere un altro corso per affrontare il problema. Il corso durò un'intera giornata e trattò in particolare la pianificazione dello sprint e la dimensione dei team.

Due degli aspetti principali che si trattarono con i consulenti di Danube furono volontariato e auto-organizzazione. Nonostante alle squadre che si erano impegnate per un periodo di prova fosse stato chiesto di aderire ai principi e alle pratiche di base, l'adozione era chiaramente più importante della sola aderenza. Dopo la prova di tre mesi, alle squadre fu data la libertà di organizzarsi e di adattare il loro approccio ad ogni sprint. Le differenze furono discusse, ma non giudicate, nei meeting settimanali. L'obiettivo principale in questa fase era l'unità, non l'uniformità.

Anche la visibilità era parte fondamentale del processo. Un wiki interno consentì ai team di documentare cosa stava funzionando per loro, cosa non stava funzionando, e realizzare best practice per l'adozione di Scrum. Dopo i primi tre mesi alcune modifiche furono apportate per adattare Scrum alla cultura e all'ambiente lavorativo di Intel.

In primo luogo, il team definì i ruoli più utili ai loro obiettivi. Furono sviluppati i seguenti:

- **Business Owners**: Dirigenti o ingegneri incaricati della sorveglianza di più squadre. BOs decidevano i piani di rilascio e definivano le caratteristiche di ogni obiettivo.
- **Product Owners:** Manager dei vari team.
- **Technical Owners**: Leader tecnici di ogni area funzionale che collaboravano sui problemi di integrazione e dipendenza per assicurare allineamento tra le squadre.
- **Scrum Master**: Ingegnere a capo del team, senza un altro ruolo specifico.
- **Transient**: Membro del team con competenze altamente specializzate, necessarie a più squadre per uno o due sprint alla volta.
- **Conduit**: Membro del team rappresentante i membri di un team remoto.
- **Story Owner:** Esperto tecnico con particolare conoscenza su come completare una storia. Poteva sviluppare compiti e chiedere la partecipazione di alcuni membri del team a completare tali compiti.

Entro la fine del primo anno Scrum mise radici all'interno dell'organizzazione e diventò la base per la pianificazione del lavoro e controllo dei requisiti.

1.4.Sopravvivenza dei team

La prima volta che si usa la tecnologia con il silicio, l'ingegnere che si occupa dello sviluppo del prodotto può incontrare varie difficoltà. Quando arriva il silicio, tutti i

requisiti sono ambigui e ci vogliono settimane per raccogliere i dati necessari a determinare il percorso che il progetto prenderà.
La prima prova sul prodotto non ebbe i risultati aspettati. Quindi si decise di fare un passo indietro e di riadattare l'approccio dell'organizzazione a Scrum.
Gli sprint inizialmente fissati di due settimane erano impossibili da mantenere, quindi molti team optarono per sprint da un giorno. I membri si riunivano in meeting di un'ora ogni giorno per pianificare le successive 24h e per riflettere sui risultati del giorno precedente. I 4 meeting previsti dallo Scrum erano crollati con la prima introduzione del silicio, però nonostante questo durante le riunioni i fondamentali di Scrum erano rispettati, come la dimensione del team, non lavorare fuori dal backlog, il miglioramento dei processi e la revisione del lavoro.

Durante i meeting giornalieri relativi al debug, dove erano presenti tutti i leader e manager dell'organizzazione, gli sviluppatori del prodotto dimostravano ai Product Owners che i contenuti aggiunti incontravano i criteri precedentemente scelti. Il periodo di intenso debug e sviluppo durò un paio di settimane. Alla fine di questo periodo i team sopravvissuti estesero gli sprint a due settimane, che funzionano anche oggi.

1.5.Preparazione della Produzione

Mentre i team si preparavano per la produzione, si notò una certa tensione nel realizzare l'handover (passaggio di conoscenze). L'handover avviene quando le attività di responsabilità, conoscenza, azione e revisione vengono assegnate a nuovi responsabili.

