Politecnico di Torino

Politecnico di Torino

Corso di Laurea Magistrale Ing. Gestionale

A.a. 2022/2023

Sessione di Laurea Dicembre 2023

Applicazione del framework Scrum su un progetto di migrazione software

Relatore:

Professore Giovanni Zenezini

Candidato:

Carlo Maria di Corrado

Abstract

La metodologia Agile, con particolare attenzione al framework Scrum, ha rivoluzionato l'approccio alla gestione dei progetti, promuovendo la flessibilità, la collaborazione e la consegna incrementale dei risultati. Questa tesi esplora l'applicazione di Scrum in un contesto di migrazione software, in un settore come quello IT noto per le sue sfide uniche e i requisiti complessi. Lo scopo principale dello studio è stato valutare gli effetti dell'implementazione di Scrum su un team senza esperienza pregressa in questa metodologia.

I risultati chiave di questa tesi indicano che, sebbene l'applicazione di Scrum abbia permesso al team di portare a termine il progetto con successo, essa potrebbe non essere l'approccio ideale per progetti di migrazione software. La metodologia Agile ha favorito la trasparenza e la comunicazione all'interno del team, migliorando la comprensione delle priorità del progetto. Tuttavia, l'adattabilità e il focus sulla consegna rapida potrebbero non essere sempre in linea con le esigenze di progetti di migrazione, dove i requisiti risultano essere già ben definiti ma che richiedono una approfondita comprensione, una pianificazione accurata, una gestione dei rischi e una conoscenza del contesto tecnico del progetto.

Questo studio offre importanti spunti sul project management, evidenziando le sfide e le soluzioni specifiche che possono emergere quando si applica Scrum a progetti di migrazione software. Suggerisce inoltre l'importanza di valutare attentamente le decisioni e gli accorgimenti presi durante lo svolgimento del progetto in modo tale che possano essere adatti a contesti di progetto di questo genere, tenendo conto delle esigenze e delle caratteristiche uniche di ciascun progetto e dei partecipanti coinvolti.

Indice

Introduzione

L'elaborato è stato realizzato grazie all'aiuto della società Aubay Italia S.p.a, la quale mi ha dato modo di poter seguire da vicino tutti i passaggi riguardanti la gestione di un progetto mediante la metodologia Agile con framework Scrum.

Questa esperienza mi ha permesso di costatare l'utilizzo dell'Agile come risposta alla necessità di gestire progetti in contesti evolutivi su un caso concreto di migrazione software, ottenendo dei risultati che hanno confermato in parte quanto descritto in letteratura, ma che hanno evidenziato l'importanza di dover adattare le nozioni teoriche al contesto del progetto in esecuzione.

Tale studio ha come obiettivo quello di valutare l'idoneità dell'utilizzo del framework Scrum su processi di migrazione in ambito IT, analizzando i successi e le difficoltà emerse durante il progetto su un Team alla prima esperienza con questa metodologia. I risultati ottenuti porteranno alla stesura di una soluzione alternativa per la gestione del progetto, che a differenza della stragrande maggioranza di quelli appartenenti al settore IT, è caratterizzato da requisiti già ben definiti provenienti dalla versione precedente del software, già implementata dalla medesima società.

La tesi inizia con una overview generale sulla nascita dell'Agile e delle varie necessità sorte nel corso degli anni per far fronte alla complessa gestione dei progetti in ambito IT, facendo poi un confronto anche con le metodologie tradizionali di gestione dei progetti come quello Waterfall. Segue poi una descrizione sulla finalità della migrazione del software e sulla filosofia adottata per la gestione del progetto, che sarà caratterizzata dall'applicazione dei principi Scrum adattati al contesto in cui si trova ad operare il team. Di seguito, verrà analizzato lo svolgimento dello stesso commentando gli Sprint chiave e i macro-temi emersi dall'applicazione di Scrum, che hanno generato delle difficoltà al team superate grazie al continuo processo di apprendimento dello stesso e dalle decisioni strategiche prese dallo Scrum Master. Infine, verranno elencati gli aspetti di contesto che hanno favorito o intralciato l'applicazione dello Scrum, arrivando alla conclusione che le tecniche Agile si sono rivelate utili alla corretta esecuzione ma forse non perfettamente idonee a progetti di migrazione software.

1. La metodologia Agile

L'Agile risulta essere ad oggi la principale tecnica utilizzata per la gestione dei progetti in ambito IT, basandosi sui concetti di "autorganizzazione" e "iterazione" che permettono ad un team di sviluppo software di superare i limiti tipici dell'applicazione del modello tradizionale *Waterfall,* come l'errore nell'interpretazione del requisito del cliente e il mancato rispetto dei tempi di consegna del software.

1.1 La sua nascita

I principi cardine della metodologia Agile furono stabiliti agli inizi degli anni 90, quando un team di sviluppatori e consulenti ritennero che le tecniche attuali di gestione dei progetti non fossero funzionali e applicabili ad un settore soggetto a frequenti cambi tecnologici e ad alta competitività come quello dello sviluppo software. L'attenzione viene spostata sul cliente, il quale diventa l'attore principale per il quale non è più sufficiente la sola differenziazione in termini di costi e prodotto, ma diventa necessario un elemento aggiuntivo: la customer experience. Con la crescita esplosiva di internet che rende la velocità di risposta al mercato un fattore sempre più determinante per il successo, i modelli tradizionali iniziano a risultare inadeguati.

Questo portò all'esigenza di un approccio alla gestione di progetto che permetta principalmente una rapida risposta ai cambiamenti esterni ed interni, potendo riorganizzare il progetto in itinere. Questi nuovi approcci si basano su: la stretta collaborazione tra il team di sviluppo e gli stakeholder aziendali, la consegna frequente di un "pezzo" di software, team affiatati e autorganizzati che utilizzano modi intelligenti di creare, confermare e consegnare il codice. Con il passare del tempo gli ideatori di queste metodologie, ritenendo che altri potessero essere interessati a condividere parte dei vantaggi sperimentati, hanno creato dei framework per diffondere le idee a team in altre organizzazioni e contesti. Iniziano così a comparire framework come Scrum, Extreme Programming, Feature-Driven Development (FDD) e Dynamic Systems Development Method (DSDM), per poi arrivare nel 2001, anno in cui fu pubblicato il "Manifesto Agile" al fine di definire i valori e i principi portanti di questa filosofia.

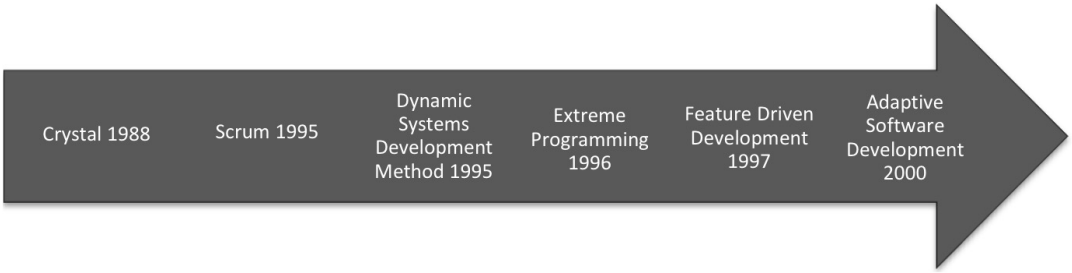

Figura 1. Framework Agile

1.2 I principi dell'Agile

L'Agile prende spunto dal "Toyota Production System", un sistema di organizzazione del lavoro basato sull'efficienza e efficacia di processi che garantiscono la produzione di un bene che miri alle esigenze del cliente, eliminando tempi morti, sprechi, sovraproduzione e sottoproduzione. Da questo sistema l'Agile riporta l'idea di lavorare in piccoli lotti in previsione di piccoli rilasci frequenti di un prodotto, piuttosto che una produzione di grandi dimensioni, focalizzando l'attenzione sul team di sviluppo e sul coinvolgimento del cliente stesso in fase di impostazione del lavoro. Ispirandosi al concetto di Kaizen, il progetto va revisionato costantemente. Si punta al miglioramento continuo, cambiando i fattori dove necessario in base a nuove informazioni o feedback restituiti dal sistema. Il risultato finale è il valore per il cliente; con tale metodologia si è in grado di riadattarsi costantemente alle esigenze del cliente per mezzo di un processo iterativo e di ridurre il rischio di fallimento.

Questi concetti furono poi delineati nel già citato "Manifesto Agile" che riporta i seguenti quattro principi guida:

- Gli individui e le interazioni più che i processi e gli strumenti
- Il software funzionante più che la documentazione esaustiva
- La collaborazione col cliente più che la negoziazione dei contratti
- Rispondere al cambiamento più che seguire un piano

Partendo dal primo, vediamo come l'attenzione ricada sugli "individui" ovvero il team di sviluppo, ma anche il cliente e le "interazioni" periodiche che si generano tra questi. Infatti, per poter affrontare al meglio la complessità organizzativa di un progetto in ambito IT,

derivanti dalla coordinazione del team e dalla corretta individuazione del requisito del cliente, è fondamentale definire varie tipologie di "incontri periodici" dette anche *Cerimonie*. Queste permetteranno al team di auto-organizzarsi per definire al meglio le architetture, i requisiti dettati dal cliente e la progettazione. Incontri a intervalli regolari, permettono inoltre al team di riflettere su come rendere il lavoro più efficace, regolando il proprio comportamento in base alle dinamiche e le riflessioni che emergono. Infine, una conversazione faccia a faccia è il modo più incisivo ed efficace per comunicare ed essere costantemente allineati sul lavoro da portare al termine.