Si formarono vari Task Force per affrontare i problemi legati all'uso del silicio. In Intel una Task Force è un team formato da esperti, che entra in azione in caso di crisi. Ogni membro è responsabile del successo della Task Force, chi viene scelto deve immediatamente abbandonare le sue precedenti mansioni e concentrarsi sulla nuova responsabilità.

Per capire come usare Scrum nella maniera più efficiente, fu organizzato un corso di Sviluppo del Prodotto usando i principi Lean, che mise in evidenza il seguente criterio:

- Team Multi-funzionali: i membri del team sono 100% specializzati nelle loro mansioni e non sono influenzati dai manager durante gli sprint.

Dopo la formazione di questi team l'handover migliorò notevolmente, i membri del team erano in grado di risolvere i problemi, la comunicazione avveniva senza intoppi, e se un particolare membro del team non era necessario durante lo sprint, contribuiva in un'altra area.

Il progetto prova fu portato a termine giusto in tempo per la riunione annuale con i manager leader. Era l'opportunità per convincere i manager ad accettare Scrum e fare alcune correzioni per farlo funzionare ancora meglio. Presentati i risultati, anche i meno convinti decisero di accettarlo.

1.6.Retrospettiva

In Intel si usò una semplice valutazione con + e – per indicare cosa andò bene e cosa male.

Definizione di "Fatto" (+)

Nello sviluppo dei microprocessori, testare significava mettere alla prova il lavoro dei team. Questo permise di scrivere buone best practice e, ancora più importante, scrivere ottimi criteri di accettazione. I Criteri di Accettazione (AC) permettevano di dettagliare i requisiti per la soddisfazione del cliente.

Fu anche implementato un semplice processo di verifica. Per completare una storia, lo sviluppatore e il Product Owner dovevano decidere insieme che i criteri di accettazione erano stati rispettati. Per misurare i risultati dell'incontro furono introdotti i termini Add, Save and Escape.

Gli Add erano nuovi criteri di accettazione che il Product Owner aveva aggiunto alla storia durante il processo di verifica e accettati dallo sviluppatore per lo sprint corrente. I Save erano bug individuati nello sprint corrente e indicavano il funzionamento del processo di verifica. Gli Escape erano bug creati in uno sprint precedente, trovati nello sprint corrente e indicavano cosa doveva essere migliorato.

Fu inoltre necessario definire un processo di validazione. La convalida doveva assicurare il funzionamento corretto della storia durante il rilascio del prodotto e non poteva essere generalizzata come per il processo di verifica. Quindi ogni Scrum team documentava le proprie regole, definendo cosa doveva essere fatto per la propria parte del prodotto.

Una storia era “fatta” solo se tutti gli incarichi erano stati portati a termine, verificati e convalidati.

Mancanza di fiducia (+)

Nella determinazione della velocità dello sprint successivo, non veniva data fiducia alle storia non “fatte”, sulla base della definizione scritta sopra. Fu necessario costringere i team a prestare la massima attenzione durante la verifica e la convalida. Se avevano assicurato di fare il 100% e facevano il 90% allora avevano fallito.

Nove giorni di Sprint (+)

Lo Sprint avveniva ogni nove giorni e le riunioni di Revisione, Retrospettiva e Pianificazione si tenevano un Venerdì sì e uno no. In questo modo il team era sempre fuori dallo sprint due weekend al mese. Questo contribuì a migliorare notevolmente la qualità della vita e il morale dei membri del team.

Nel weekend nel quale i team erano in mezzo a uno sprint, i membri potevano decidere tra di loro se dovevano lavorare per raggiungere i loro obiettivi.

Ritmo (+)

Il ritmo dei nove-giorni di sprint consentiva ai Product Owner e ai team di cambiare direzione, se necessario, ad intervalli frequenti. Questo ritmo contribuì a diminuire notevolmente i requisiti che si erano avuti nei precedenti progetti e i manager leader potevano vedere che i team erano in grado di produrre risultati reali un Venerdì si e uno no. I dati raccolti dimostrarono che ogni volta che uno sprint veniva interrotto si perdeva dal 10% al 20% della velocità.

Questo risultato fu chiamato “tassa interruzione sprint”. Si perdeva 10% della velocità se l’interruzione avveniva nella prima settimana dello sprint ed il 20% se avveniva durante la seconda. I manager furono messi al corrente di questo e iniziarono a rispettare la cadenza dei cicli di produzione. Fu anche introdotta la regola che ogni modifica fatta ad uno sprint attivo imponeva una rinegoziazione degli obiettivi.

Product Owner nel team (-)

Per facilitare la comunicazione tra i team e i Product Owner, a quest’ultimi fu permesso di partecipare alle riunioni. In alcuni casi funzionò abbastanza bene, ma in altri i PO dettavano le mansioni giornaliere impedendo un’onesta comunicazione tra i membri dei team. Questo spinse i team a organizzare riunioni senza la presenza del PO. Anche se queste situazioni sembrarono risolversi nel tempo, la capacità della

squadra di auto-organizzarsi diminuì. Quando si formarono i team multi-funzionali, questa pratica fu abolita.

Scrum Tool (+)

Scrum richiedeva la stesura di una dettagliata documentazione. Era necessario trovare modi per misurare i risultati ogni giorno. Creare uno strumento per fare ciò contribuì al successo del progetto.

Si partì dalla piattaforma XPlanner che, modificata tramite JAVA e SOAP, si trasformò in "XPlanner 2". Sulla base di questa nuova piattaforma fu creata un'apposita applicazione Windows.

Grandi Backlogs (-)

Gestire un backlog può diventare una sfida. Se un membro del team, in qualsiasi momento, aggiungeva qualcosa al backlog, gli altri membri potevano avere l'impressione di essere bombardati di richieste. Alcuni Product Owner volevano bloccare i backlog che non consentivano l'ingresso di input dagli stakeholders.

La piattaforma spiegata precedentemente catalogava le nuove storie in una categoria diversa da quelle accettate. Il PO era quindi in grado di esaminare ogni nuova storia, discutere con le parti interessate, e decomporre la storia in modo appropriato. Fu anche implementato un "freezer" (congelatore) per le storie che non era possibile accettare. Gli stakeholder potevano così monitorare le loro storie.

Punti della storia (+)

Poiché la maggior parte delle squadre non aveva una unità di misura standard, venivano usati i giorni. Si doveva fare attenzione al modo di esprimersi con i manager leader, che non avevano familiarità con Scrum. Quindi si decise di parlare di "punti" piuttosto che di "giorni" quando si rivedevano i dati con i manager.

Incarichi per meno di un giorno (+)

Non spendere più ore a fare stime fu liberatorio per i team quanto per i manager. Alle storie veniva assegnato un certo grado di difficoltà in "punti" e i compiti potevano essere solo elementi binari, "fatto" o "non fatto". Un compito durava sempre meno di un giorno, quindi se una persona ci metteva di più a svolgerlo si capiva che non era in grado di eseguirlo.

Osservazioni nello Scrum giornaliero (+)

Le unità di misura e le rappresentazioni visive erano di vitale importanza per il successo del progetto. Un grafico sullo sprint si dimostrò efficace nell'avvertire i team se rimanevano indietro e creò discussioni con i PO sullo stato del progetto prima del termine dello sprint. Lo Scrum Master era sempre aggiornato sul grafico, in questo modo nessuno era sorpreso dai risultati durante la riunione di controllo.

Revisione Incrementale (+)

Per non aspettare la riunione di revisione per avere l'approvazione del PO, si decise che il PO e lo sviluppatore ne avrebbero parlato ogni volta che una storia era pronta per la verifica. Questo eliminò la maggior parte delle sorprese durante la riunione nella quale il prodotto finale era rivisto e accorciò la durata della stessa.

Velocità (+)

Completa visibilità sul backlog, sul report e sui metodi di misura aiutò i manager a prendere decisioni basate sul lavoro attuale dei team. La velocità di ogni team forzò il PO a decidere la quantità del lavoro a seconda della capacità.