Il secondo punto si focalizza sul prodotto, ovvero sul software che deve essere implementato. Nell'Agile, oltre che ad essere il bene finale da consegnare, è anche un metro di misura: rilasciando periodicamente del software funzionante è possibile fare una overview generale sul *Debito Tecnico* che deve ancora essere implementato e delle *Features* (funzionalità) già presenti e osservabili dal cliente.

La collaborazione con il cliente, rappresentato nella stragrande maggioranza dei casi da un *Product Owner* (PO), figura che approfondiremo nei paragrafi successivi, permette al team di captare eventuali cambiamenti di requisiti e mostrare con cadenza periodica porzioni di software funzionante, che costituiscono per il cliente stesso del valore aggiunto. Il cambiamento dei requisiti da parte del cliente in corso d'opera, anche in stati avanzati di sviluppo, nell'Agile vengono visti come una opportunità per poter generare maggior vantaggio competitivo.

L'ultimo macro-principio dell'Agile si estrinseca nella prontezza di risposta al cambiamento. Le *Change Request* (CR) da parte del cliente sono spesso riconducibili a cambiamenti dettati dalle esigenze del mercato e all' errore nello stimare l'implementazione di una determinata feature del software. In entrambi i casi, il team deve essere pronto a far fronte a questi cambiamenti in qualsiasi momento, senza distorcere le dinamiche organizzative instaurate nel tempo e individuando nel minor tempo possibile le modifiche da apportare.

1.3 Confronto con la metodologia Waterfall

Prima che l'Agile diventasse la metodologia maggiormente utilizzata per la gestione dei progetti di sviluppo software, i team prediligevano l'utilizzo della metodologia Waterfall, una forma lineare di gestione dei progetti ideale per commesse in cui il risultato finale è chiaramente stabilito fin dall'inizio. Le aspettative per il progetto e i risultati di ogni fase devono essere chiari e sono necessari per passare alla fase successiva. Implementare un modello *a cascata* è molto semplice, poiché i concetti di base riguardanti la struttura del modello rimangono invariati indipendentemente dalla dimensione e dalla tipologia del progetto. A differenza dell'Agile non è possibile individuare dei principi guida da rispettare, bensì dei veri e propri step da seguire in ordine sequenziale. Questi passaggi possono essere raggruppati in:

- Analisi dei requisiti
- Architettura e Design
- Pianificazione
- Implementazione
- Testing
- Rilascio

L'analisi dei requisiti sancisce una delle principali differenze tra il Waterfall e l'Agile. Questa fase risulta essere centrale in tutte e due le metodologie, ma viene applicata con dinamiche diverse. È stato sottolineato negli scorsi paragrafi come la fluidità delle metodologie agili permettono un continuo confronto tra il team e il cliente, rendendolo partecipe dello sviluppo e raccogliendo eventuali modifiche e nuovi requisiti che possono emergere in itinere. Ciò implica che il processo di raccolta dei requisiti va di pari passo con lo svolgimento del progetto e volgerà al termine nel momento in cui il prodotto finale rispecchi in pieno quanto richiesto dal cliente. Il Waterfall, segue uno schema sequenziale, prevede il completamento di ciascuna fase per passare a quella successiva. È, dunque, essenziale in fase di inizio di progetto definire al meglio tutti i requisiti per eseguire correttamente gli step successivi. Consegue che il metodo a cascata si dimostra maggiormente appropriato per progetti dove l'output risulta già ben definito a priori, con variazioni in corso d'opera minime o assenti. Avere dei requisiti ben definiti prima dell'inizio del progetto potrebbe essere un vantaggio per il team di sviluppo, in quanto renderebbe la pianificazione e la progettazione più semplici. Inoltre, gli sviluppatori e i

clienti concordano su ciò che verrà consegnato già all'inizio del ciclo di vita del processo, rendendo così possibile una consegna più rapida del progetto. Nel campo di sviluppo software una raccolta esaustiva dei requisiti appare estremamente complicata poiché i problemi legati alla corretta definizione di questi sono molto frequenti. Con l'Agile questo problema non si presenta, in quanto i requisiti vengono confermati più volte durante lo svolgimento del progetto garantendo un maggior coinvolgimento del cliente per tutta la durata del processo e non soltanto nella fase iniziale.

Lo step successivo è quello di progettazione, una delle fasi più critiche del Waterfall, poiché il team è spinto a creare in anticipo un'architettura che rispetti tutti i requisiti. Qui si definisce sia la struttura di massima (architettura di alto livello) sia le caratteristiche dei singoli moduli, individuando i componenti necessari e le loro caratteristiche, attraverso una effettiva scomposizione gerarchica e dettagliata. Questa fase è molto delicata e richiede particolare attenzione, poiché l'architettura realizzata rimarrà invariata per tutto il processo. L'esperienza insegna che nessuna architettura definita a inizio progetto può essere completamente corretta durante un processo di sviluppo software. In ragione di ciò l'Agile nasce per poter trattare al meglio questa incertezza, gestendo con facilità eventuali cambiamenti di architettura e design di un progetto.

L'implementazione è la fase operativa di *Coding* (in caso di sviluppo software), fatta tenendo conto di tutte le informazioni raccolte durante le prime due fasi di progetto.

Finito il coding si effettuano dei test sul prodotto realizzato prima di rilasciarlo al cliente. Se le esigenze del cliente non sono state colte bene all'inizio o sono cambiate dall'inizio del progetto, il collaudo del prodotto potrebbe arrivare troppo tardi per poter apportare grandi modifiche. Il cliente deve quindi trovare un budget aggiuntivo per ottenere il prodotto di cui ha bisogno. Nell'Agile il testing avviene regolarmente durante l'intero processo, in modo che il cliente possa verificare periodicamente che il prodotto sia quello che aveva immaginato. In questo modo è più probabile che il progetto venga portato a termine nei tempi e nei budget previsti.

Una volta rilasciato il prodotto, nel Waterfall è difficile tornare indietro e apportare modifiche, risulta quindi fondamentale che le fasi antecedenti al rilascio siano state svolte con la massima precisione. Utilizzando l'Agile viene rilasciata periodicamente una versione funzionante di software, in modo che il cliente possa definire il modo in cui viene costruito. Vedere una versione funzionante fin dalle prime fasi del progetto permette al

cliente di modellare il prodotto in base alle sue esigenze. Questo è più difficile da attuare con il metodo a cascata, perché il cliente deve delineare tutte le sue preferenze in anticipo senza poter vedere un pezzo del prodotto funzionante.

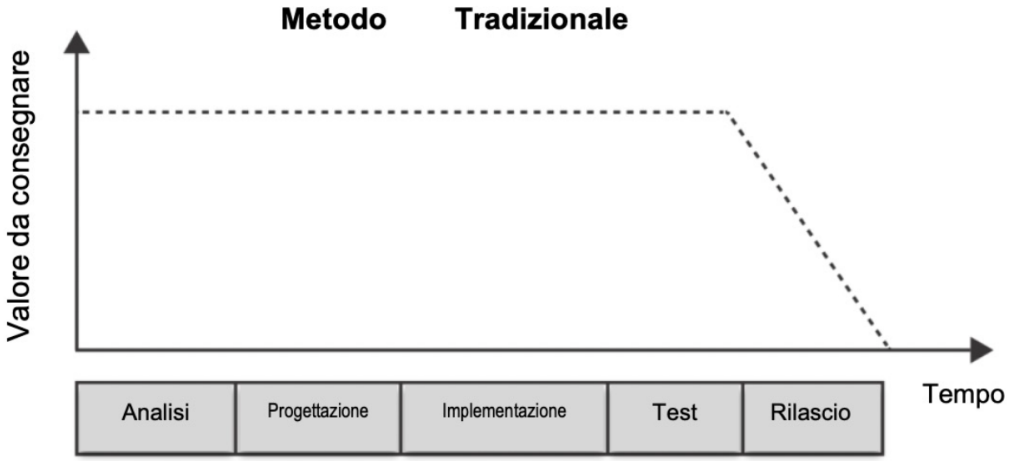

Figura 2. Timeline metodologia Waterfall

1.4 Il framework Scrum

Tra tutti i framework citati nel paragrafo 2.1, lo Scrum è ad oggi il più utilizzato nel mondo dei progetti di sviluppo software, come lo è stato anche per lo svolgimento del progetto descritto in questa tesi.

Lo Scrum non è un processo standardizzato in cui si segue metodicamente una serie di step, è un framework per l'organizzazione e gestione del lavoro. Questo framework si basa su un insieme di valori e principi che forniscono le basi sulle quali il team aggiungerà le sue caratteristiche di implementazione tecniche e approcci. Per cui ogni progetto avrà un team che imposterà il proprio Scrum in base alle proprie esigenze. I principi, i valori e le pratiche di tale framework non posso essere stravolte, ma si possono personalizzare in base alle caratteristiche delle organizzazioni che la applicano. I principi del framework Scrum sono:

- Controllo Empirico del processo: prendere decisioni basandosi sull'osservazione e sulla sperimentazione piuttosto che su una dettagliata pianificazione a priori.

- Auto-Organizzazione: valore maggiore dal gruppo auto-organizzato rispetto alla somma dei singoli. Maggiore coinvolgimento ed assunzione di responsabilità.
- Collaborazione: lo sviluppo di un prodotto è un processo condiviso di creazione di valore. Lavorare insieme per ottenere un obiettivo comune.
- Prioritizzazione basata sul valore: attenzione per la consegna del massimo valore per il business per tutta la durata del progetto
- Time-boxing: il tempo è il vincolo principale. Aiuta la gestione della pianificazione ed esecuzione del progetto. Si applica a Sprint, Daily Standup meeting, Sprint Planning meeting, Sprint Review meeting.
- Sviluppo Iterativo: ogni modifica richiesta dal cliente può entrare a far parte del progetto.

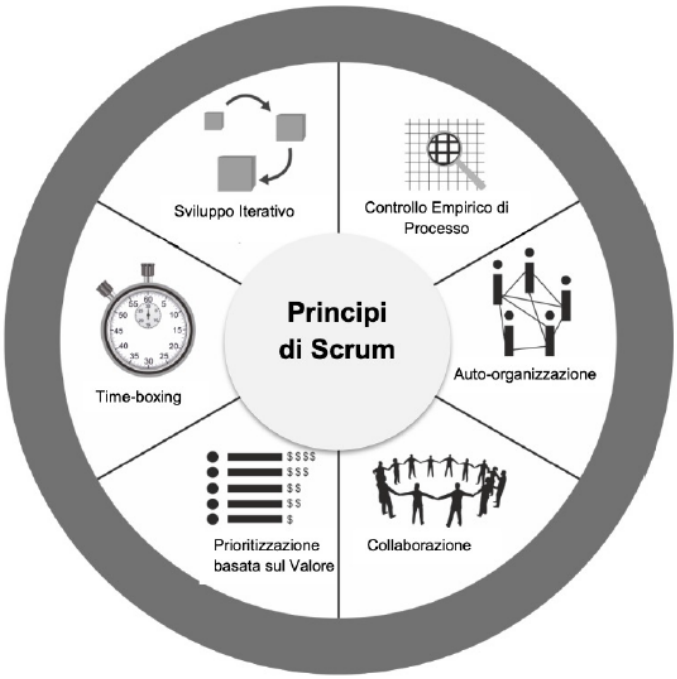

Figura 3. I principi del framework Scrum

1.4.1 Scrum Team

Lo Scrum Team è formato dal Product Owner, uno Scrum Master e dal team di sviluppo (Dev Team).

I Team Scrum decidono come svolgere il lavoro e hanno tutte le competenze necessarie per svolgere le attività senza dover dipendere da nessuno al di fuori del gruppo. Il tutto viene impostato per ottimizzare la flessibilità e la creatività.

Il *Product Owner (PO)* è la voce del cliente e degli stakeholder. Ha la responsabilità di tenere aggiornato il *Product Backlog*, ovvero la lista contenente tutti i requisiti espressi dal cliente scritti sotto forma di *User Story*. Il PO deve cercare di massimizzare il valore per il cliente facendo in modo che i requisiti siano espressi in modo chiaro, siano ordinati e ottimizzino il valore del lavoro del team di sviluppo. Le decisioni del Product Owner sono visibili nel contenuto e nell'ordine delle priorità del Product Backlog ed è fondamentale che siano rispettate da tutti e che siano le uniche a guidare il Dev team durante tutto il progetto.

Lo *Scrum Master* è responsabile di promuovere e sostenere Scrum, aiutando il team a comprendere la teoria, le pratiche, le regole, ed i valori di Scrum. Inoltre, svolge anche il ruolo di facilitatore all'interno del team ed è al servizio del Product Owner, fornendogli dei consigli sulla prioritizzazione delle attività da eseguire.

Il *team di sviluppo (Dev Team)* è formato da Tecnici, nel caso di progetti di sviluppo software da Sviluppatori/Programmatori che lavorano per produrre un incremento del prodotto finale alla fine di ogni sprint. Scrum non riconosce alcun titolo ai membri del team di sviluppo, indipendentemente dal lavoro eseguito dalla persona. Ogni membro del team può essere specializzato su un tema in particolare o avere più esperienza rispetto agli altri, ma i successi e i fallimenti vengono distribuiti in ugual misura tra tutti i membri. La dimensione del team deve essere compresa tra le tre e le nove persone, esclusi Product Owner e Scrum Master. Avere meno di tre persone all'interno del team diminuisce l'interazione e comporta un minore guadagno in termini di produttività. Invece più di nove persone richiede un eccessivo lavoro di coordinamento.

1.4.2 Gli eventi Scrum

Gli eventi descritti dal framework sono finalizzati a creare regolarità e ridurre al minimo la necessità di riunioni straordinarie. Tutti gli eventi sono limitati temporalmente, così da avere una durata massima fissa. Quando uno Sprint inizia, la sua durata è prefissata e non può essere né accorciata né allungata.

Lo Sprint si svolge in un periodo ben definito a priori, solitamente di un mese o due settimane nel quale viene creato un pezzo di prodotto utilizzabile e utile a mostrare alcune funzionalità al cliente. Gli Sprint si svolgono in maniera sequenziale partendo immediatamente dopo la conclusione del precedente. Per la definizione e gestione dello Sprint vengono svolti degli incontri, anche questi di durata ben definita e con obiettivi ben delineati: Sprint Planning, Daily Scrum, Sprint Review e lo Sprint Retrospective. Solo il Product Owner ha l'autorità di annullare lo Sprint e può farlo anche sotto l'influenza degli stakeholder, del team di sviluppo o dello Scrum Master.

- *Sprint Planning Meeting*: è un incontro che avviene all'inizio dello Sprint per pianificare il lavoro che deve essere svolto da tutto il Team Scrum. Si fa riferimento al Product Backlog, all'ultimo incremento del prodotto rilasciato e al carico di lavoro sopportabile del team di sviluppo durante uno Sprint. Il numero di elementi selezionati dal Product Backlog per lo Sprint è definito esclusivamente dal team di sviluppo ed è l'unico in grado di valutare il lavoro che può essere svolto al prossimo Sprint. Durante lo Sprint Planning, lo Scrum Team modella anche uno *Sprint Goal*, che è un obiettivo che deve essere raggiunto all'interno dello Sprint attraverso l'implementazione del Product Backlog.

- *Daily Meeting*: viene chiamato anche *Stand-up meeting* ed è un evento della durata massima di 15 minuti, durante il quale il team di sviluppo pianifica il lavoro per le prossime 24 ore, ma serve anche per migliorare la comunicazione, eliminare altri incontri, identificare gli eventuali ostacoli allo sviluppo, evidenziare e promuovere il rapido processo decisionale e migliorare il livello di conoscenza del team. Il Daily Scrum si svolge ogni giorno allo stesso orario e nello stesso luogo per ridurre la complessità. Il team di sviluppo utilizza questa riunione per ispezionare l'avanzamento verso lo Sprint Goal e come tale avanzamento stia procedendo verso il completamento del lavoro del Product Backlog. Ogni giorno il team di sviluppo

dovrebbe capire come ha intenzione di lavorare per raggiungere lo Sprint Goal e realizzare l'incremento entro la fine dello Sprint.

- *Sprint Review*: viene fatto alla fine di ogni sprint per controllare che il lavoro impostato durante lo Sprint Planning sia stato effettivamente svolto ed eventualmente adattare il Product Backlog. A partire da questo e dai cambiamenti apportati al Product Backlog durante lo Sprint, i partecipanti si confrontano su come si potrebbe ottimizzare il valore negli sprint successivi. Si tratta di un incontro informale dove viene presentato l'output derivante dal lavoro svolto durante lo sprint per invogliare il team a rilasciare feedback a riguardo e promuovere la collaborazione.

- *Sprint Retrospective*: è un'occasione per lo Scrum Team per ispezionare sé stesso e creare un piano di miglioramento da attuare al prossimo Sprint. Lo Sprint Retrospective si svolge dopo lo Sprint Review e prima del successivo Sprint Planning. È una riunione della durata massima di tre ore per uno Sprint di un mese. Lo Scrum Master si assicura che l'evento abbia luogo e che i partecipanti ne comprendano l'effettiva finalità.

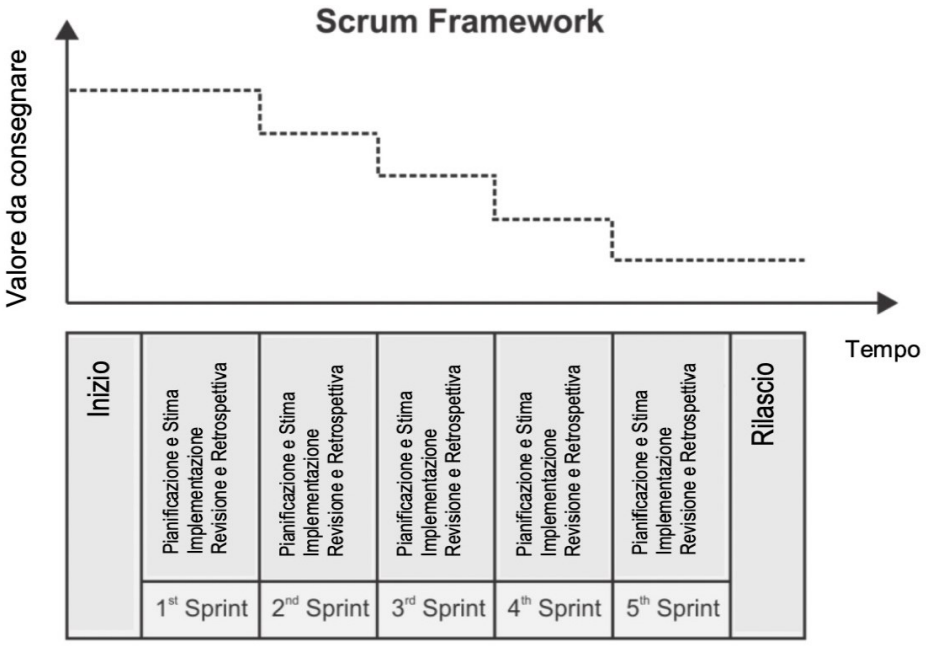

Figura 4. Timeline metodologia Scrum

1.4.3 Gli artefatti

Gli artefatti sono gli strumenti utilizzati per fornire trasparenza e opportunità di ispezione e adattamento. Gli artefatti definiti da Scrum sono specificatamente progettati per massimizzare la trasparenza delle informazioni chiave, in modo che ognuno abbia la stessa comprensione dell'artefatto.