Supporto esecutivo (+)

Senza il sostegno del manager la transizione non sarebbe stata possibile. Il management team fornì la struttura e gli incentivi per chi aveva un ruolo leader nei team. Crearono anche disincentivi per chi non seguiva il processo. Questo portò a nessuna perdita delle risorse umane durante la transizione.

Cambio dei Comportamenti (+)

Il giusto modo di comportarsi non viene appreso senza pratica. Negoziando costantemente lo scopo, le priorità e i requisiti, rispettando le scadenze, tenendo d'occhio le modalità di misura e puntando all'auto-organizzazione dei team, lo Scrum sopravvisse e prosperò per ancora due anni dopo i primi passi.

1.7.Risultati

Scrum ha avuto un forte impatto in quattro aree: **Tempo di ciclo, Pianificazione, Morale e Trasparenza.**

Tempo di ciclo ridotto:

- Scrum ha portato a una riduzione del 66%

Pianificazione:

- Fu stabilita e mantenuta la pianificazione basata sulle capacità e una cadenza di due settimane per più di un anno
- Furono virtualmente eliminati gli impegni non rispettati

Miglioramento del morale:

- Miglioramento della comunicazione e della soddisfazione lavorativa
- Il team che era più giù di morale diventò il migliore

Aumento della trasparenza:

- Portò all'adozione dello stile CMMI e degli standard VER e VAL
- Scrum scoprì bug, strumenti non adeguati, e deboli abitudini ingegneristiche

La cadenza di nove giorni dello sprint forniva una certa prevedibilità, che portò effettivamente a meno sbagli nei requisiti del team, mentre i manager cercavano di evitare di pagare la tassa di interruzione.

La soddisfazione lavorativa deriva dagli obiettivi coerentemente definiti e raggiunti con la velocità decisa dai team. Il team si sentiva incredibilmente orgoglioso della propria capacità di rispettare gli impegni presi. Il morale aumentò e il ritmo diventò sostenibile. Molte pratiche e sistemi ingegneristici tradizionali furono messi in discussione, Scrum mise in evidenza le inadeguatezze. Questo portò a un investimento in ulteriore infrastrutture agili.

2.Ferrovie olandesi

2.1.Background

Le ferrovie olandesi sono tra le più utilizzate al mondo, fornendo il trasporto a 1,2 milioni di passeggeri al giorno. Si decise di implementare un nuovo sistema informativo che richiedeva meno interventi manuali e forniva ai clienti informazioni più accurate sui propri viaggi. Come parte di questo programma, fu costruito il sistema PUB (pubblicare) che controllava i display informativi e i sistemi di trasmissione audio in tutte le stazioni ferroviarie.

Il primo tentativo di costruire il sistema PUB fu eseguito utilizzando un approccio a cascata tradizionale. Dettagliate specifiche sui requisiti sono state consegnate al fornitore IT, pensando di poter avere un sistema completo senza il diretto coinvolgimento del cliente. Dopo 3 anni il progetto fu annullato perché il fornitore non era riuscito a fornire un sistema che funzionasse. Il cliente allora si rivolse all'azienda

Xebia perché costruisse un sistema PUB partendo da zero. E' stato introdotto un approccio agile tramite l'uso di Scrum, concentrandosi sulla stretta collaborazione con il cliente, sulla comunicazione e sul lavoro fatto con piccoli incrementi.

2.2. Impostazione

Si iniziò con un progetto di prova di 3 settimane per impostare tutto ciò che era necessario prima del primo sprint. Furono coinvolti il project manager, un architetto e uno Scrum Master.

La scelta di un Product Owner si rivelò una sfida. Non era possibile trovare una persona che avesse il tempo e le conoscenze per dare la giusta priorità ai requisiti. Furono quindi nominati due business analyst per la carica di Product Owner. Le due figure erano disponibili e avendo partecipato al primo tentativo di costruire PUB avevano acquisito le conoscenze per essere PO per diversi gruppi di clienti. Le decisioni di alto livello sulle priorità erano fatte da un Project Manager incaricato dal cliente. In generale, questo modello funzionò bene, però in alcune occasioni il Project Manager cambiava le priorità che erano state decise durante la riunione di pianificazione (fatta in sua assenza) e la riunione doveva essere rifatta.