Il Product Backlog è l'elenco di *attività* necessarie a raggiungere la visione di progetto. Ogni attività (*Epica*) rappresenta del valore per il cliente. È ordinato per priorità: in cima ci sono le funzionalità di maggiore valore per il cliente, o quelle più rischiose. Il Product Backlog cambia nel tempo, evolve così come evolve il prodotto e l'ambiente stesso nel quale sarà utilizzato; è dinamico e cambia continuamente sotto le indicazioni del cliente e del mercato per poter essere appropriato e competitivo. Il Product Owner è l'unico responsabile della gestione, aggiornamento e prioritizzazione del Product Backlog. L'*Epica* è una funzionalità di grandi dimensioni che non può essere completata in uno sprint e vengono create dal Product Owner raccogliendo le esigenze degli stakeholder e successivamente suddividendole in *User Story*. Le epiche sono quindi scomposte in User Story e sono scritte mediante frasi che esprimono una funzionalità dettata da un utente attraverso un linguaggio comprensibile a tutto il team di sviluppo. In particolare, per la scrittura delle US viene utilizzato il formato 3W: Who, What, Why. Esempio: *Io come *persona* vorrei poter *requisito* in modo da *beneficio**. Ogni User Story crea del valore per il cliente. Durante lo Sprint Planning, le User Story sono scomposte in *Task,* che definiscono le operazioni tecniche devono essere prese in carico e chiuse per rilasciare una US. Questo livello di scomposizione da Epica a Task viene utilizzato in tutti i progetti Agile, compreso quello che analizzeremo nei capitoli successivi. Questa suddivisione è essenziale per lo svolgimento di un progetto, poiché partendo da una descrizione di alto livello della feature da implementare (Epica), si passa alla descrizione di un singolo Task descritto mediante un "linguaggio tecnico" di facile comprensione per il Dev team che dovrà completare l'attività. Un singolo Product Backlog è usato per descrivere tutto il lavoro in arrivo da svolgere sul prodotto, ma può capitare che più team lavorano sullo stesso prodotto. In questi casi viene utilizzato un attributo (Componente) del Product Backlog che raggruppa gli elementi in modo tale che ogni team possa orientarsi meglio sul lavoro che deve svolgere.

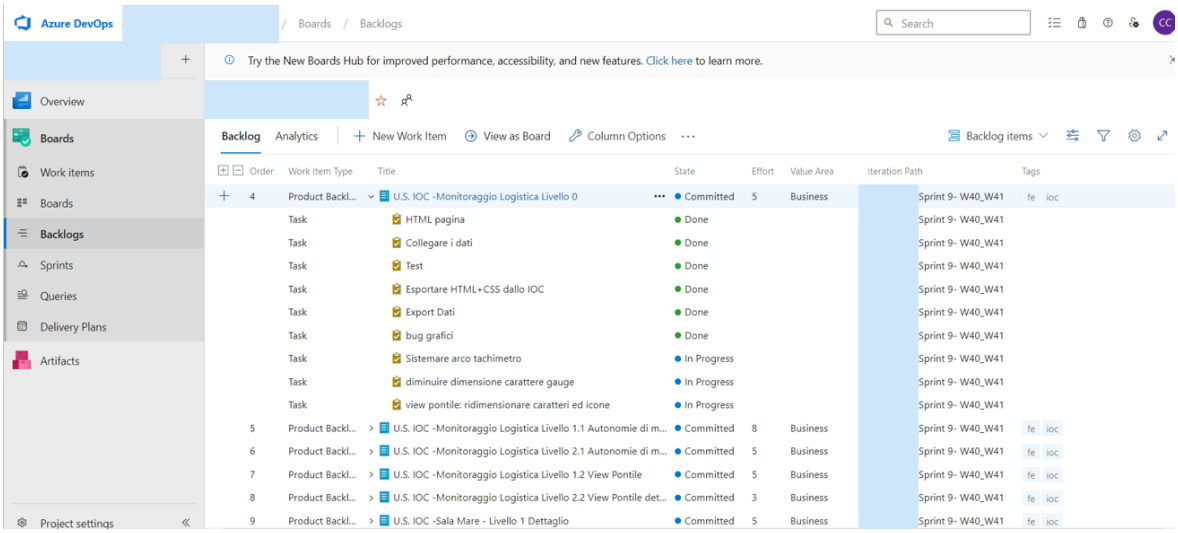

Figura 5. Scomposizione di una User Story in Task

Lo Sprint Backlog è l'insieme degli elementi del Product Backlog selezionati per lo Sprint, costituisce quindi parte del piano per realizzare lo Sprint Goal. Lo Sprint Backlog è una previsione fatta dal team di sviluppo riguardo a quali saranno le funzionalità presenti nell'output uscente dal successivo Sprint e il lavoro necessario per poter arrivare a questo. Lo Sprint Backlog rende visibile tutto il lavoro che il team di sviluppo identifica come necessario per raggiungere lo Sprint Goal. Lo Sprint Backlog è di proprietà del team di sviluppo, e tutte le stime incluse sono effettuate dal gruppo stesso. Spesso viene utilizzata una *Task Board* per visualizzare i cambiamenti di stato del Task nello Sprint corrente.

Gli stati che possono assumere i ticket relativi alle User Story e Task sono 4:

- "To Do": non ancora preso in carico da nessun membro del team;

- "Approved": assegnato ad un determinato membro del team.

- "Committed": al momento in lavorazione;

- "Done": l'attività è stata completata.

Lo Scrum Poker consente allo Scrum Team di stimare le varie User Story presenti all'interno del Backlog. Ogni membro del team fornisce una stima dell'*Effort* necessario per poter completare quella determinata User Story secondo la propria opinione ed esperienza, tenendo conto anche dei *Done Criteria*, ovvero dei punti che devono essere soddisfatti affinché la US possa essere considerata "Done", completata. La stima delle User Story è un'attività in cui tutto lo Scrum Team è coinvolto e viene svolta nella sessione di *Planning*

Poker a cavallo tra lo Sprint Review e lo Sprint Planning, nel momento in cui ci sono delle nuove User Story o delle modifiche fatte alle User Story già presenti in backlog. In entrambi i casi deve essere sempre essere presente il Product Owner, il quale deve assicurarsi che le stime siano idonee con le aspettative del cliente. Per dare una grandezza numerica al concetto di sforzo necessario per completare una User Story è buona prassi utilizzare la serie di Fibonacci come pesi di misura (Es: 1,2,3,5,8,13,21) per definire gli *Story Point*. Gli Story Point non danno un'indicazione del tempo necessario a chiudere una User Story, ma dello sforzo necessario o della difficoltà intrinseca del requisito. Due sviluppatori potrebbero impiegare diverso tempo per chiudere la stessa User Story, ma entrambi avranno la stessa percezioni di sforzo necessario su di essa. Una User Story facile deve essere considerata tale da tutti e per questo serve un metro di misura condiviso per definire cosa sia di livello minimo/facile (1), in modo da poterlo usare come peso per confrontare tutte le altre.

Solitamente una sessione di *Planning Poker* si svolge nel seguente modo**:**

- Ogni membro del team ha un mazzo di carte su cui sono riportati i numeri di Fibonacci
- Le User Story sono presentate singolarmente
- I membri del team "calano" coperta la carta corrispondente alla loro stima
- Se le stime sono molto diverse si discutono le differenze e il processo ricomincia

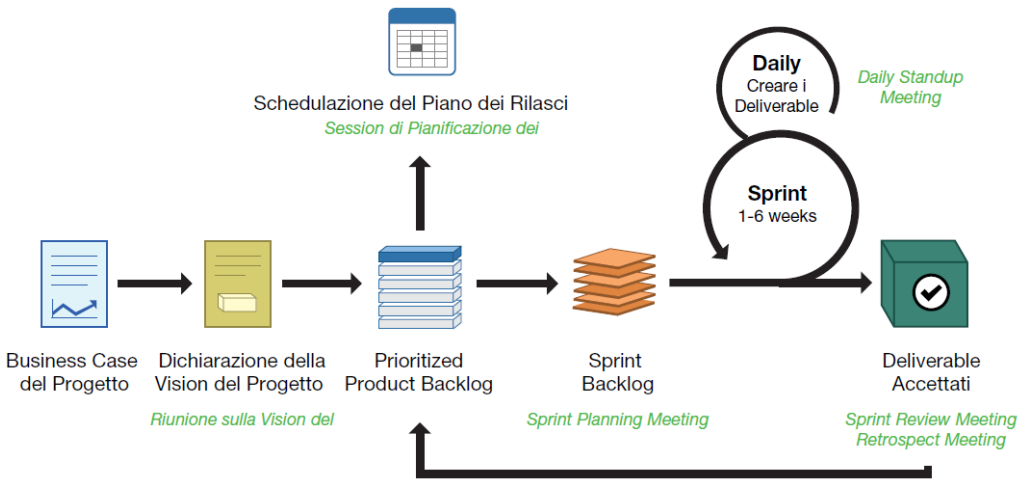

Figura 6. Eventi ed Artefatti dello Scrum

2. Il progetto di migrazione

Il progetto su cui verrà applicata la metodologia Agile con framework Scrum, tratta di un "*processo di migrazione*" di un software *MES (Manufacturing execution system)*, un sistema che attraverso delle dashboard permette di visualizzare varie informazioni relative ad un petrolchimico di una multinazionale del settore energetico.

2.1 Introduzione al MES

Il MES può essere definito come un sistema software completo e dinamico che consente di monitorare, tracciare, documentare e controllare il processo di produzione dei beni, dalle materie prime ai prodotti finiti. Ad oggi, esistono diversi fornitori di questo particolare software, come Ge Digital, Critical Manufacturing, Siemens Digital Industries, Rockwell Automation, Itac Software. Questi forniscono prodotti leggermente diversi tra loro ma con alla base una idea molto semplice: tengono sotto controllo tutte le fasi dei processi produttivi, garantiscono visibilità e di conseguenza una spinta alla produttività, migliorando la qualità dei prodotti e un migliore tracciamento. Tutto ciò è utile per semplificare le procedure per assicurare la conformità normativa e per migliorare l'assistenza post-vendita. Un altro aspetto da non sottovalutare è che l'adozione di un MES spinge a digitalizzare ulteriormente i processi, così da eliminare completamente il consumo di carta.

I principali vantaggi che una azienda può trarre dall'implementazione di un software MES sono:

- Riduzione degli sprechi, individuando in breve tempo eventuali problemi di produzione.
- Flessibilità della produzione nel caso di modifiche da apportare ai nuovi prodotti, in linea con le esigenze dei clienti.
- Aumento della capacità produttiva segnalando le aree in cui i processi produttivi possono essere affinati.

- Ottimizzazione dell'inventario, segnalando il momento opportuno in cui l'impresa necessità di approvvigionarsi di materie prime.
- Stato dei macchinari, aiutando la programmazione di manutenzioni periodiche.

2.2 Lo IOC

Il software MES trattato in questa sede prende il nome di *IOC,* un sistema che nello specifico tratta informazioni relative alla produzione di materiali plastici. Ne è un esempio ciò che viene emesso dai camini, i tipi di gas emessi nell'ambiente e i livelli di pericolosità di questi. Altre informazioni sono invece relative alla sicurezza dell'impianto, con degli alert che si accendono su una mappa dello stabilimento indicando in quale zona è necessario intervenire. Tutti questi dati vengono gestiti attraverso un modello integrato e digitale di gestione degli asset dello stabilimento, capace di integrare in modo semplice tutte le applicazioni verticali con cui raggiungere diversi risultati:

- Process Optimization via real-time data, ovvero i dati sono letti in maniera schedulata andandole a leggere ogni 5 minuti, fornendo informazioni in tempo reale.
- Riduzione dell'impatto ambientale.
- Riduzione dei tempi morti di produzione.
- Miglioramento della sicurezza sul lavoro.

La visualizzazione dei contenuti sullo IOC è determinata da una matrice di profili: l'utente sarà in grado di visualizzare contenuti a seconda del profilo assegnatogli.

I due principali componenti dello IOC sono la *Dashboard* e la vista *Emissioni e Flaring*

- La Dashboard permette all'utente di avere una panoramica generale delle funzionalità dello IOC, divise per sezione (*Card*). Ogni Card mostra una sintesi delle principali informazioni/KPI. Cliccando su di essa l'utente può raggiungere la vista di riferimento e analizzare nel dettaglio le icone presenti per costatare eventuali criticità e approfondirle cliccando sulle icone stesse. Le Card posso essere anche disattivate e quindi non consultabili.

- La vista di Emission e Flaring permette di monitorare l'andamento dei punti di emissione e di flaring. Essa si suddivide in quattro sezioni principali.

    1. Stato emissione camini**:** permette di monitorare lo stato dei camini e delle emissioni.
    2. Stato strumentazione SME**:** permette di monitorare la validità dei dati misurati ed i sistemi di backup e malfunzionamento in funzione.
    3. Focus Torce: permette di monitorare lo stato delle torce e gas inviati/etilene
    4. Stato recupero gas torce: indica il peso della composizione dei Gas e a portanza parziale.

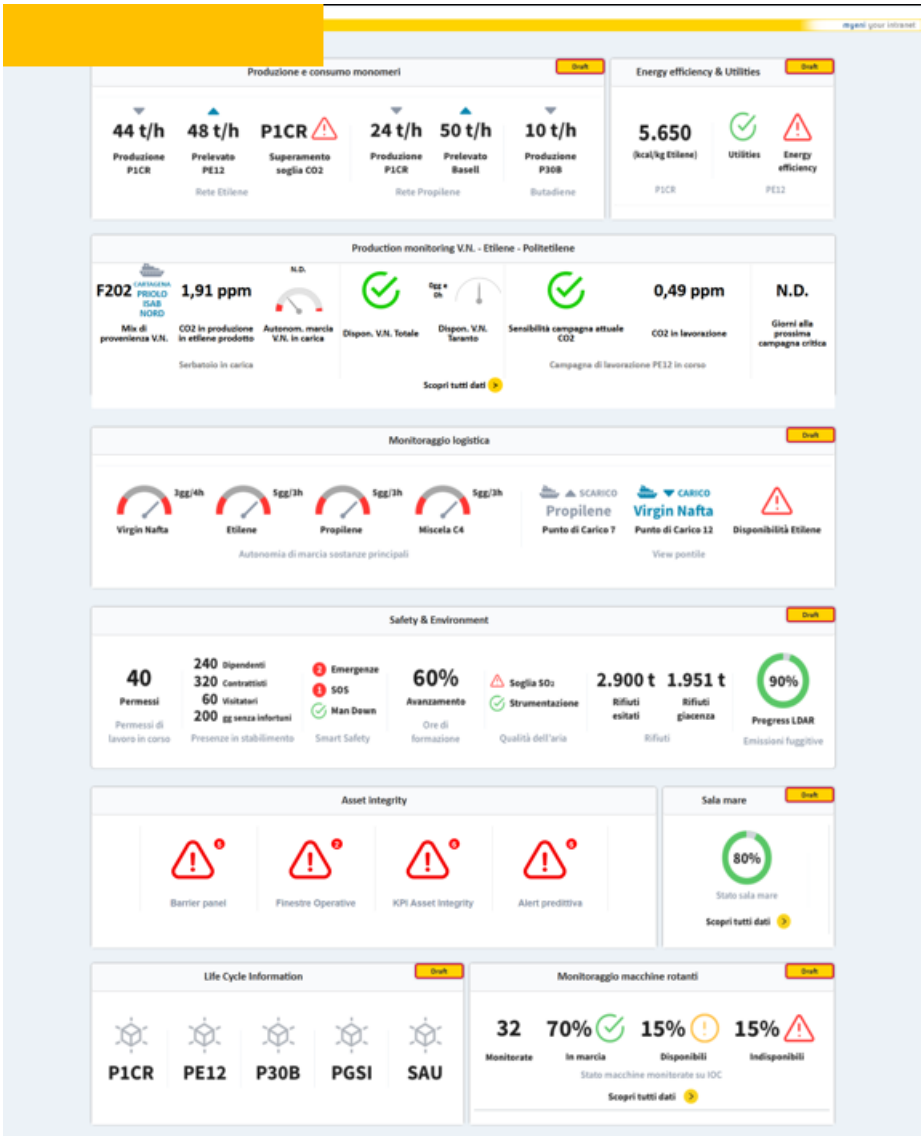

Figura 7. Dashboard dello IOC

2.3 Migrazione da Cloud View ad Apriso

Lo IOC è stato implementato da Aubay Italia, sempre per la medesima azienda cliente, su una piattaforma Cloud View. A distanza di alcuni anni, il cliente ha la necessità di migrare il proprio MES da Cloud ad *Apriso* per motivi prettamente strategici e gestionali.

Focalizzandoci in primis sulla soluzione in Cloud, l'implementazione di un MES su questa piattaforma viene vista tutt'oggi come una delle soluzioni migliori tra le aziende manufatturiere, in quanto è possibile avere:

"Semplicità di implementazione" del software, infatti le installazioni su piattaforma Cloud spesso vengono eseguite dalle aziende clienti in maniera autonoma, senza avere un grosso dispendio di tempo e risorse. Allo stesso tempo, questo genere di soluzione non necessita alcuna formazione specifica da parte dell'utilizzatore finale, per cui non è importante la formazione degli operatori. Lato IT, invece, significa implementare una soluzione con un'infrastruttura che è agile, snella, facile da controllare.

"Scalabilità" su una vasta gamma di dispositivi e macchine garantendo così dati uniformi da tutte le linee e processi, potendo raccogliere le informazioni da un'unica infrastruttura.

Facilità di "Manutenzione", intervenendo rapidamente per risolvere eventuali problemi e garantendo un software sempre funzionante

"Aggiornamenti frequenti" permettendo di aggiungere, anche dopo l'installazione iniziale, nuovi moduli e aggiornamenti che possono tornare utili al cliente.

D'altro canto, esistono degli svantaggi non indifferenti nei sistemi Cloud computing legati soprattutto alla perenne condivisione e gestione dell'intero universo dei dati in rete, determinando problematiche di sicurezza, e alla natura del provider che ne cura la fornitura che deve garantire nel tempo l'affidabilità dell'infrastruttura e la manutenzione necessaria.
Migrare il MES su Apriso risulta essere ad oggi una delle soluzioni più evolute, che integra con facilità i sistemi aziendali ed espande le proprietà MES a qualsiasi tipo di dispositivo. Un MES che tende a digitalizzare ogni fase della produzione**,** design dei prodotti inclusi e che non lascia nulla al di fuori delle logiche di controllo e di raccolta di dati, restituendo ogni informazione in viste e report di facile consultazione per garantirne una lettura

immediata e chiara. Inoltre, tale infrastruttura permetterebbe di ovviare ai problemi di privacy dei dati tipici delle piattaforme cloud, permettendo anche una più facile integrazione con gli altri sistemi aziendali esterni di terze parti come ERP (Enterprise Resource Planning), PLM (Product Lifecycle Management) e QMS (Quality Management Systems).