A causa del precedente tentativo di costruire PUB, era disponibile una dettagliata documentazione dei requisiti necessari per le parti del sistema. I requisiti seguivano lo standard MIL (MIL-STD-498), però il modulo non era adatto alla pianificazione Agile. I requisiti non erano raggruppati in piccoli gruppi che potevano essere testati e dimostrati durante uno sprint. I Product Owner non avevano esperienza nel scrivere le storie utente, quindi lo Scrum Master li aiutò a produrre il backlog iniziale del prodotto contenente le storie utente per le prime iterazioni.

Il progetto faceva parte di un piano più grande che coinvolgeva più sistemi software, insieme alla costruzione e l'installazione dei display nelle stazioni ferroviarie. Per essere in grado di gestire i lavori, le scadenze erano molto importanti. Pertanto, si doveva fornire un quadro generale completo della pianificazione. Il problema fu affrontato dopo alcune prove, capendo anche la velocità con la quale si poteva lavorare.

2.3.Espansione del team

Dopo il periodo di prova, il progetto partì con un team di 7 persone che lavoravano in iterazioni di due settimane. All'inizio del progetto si decise di lavorare anche con due

specialisti indiani. Il team lavorava direttamente nell'azienda cliente per 6 settimane, per familiarizzare con l'ambiente lavorativo, i responsabili clienti e con il resto del personale.
Il primo passo per diventare un team era decidere come lavorare insieme. La riunione si svolse insieme ai membri indiani. Qui si presero decisioni riguardante la programmazione, l'uso degli strumenti, i target qualità, ore lavorative. Il risultato fu caricato in una pagina Wiki. Se una decisione veniva modificata, per esempio durante la retrospettiva, la pagina Wiki doveva essere aggiornata.

Durante le prime iterazioni, il team dimostrò di poter costruire, testare e dimostrare le storie utente che formavano il cuore del sistema. Questo era piaciuto al cliente: in comparazione con gli altri tentativi questa volta si potevano vedere i risultati molto prima e il cliente poteva avere più controllo sullo sviluppo del progetto.

Dopo le prime iterazioni, il progetto fu esteso: gli sviluppatori indiani ritornarono a casa e furono aggiunte risorse in India e nei Paesi Bassi per creare due team, ognuno con 5 sviluppatori, che condividevano un unico tester. Più tardi fu ulteriormente esteso a 3 team con 3 tester. La chiave del successo consisteva nel avere risorse sia in India che nei Paesi Bassi, perché questo permetteva di avere una maggiore velocità e alta qualità.

Per lavorare insieme dai vari paesi si usava Skype, per meeting individuali e con tutto il team. In questo modo non fu necessario fare ulteriori investimenti in altri tipi di comunicazione. Poi si decise di fare la programmazione condivisa solo con risorse residenti nello stesso paese, nonostante gli strumenti a disposizione era necessario lavorare fisicamente insieme per un risultato accettabile. Come ultimo criterio, si usava ScrumWorks come strumento per monitorizzare chi lavorava su cosa, il progresso dello sprint e anche l'uso del backlog.

Ci furono alcuni problemi per implementare questo modello distributivo, primo fra tutti i Product Owners non si sentivano a loro agio a parlare in inglese. Secondo i principi Scrum la riunione per la pianificazione dello sprint consiste in due parti. Nella prima il Product Owner e il team decidono insieme le storie utente e definiscono le priorità per lo sprint; considerando il problema di una lingua comune, alla fine si decise di fare i meeting in olandese senza i membri indiani. La seconda parte della riunione usualmente consiste nell'identificazione dei compiti per implementare le storie utente. Questa parte si decise di farla in inglese, insieme ai membri indiani attraverso Skype,

però senza i Product Owner. La prova degli sprint fu tenuta solo localmente e in olandese. Per aggiornare i membri del team in India, i membri Olandesi scrissero una newsletter documentando i progressi.