3. La filosofia applicata per la gestione del progetto

La decisione di gestire il progetto mediante il framework Scrum da parte di Aubay Italia nasce dall'esigenza di instaurare delle logiche di iterazione e comunicazione volte a massimizzare le potenzialità del team, oltre che a poter usufruire di strumenti utili al monitoraggio e al tracciamento dello stato avanzamento lavori. Nonostante le caratteristiche del progetto si discostino in parte da quelle che solitamente si trovano nel settore IT, presentando una lista di requisiti già ben definiti, l'efficacia del framework Scrum può portare alla risoluzione di bisogni legati alla gestione di una vasta quantità di attività che necessitano l'applicazione di un team coeso e orientato al medesimo risultato.

L'utilizzo del framework Scrum non costituisce una innovazione per Aubay Italia, poiché essendo una società operante nel campo dello sviluppo software da molti anni, non è nuova all'utilizzo delle metodologie agili per portare al termine progetti di questo genere. Innovativo risulta essere l'utilizzo di questo framework per un progetto di mera migrazione ed eseguito da un team che non ha mai lavorato in Agile. Si evince quindi che l'applicazione dello Scrum, oltre che per i motivi citati prima di gestione e monitoraggio, è utile anche per sperimentare gli effetti di questo su un progetto con Product Backlog già definito oltre che ad analizzare la risposta di un team giovane, che avrà la possibilità di interfacciarsi con nuovo approccio di lavoro che gli potrà tornare utile in futuro.

Per poter applicare nel modo corretto il framework Scrum, è necessario seguirne i principi nella loro totalità, avvalendosi di cerimonie e artefatti utili per il monitoraggio e la gestione del progetto. Ogni Scrum Team però, presenta caratteristiche ed esigenze diverse delineate dal contesto all'interno in cui si trova ad operare, portando ad una rivisitazione della metodologia descritta in letteratura. In questo capitolo vedremo gli strumenti e le modalità definite dallo Scrum Master per la gestione del progetto, introducendo anche caratteristiche e ruoli degli attori coinvolti.

3.1 Azure DevOps

Come accennato nei capitoli precedenti, il progetto di migrazione dello IOC verrà gestito mediante il framework Scrum descritto in precedenza. Trattandosi di un processo di migrazione, il software in sé è già presente e utilizzabile dall'azienda cliente. Questo risulta essere un grande vantaggio per il team che dovrà affrontare questo progetto, in quanto tutte le funzionalità e i requisiti del software sono già stati identificati nel passato e devono soltanto essere spostati su Apriso. Tutto ciò permette un'immediata identificazione delle User Story da dover svolgere durante gli Sprint, semplificando di non poco il lavoro del PO che non dovrà effettuare le attività preliminari di raccolta del requisito ma avrà solo il compito di modificare le US in caso di esigenze particolari da parte del cliente o per semplificare il lavoro al Dev Team, ad esempio, spacchettando e dettagliando più nello specifico le US per una migliore comprensione.

Tutte le Epiche con relative User Story e Task sono raccolte all'interno di *Azure DevOps*, un tool Microsoft utile per il reporting e la gestione di un progetto gestito sia in modalità Waterfall che in modalità Agile. I team di sviluppo che decidono di affidarsi ad Azure DevOps hanno la possibilità di inserire e modificare le sezioni che più gli interessano. Nello specifico è lo Scrum Master che, a seconda delle modalità decise per la gestione del progetto, aggiungerà e modificherà le sezioni del tool. Per il progetto oggetto di questa tesi troviamo tre macro-sezioni divise a sua volta da delle sezioni foglia. Tali sezioni sono:

"Overview": Contiene delle informazioni generali sulla natura e lo scopo del progetto, al suo interno troviamo:

- La *Summary*, dove troviamo una descrizione del progetto e informazioni relative ai membri dello Scrum Team, con relativi nomi e ruoli di ciascun membro.
- La sezione *Dashboard* dà la possibilità di consultare alcuni artefatti, come lo Sprint Burndown, che fornisce informazioni relative all'avanzamento dello sprint attualmente in corso confrontandolo con ciò che era stato pianificato in fase di Sprint Panning e lo Burnup per mettere a confronto i vari Sprint messi in atto fino ad oggi. Oltre a questi, è possibile trovare anche il Cumulative Flow Diagram per visualizzare il flusso di lavoro e identificare i colli di bottiglia nel processo di sviluppo del software, il Cycle Time che visualizza e analizza il tempo di ciclo del

gruppo utilizzando un grafico di controllo e il Lead Time utile per visualizzare e analizzare i tempi di consegna del team utilizzando sempre un grafico di controllo.

- *Il Wiki* è una sezione dove è presente della documentazione contenente delle informazioni utili al team per poter svolgere a meglio il lavoro. È possibile, infatti, trovare dei veri e propri manuali su come svolgere determinate attività necessarie per chiudere Task e User Story. Questi manuali sono scritti dai membri del Team con più esperienza e che hanno già lavorato all'implementazione dello IOC su Cloud View. Oltre ai manuali tecnici si possono consultare anche documenti sulla corretta implementazione del framework Scrum, per istruire il team sulle tecniche Agile che risultano essere nuove alla stragrande maggioranza dei membri del team.

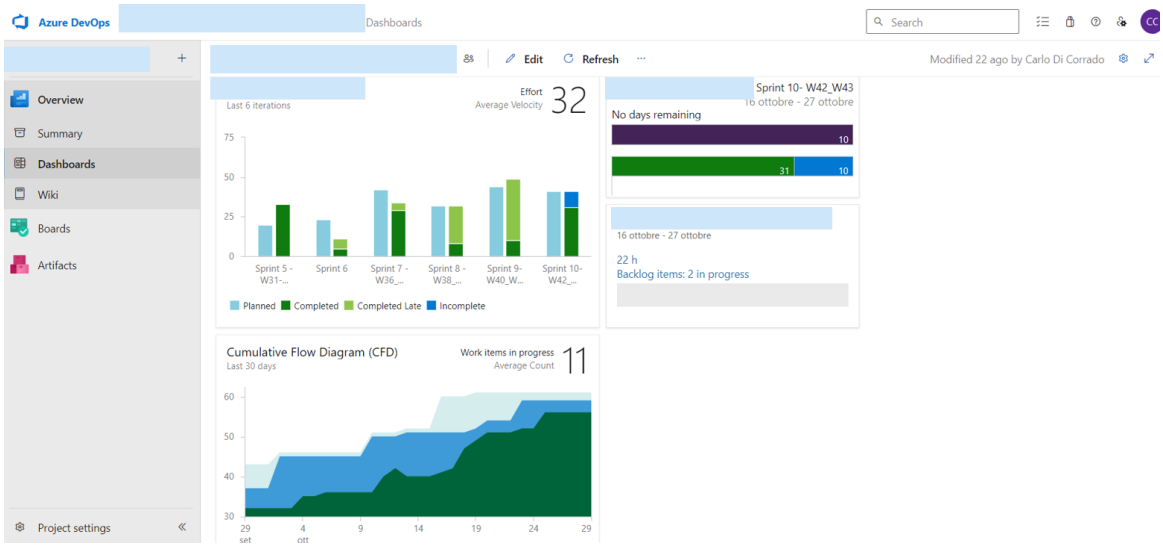

Figura 8. Dashboard su Azure Devops

"Boards": Contiene tutte le User Story inserite dal PO e quindi da implementare durante il progetto. In particolare, troveremo:

- La sezione *Work Items* presenta tutte le US tranne quelle allo stato "Done"; quindi, sono presenti solo quelle da completare. Si ha la possibilità di filtrare le varie US per stato e per assegnatario, oltre che a poter cambiare vista e visualizzare non più le User Story ma i Task che le compongono oppure le Epiche da cui derivano.
- Nella *Board* troviamo ancora una volta le US, ma questa volta divise per colonne a seconda dello stato in cui si trovano. La prima colonna è quella dei "New" fino ad

arrivare ai “Done”. Lo scopo di questa sezione è quella di fare una fotografia dello stato del progetto fornendo in tempo reale allo Scrum Team una comprensione immediata delle attività nuove, quelle attualmente in corso e quelle completate. Non a caso risulta essere una sezione molto utilizzata durante gli Sprint Review e gli Stand-up meeting.

- Il *Backlog* contiene in ordine di priorità, dall’alto verso il basso, le US che dovranno essere inserite all’interno di nuovi Sprint in fase di Sprint Planning.
- In *Sprints* visualizziamo le US provenienti dal Backlog e inseriti nello Sprint in corso con il rispettivo stato e assegnatario. Selezionando su Task Board si accede ad una pagina molto interessante che permette di vedere per ogni US all’interno dello Sprint lo stato dei rispettivi Task che le compongono, con una vista simile a quella della Board. Andando su Analytics, è possibile visualizzare il Burndown dello Sprint corrente, come quello presente nella sezione *Dashboard*. Infine, si ha la possibilità di visualizzare gli Sprint passati e analizzare il debito tecnico che è stato lasciato da ognuno e che è stato trasferito nello sprint successivo.