2.4.Team per raccogliere informazioni

Il progetto faceva parte di una catena di applicazioni e doveva essere adattato all' infrastruttura del cliente. I Product Owner avevano approfondite conoscenze sui principali requisiti, però non sulla sicurezza, registrazione, indici di prestazione etc. Fu complicato ottenere queste informazioni dal cliente, si sono dovute avere multiple riunioni con vari dipartimenti. Per risolvere il problema si decise di costituire un team che avesse come focus l'ottenere informazioni "non funzionali" che permettessero al team di adattare le storie utente al backlog.

2.5.Documentazione

Il cliente richiedeva una dettagliata documentazione che rispettasse gli standard MIL. Considerando che doveva essere in olandese, era chiaro che solo risorse olandesi potevano farlo. Tuttavia i membri del team non erano familiari con lo standard MIL e scrivere istruzioni non era il loro compito, quindi si decise di assumere uno scrittore specializzato al MIL. Questo permise al team di concentrarsi sui risultati del progetto. La decisione si rivelò di successo, però era necessaria un'ottima e costante comunicazione tra lo scrittore e i membri del team, che doveva essere tenuta sotto controllo.

2.6.Esigenze

I Product Owner, che erano business analyst, usualmente scrivevano dettagliate documentazioni in Olandese. Per il progetto potevano bastare le storie utente sul backlog e la documentazione fornita dai Product Owner, però il cliente esigeva una documentazione più precisa. Per adattare questa richiesta al processo Scrum si decise di tradurre i requisiti in storie utente con la collaborazione del Product Owner. Il risultato fu che i requisiti erano tenuti i due categorie, quelli documentati e i backlog, e certe volte questo creò problemi quando si dovevano fare degli aggiornamenti.

2.7.Testare

Durante il progetto si usarono software che permisero di avere risultati testati alla fine di ogni sprint, e senza bug. I test furono automatizzati su due livelli: prove di reparto e prove di accettazione. Questa procedura portò vari vantaggi: i bug venivano trovati e

corretti durante lo sprint e i tester potevano operare all'inizio dello sprint testando prima dell'implementazione delle storie utente.
Ci furono problemi con quest'approccio in un'area, dove parte del sistema era un'applicazione con una complicata interfaccia utente. In questo caso si sono dovuti fare i test manualmente. Con la crescita del sistema, i test di regressione chiedevano più tempo e peggio ancora, i bug venivano trovati solo in quest'area. La lezione appresa fu che, anche se i test automatici a volte sono difficili da implementare, ne vale la pena e i risultati si vedono soprattutto verso la fine del progetto.

2.8.Risultati

Il cliente fu contento dei risultati. Un importante fattore è stato discutere degli indicatori di successo con il cliente durante l'avanzamento del progetto. Il cliente chiese a un'agenzia esterna di fare l'audit per il software. Le conclusioni furono:

- La manutenibilità del sistema era molto buona
- La qualità del progetto era alta

Durante la presentazione dell'audit, gli auditor commentarono che non avevano mai avuto un progetto da analizzare così buono.

2.9.Conclusione

Le lezioni più importanti apprese durante il progetto sono state:

- Può essere difficile trovare un Product Owner che abbia tutte le necessarie conoscenze. Spesso è inevitabile la scelta di più persone come Product Owner, soprattutto in grandi progetti;
- Anche se rispettare una scadenza è importante, è essenziale assicurarsi che il backlog del prodotto sia completo. Riguardo i requisiti, ogni stima è migliore di nessuna stima, anche se si ha a disposizione limitate informazioni;
- Scrum è adatto a team distribuiti in vari paesi;
- Per progetti con team remoti è meglio iniziare con una riunione per ogni location;
- I test automatizzati sono vitali per migliorare i software, senza la presenza dei bug.

Conclusioni

L'obiettivo di questo elaborato era di valutare l'adozione delle metodologie Agili, in particolare della tecnica Scrum, e stabilire se queste metodologie abbiano influenzato anche il modo di lavorare di chi segue approcci più tradizionali.

Lo studio evidenzia che negli ambienti odierni, caratterizzati da veloci ritmi di produzione, alta incertezza e dove il cambiamento dei requisiti è all'ordine del giorno, non è possibile realizzare pianificazioni a medio/lungo termine. Questo rende inefficaci gli approcci tradizionali e richiede l'adozione di metodi che possano accogliere e gestire questo concetto del "cambiamento", ovvero dei metodi agili.

Analizzando i due casi studio che prendono in considerazione il passaggio da un approccio tradizionale a quello agile, si nota che, come qualsiasi nuovo processo che modifichi radicalmente il business, i metodi agili hanno generato una serie di difficoltà. Mentre è vero che molte delle pratiche associate con lo sviluppo Agile esistono da un discreto lasso di tempo, nella media i team di sviluppo software devono ancora abbracciare gran parte di tali principi e pratiche. Le metodologie agili richiedono essenzialmente un diverso approccio mentale, oltre che l'acquisizione di nuove competenze, e quindi lanciano una sfida all'approccio convenzionale.

Difficoltà

La metodologie Agili sono difficili in quanto richiedono ulteriori test e la partecipazione attiva dei committenti. Hanno inoltre un impatto maggiore sul management rispetto che sugli sviluppatori. Al management viene richiesta una maggiore apertura mentale, e di essere attivamente coinvolto nel processo di sviluppo e soprattutto di consentire ai team di prendere decisioni in modo indipendente.

Disciplina

L'Agile richiede molta più disciplina delle tecniche tradizionali. In primo luogo, durante lo sviluppo, il codice può essere integrato in modo continuo, a volte anche dopo ogni aggiornamento. Occorre inoltre ammettere che nello sviluppo Agile sono richieste competenze tecniche e manageriali più elevate. In terzo luogo, ogni funzionalità deve essere completata prima di passare alla successiva.

Pianificazione

Rispetto a quelle tradizionali, nelle metodologie agili è necessaria una maggiore e più raffinata forma di pianificazione. Poiché la pianificazione viene effettuata frequentemente (quanto meno per ogni iterazione), e il piano viene aggiornato come e quando necessario, deve essere fatta una pianificazione più mirata.

Supporto

Infine, per implementare le metodologie Agili sono necessari diversi tipi di supporto: organizzativo, infrastrutturale e logistico, a livello di team.

Per concludere, il lavoro evidenzia che, benché lo sviluppo software secondo un approccio agile sia perfetto per piccoli progetti, questo può essere adottato anche nel caso di organizzazioni di dimensioni più elevate, il che dimostra la potenza e la scalabilità di tali metodologie.

Bibliografia

Danube Technologies, Inc (http:\\danube.com), 8 Settembre 2014

Davide Vernole, 2009, *“Scrum e Team Foundation Server per gestire lo svillupo software”, Tool News 7/2009, pp 27-29*

Derby, Larsen, 2006, *Agile Retrospectives: Making good teams great,* Pragmatic Bookshelf, Washington DC

Ken Schwaber, 2003, *Agile Project Management with Scrum,* Microsoft Press

Ken Schwaber, 2007, *The Enterprise and Scrum,* Microsoft Press

Ken Schwaber, Jeff Sutherland, 2011, *The Scrum Guide – The definitive guide to Scrum: The rules of the game*

Marco Mulder, Martin van Vliet, 2008, *Distributed Scrum Project for Dutch Railways*

Marina Gil Santamaria, 2007, *Agile&Scrum: What are these methodologies and how will they impact QA/testing roles?,* http:\\www.oracle.com

Microsoft, (http:\\microsoft.com), 8 Settembre 2014

Michael James, Danube Technologies, Inc, *Scrum, pp 1-6*

Mike Cohn, 2004, *User Stories Applied,* Addison-Wesley Professional, Crawfordsville

Mike Cohn, 2005, *Agile Estimation and Planning,* Pearson, Upper Saddle River

Pat Elwer, 2008, *Agile Project Development at Intel: A Scrum Odyssey, pp 1-14*

Pravin Mukhedkar, Mukesh Jain, 2008, *Scrum Management: An offshore perspective*

Project Management Institute, 2004, *A guide to the Project Management Body of Knowledge (PMBOK Guide) Third Edition,* Project Management Institute, Pennsylvania