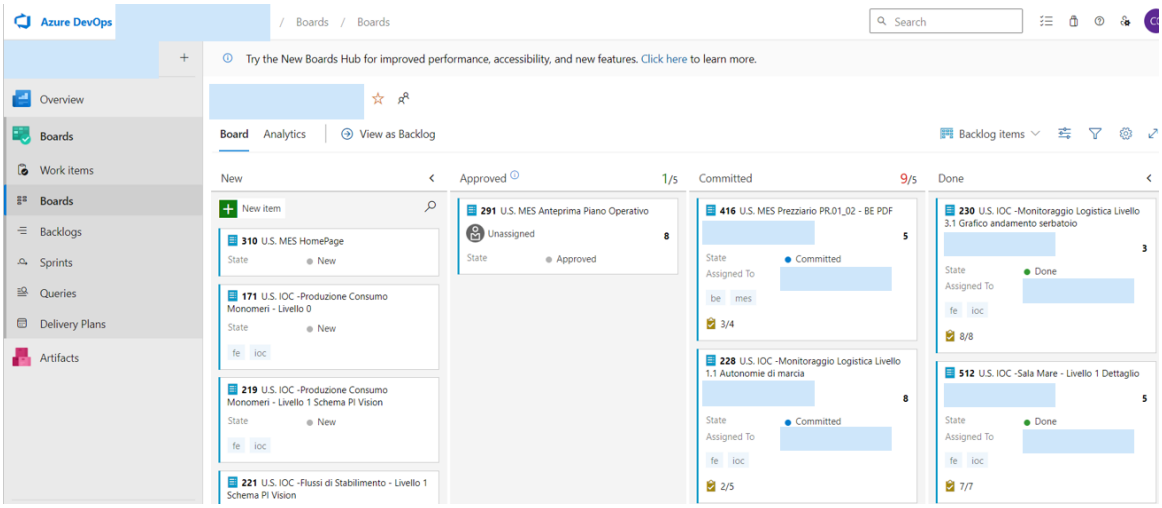

Figura 9. Boards su Azure Devops

3.2 Lo Scrum Team

Passiamo ora a descrivere le caratteristiche e l'organizzazione del team di sviluppo scelto per questo progetto di migrazione.

Come già accennato, il team di sviluppo è composto da membri alle prime armi con l'applicazione della metodologia Agile con framework Scrum, in quanto hanno sempre operato in contesti dove gli venivano assegnate delle attività già ben definite e che dovevano solo essere eseguite. Detto ciò, vedremo come nei capitoli successivi le prime riunioni saranno focalizzate sull'istruire il team a quelli che sono i principi e gli eventi fondamentali della metodologia Agile, effettuando delle lezioni teoriche accompagnate da casi di studio ed esercizi per apprendere al meglio questa nuova metodologia di lavoro. L'istruzione del team sarà facilitata dal fatto che tutti i membri sono dipendenti di Auaby Italia e alcuni hanno già lavorato tra di loro in passato.

Le figure *tecniche* del team (Developer e Tester) sono molto giovani ma con una notevole esperienza alle spalle. Spicca tra loro un membro con una seniority leggermente più alta rispetto agli altri, che chiameremo "*Tech A*": si preoccuperà di consigliare e monitorare la corretta esecuzione dei Task messi in pianificazione nello Sprint in corso. Rispetto a quanto detto in precedenza, quando abbiamo introdotto le caratteristiche di uno Scrum Team tradizionale, troviamo già delle contraddizioni; infatti, è stato detto che all'interno del team di sviluppo *non viene riconosciuto alcun titolo* ai membri, tutti vengono messi allo stesso livello per poter facilitare le interazioni tra loro ed instaurare un rapporto di fiducia essenziale per organizzare al meglio il lavoro all'interno del team. In questo caso al Tech A viene riconosciuto da tutti i partecipanti del progetto come con un punto di riferimento, dato anche dal fatto che in passato ha partecipato alla realizzazione della versione originale dello IOC su Cloud. Dunque, conosce alla perfezione il contesto e sa dove intervenire per effettuare modifiche.

Altra diversificazione dalla definizione teorica dello Scrum Team è la figura del *Product Owner*. Solitamente è una figura a stretto contatto con il cliente che rappresenta in tutto e per tutto le sue esigenze. In questo progetto il PO è una figura molto interna al team, che parteciperà assiduamente a tutte le cerimonie che verranno svolte. Questa decisione nasce

dalla partecipazione in prima persona del PO, insieme al Tech A, all'implementazione della versione iniziale dello IOC, maturando nel tempo una profonda conoscenza del tool e dei requisiti essenziali da soddisfare. Inoltre, è riuscito ad instaurare un rapporto di fiducia con il cliente, ciò potrà essere un punto a favore del team qualora sorgessero problematiche legate al mancato rilascio di alcuni pezzi di software nei tempi concordati o l'impossibilità di realizzazione di alcune modifiche per motivi tecnici. Trattandosi inoltre di un processo di migrazione, tutti i requisiti relativi alle modifiche e alle funzionalità che devono essere migrate sono già ben definite: il compito del PO sarà quindi quello di introdurre, da un punto di vista *funzionale,* le varie US. Questo per il team risulterà molto utile per poter capire il contesto in cui agiscono e soprattutto giustificare gli "Acceptance Criteria", che sono già stati definiti insieme a ciascuna feature, oltre a dare priorità alle varie attività da pianificare.

L'ultimo membro del team da attenzionare è lo *Scrum Master*. Oltre ad avere i compiti tipici di organizzazione, monitoraggio e semplificazione del lavoro, avrà anche il compito di istruire al meglio il team alle logiche del framework Scrum. Questa figura risulta essere un profondo conoscitore del mondo Agile e vanta una notevole esperienza nel campo di gestione di progetti IT. Ciò nonostante, anche per questo membro il progetto risulterà essere sfidante, in quanto non ha mai avuto a che fare con un team alle prime armi con il metodo Agile. Sarà fondamentale, all'inizio del progetto, effettuare una precisa ed efficace comunicazione delle nozioni dello Scrum per poter partire nel miglior modo possibile. Una cattiva comunicazione porterebbe a uno dei rischi principali di questo progetto, ovvero la mancanza di un gruppo perfettamente allineato sulle tecniche che si stanno mettendo in atto, causando confusione, incertezza e ritardi. D'altro canto, essendo un gruppo di ragazzi inesperti del mondo Agile, lo Scrum master ha già messo in preventivo eventuali difficoltà di apprendimento a inizio progetto che potrebbero ritardare l'inizio dei lavori.

3.3 Definizione di cerimonie e artefatti

Infine, per una corretta introduzione al processo di migrazione analizziamo le scelte dello Scrum Master riguardo la *governance*. Nell'Agile, prima che il progetto inizi in modo definitivo, è essenziale prendere decisioni riguardo lo scopo e la durata delle varie tipologie di cerimonie e degli artefatti utilizzati. Tali decisioni, intaccheranno la qualità e le

tempistiche del progetto. Sarà necessario anche un parere diretto del PO che, rappresentando la voce del cliente, dovrà assicurarsi che il tutto sia in linea con le sue aspettative.

Notiamo ora come lo Scrum Master ha deciso di gestire ciascuna cerimonia

- *Scrum Poker:* Avrà la durata di un'ora ma il tempo può variare a seconda della quantità di US e sarà svolto successivamente alla conclusione della Sprint Review. Ogni membro del Dev Team sarà chiamato a collegarsi all'interno di una "stanza virtuale" (Planning Poker) ad esporre il proprio giudizio riguardo alle User story che dovranno essere stimate. In particolare, ogni giudizio dovrà esprimere un voto secondo la scala di Fibonacci, tale voto sarà per ogni membro del team "l'effort necessario" per concludere l'attività. I giudizi relativi a ciascuna US sono frutto di una analisi effettuata dal singolo tecnico secondo quella che è la sua conoscenza della materia e la sua esperienza, ma per dare un parametro oggettivo da prendere come riferimento, lo Scrum Master, ascoltando i pareri del team, ha definito qual è il *valore di uno story point.* In particolare, "*la creazione delle Operations: view, layout, screen è considerata dal team come attività facile, che vale uno story point*". Vedremo con l'avanzare del progetto come anche questo riferimento cambierà, poiché il team avrà acquisito con l'esperienza delle nuove conoscenze che gli faranno cambiare la percezione di "attività semplice".
  Tutte le US da stimare verranno decise ad ogni sessione dal PO e avrà un tempo massimo di cinque minuti per introdurle. Il Dev team voterà contemporaneamente la US senza conoscere il giudizio degli altri e avrà una durata massima di due minuti. Concluso il primo giro di votazioni lo Scrum Master guiderà la discussione per far emergere le motivazioni di ciascun componente che lo hanno portato ad esprimere quel voto, questo momento di confronto può avere una durata variabile (solitamente non più di 10 minuti). È importante in questa fase che il team si confronti al meglio e arrivi ad un punto di incontro. Successivamente, inizia un secondo giro di votazione che sancisce l'effort definitivo della US. In caso di pareri divergenti prevarrà la stima che ha ricevuto più voti. Finito lo Scrum Poker tutte le User Story stimate verranno inserite nel product backlog con lo stato "To Do" in attesa di essere aggiunte in pianificazione e il valore della stima sarà inserito nel ticket nella sezione Story Point.

- *Sprint Planning:* Verrà svolto ogni lunedì mattina, successivamente alla conclusione della Sprint review inerente allo Sprint appena concluso, qualora ce ne fosse uno. Lo Scrum master mostrerà l'elenco delle US presenti nel Product Backlog e, con l'ausilio del PO che indicherà le attività con priorità maggiore, inserirà le User Stoy nello Sprint in partenza. Ogni User Story indicata dal PO come potenziale attività da inserire nel nuovo Sprint verrà analizzata nuovamente insieme al team, in questa fase ogni membro dovrà dare il suo consenso all'inserimento della US e in caso di dissenso esporre le proprie motivazioni. L'ultima parola per la messa in pianificazione spetta allo Scrum Master, che dopo aver ascoltato i pareri di ciascun tecnico, può decidere se inserire o no l'attività nello Sprint. Solitamente un tecnico è contrario all'aggiunta di una US poiché allarmato della quantità di effort necessaria per il completamente di questa; infatti, tutte le attività inserite nello Sprint devono essere completate entro la scadenza definita per evitare di ritrovarsi con del debito tecnico da dover poi inserire nello Sprint successivo. Di conseguenza le US che hanno ricevuto una stima elevata durante il Planning Poker saranno quelle da attenzionare maggiormente durante l'esecuzione dello Sprint poiché sono quelle che richiederanno più tempo. Ovviamente è nell'interesse di tutto il team avere Sprint con una quantità di effort idonea alle loro capacità. Dopo aver inserito le attività in pianificazione, partirà un secondo momento di confronto tra le sole figure tecniche del team, necessario per stabilire un *Assegnatario* a ciascun Task inserito nel nuovo Sprint. L'assegnatario sarà il "responsabile della corretta esecuzione dell'attività". Questo non implica necessariamente che sarà l'unico membro del team a lavorarci, ma sarà sicuramente il principale attore coinvolto. Avendo definito un assegnatario per ogni Task appartenente ad una determinata User Story messa in pianificazione, lo stato dei ticket cambierà da "To Do" ad "Approved" e lo Scrum Master darà ufficialmente inizio allo Sprint. Il team avrà a disposizione 15 giorni per terminare il lavoro nelle modalità previste.

- *Sprint Review:* Verrà svolto ogni lunedì mattina alla conclusione di ogni Sprint e avrà la durata massima di 30min. Lo Scrum Master, mediante sempre l'utilizzo di Azure Devops, elencherà le US da completare nello Sprint appena concluso. Ogni US verrà analizzata una ad una e verrà chiesto ad ogni assegnatario se l'attività è stata conclusa, e quindi può essere chiusa, oppure qual è la "percentuale di

completamento" della US qualora non fosse completata. Tutte le User Story non completate costituiranno il debito tecnico, che dovrà essere inserito nuovamente in pianificazione. L'effort del debito tecnico di uno sprint viene stimato proprio in base alla percentuale di completamento di ciascuna User Story incompleta, determinata dalla quantità di Task non chiusi, per questo è fondamentale che ciascun tecnico arrivi alla Sprint review con un valore percentuale che rispecchi esattamente l'ammontare di lavoro necessario per completare i Task rimanenti per poter chiudere la User Story. A fronte di una US non portata al termine, lo Scrum Master chiederà all'assegnatario di esporre i motivi per il mancato completamento, in modo tale che gli stessi problemi non emergano negli Sprint successivi. Per quello che riguarda le US definite dall'assegnatario come completate, i rispettivi ticket subiranno un cambiamento di stato da "Committed" a "Done", previo accertamento da parte del PO che ciascun Acceptance Criteria sia rispettato a pieno. Solo dopo questo controllo lo Scrum Master considererà ufficialmente conclusa la US.

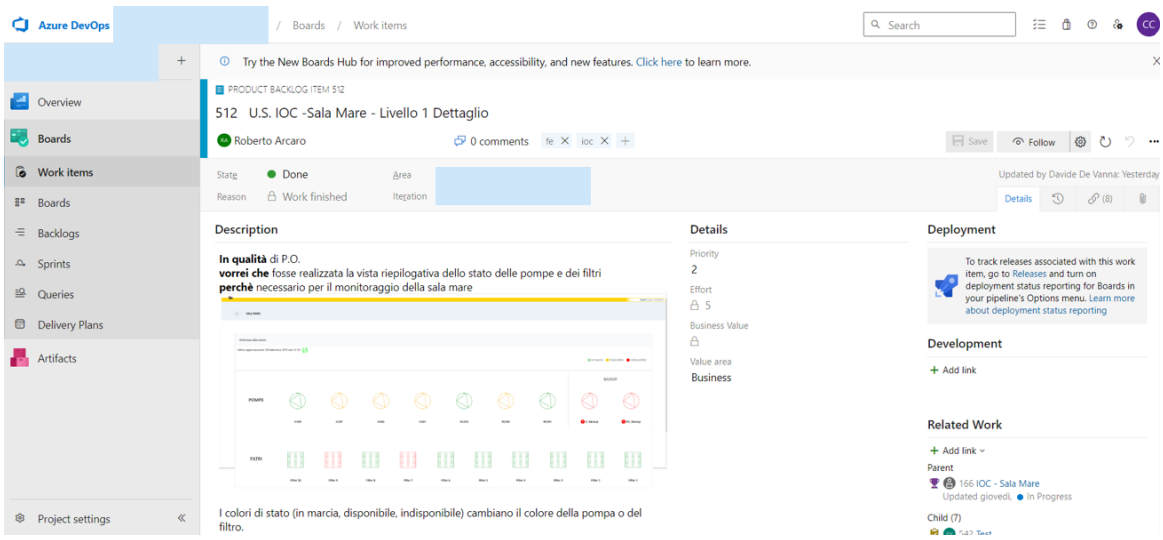

Figura 10. Ticket in Done

- *Sprint Retrospective:* Si svolgerà necessariamente dopo ogni Sprint Review e può avere una durata di 1h e 30min. Lo Scrum Master presenterà al team una lavagna divisa in cinque sezioni, dove ogni sezione contiene una domanda inerente allo Sprint appena concluso. Ogni membro del Dev Team avrà la possibilità di rispondere e di lasciare la propria opinione riguardo i punti di forza e di debolezza

inerenti soprattutto all'organizzazione del lavoro e alle difficoltà tecniche di completamento dei Task.

Le sezioni in questione sono:

- *What slowed us down?* Quali sono stati gli impedimenti che non hanno permesso al team di completare le attività inserite nello Sprint.
- *What powered us up?* Quali sono i fattori che hanno facilitato il lavoro durante lo Sprint
- *What are we worried about?* Cos'è che genera maggiore preoccupazione durante lo svolgimento di uno Sprint
- *I wish I had*. Quali modifiche il team vorrebbe vedere nel nuovo team
- *Actions*. Le azioni che verranno intraprese per migliorare la qualità del lavoro.

Ogni sezione verrà analizzata singolarmente e ciascuna risposta data dovrà essere giustificata da ciascun membro. Noteremo nei capitoli successivi come uno dei punti che susciterà preoccupazione all'interno del team è relativa all'applicazione delle metodologie Agile.

- *Daily Meeting:* Tutto il team dovrà obbligatoriamente collegarsi ogni giorno lavorativo ad un incontro schedulato per le 9:30. L'obiettivo è quello di condividere quali attività verranno portate avanti oggi e cosa è stato concluso il giorno precedente. Tutto ciò permetterà al team di rimanere costantemente aggiornati sul lavoro degli altri membri facilitando la collaborazione e lo scambio di informazioni.

4. Lo svolgimento

Dopo una lunga pre-analisi svolta per spiegare il contesto e le modalità con cui Aubay Italia ha deciso di intraprendere questo progetto e iniziare il processo di migrazione dello IOC, ci concentriamo ora ad analizzare quali sono stati gli elementi di criticità dovuti all'applicazione della metodologia Agile. Sono state analizzate tutte le cerimonie avvenute dal 30/05/2023 fino al 17/10/2023, osservando quali sono stati gli effetti dell'applicazione del metodo Scrum sull' esecuzione del progetto e come questi abbiano portato benefici o rallentamenti.

L'analisi verrà fatta su tre macro-temi collegati direttamente all'applicazione del framework Scrum e che si sono rilevati dei punti da attenzionare dagli attori direttamente coinvolti per apportare eventuali migliorie. Noteremo infatti come questi punti di analisi subiranno una evoluzione durante il periodo di svolgimento del progetto, grazie soprattutto alla maggiore consapevolezza che avrà acquisito il team sui temi inerenti allo Scrum e all'approfondimento del contesto tecnico-esecutivo del progetto. Prima di partire è essenziale introdurre quelli che sono stati i tre Sprint che si sono rilevati cruciali e che hanno costituito un punto di svolta nelle dinamiche interne del team.

4.1 Gli Sprint salienti

Osservando l'intera evoluzione del progetto, è possibile affermare che il team abbia risposto in modo più che positivo all'applicazione dello Scrum, percependo alcuni dei vantaggi derivanti da questa metodologia, come avere dei Task definiti e dettagliati e degli obiettivi sia giornalieri (definiti durante i Daily Meeting) che settimanali. Inoltre, la possibilità di accedere in qualsiasi momento su Azure Devops per poter analizzare i documenti presenti sul wiki e le varie User Story che descrivono i requisiti attraverso immagini, file html, query da utilizzare ecc.… facilita il lavoro e aumenta la produttività.

I membri del team sono abituati a lavorare su progetti gestiti attraverso la metodologia tradizionale, dove erano sempre presenti degli obiettivi periodici da centrare: ora sono accompagnati da una data da rispettare e da degli Acceptance Criteria da soddisfare. Nella

parte iniziale del progetto questi due elementi provocarono un effetto di oppressione sul team, generando apprensione di non completare le attività con la qualità e nelle tempistiche previste, dando vita così ai risultati deludenti dei primi tre Sprint.

Altra peculiarità molto apprezzata dal team è legata all'attribuzione di tutti i successi e gli insuccessi maturati durante lo svolgimento del progetto, poiché questi vengono divisi in parte uguale tra tutti i membri senza accusare in maniera diretta un componente piuttosto che un altro per eventuali errori e ritardi durante uno Sprint. La equa distribuzione delle responsabilità sugli insuccessi del progetto dà la possibilità ai membri meno esperti di poter lavorare ai vari Task in modo più sereno, sapendo che in caso di necessità ci sarà sempre un collega pronto a dare una mano. La forte coesione del Team è un effetto diretto di uno dei principi del framework Scrum basato su un team totalmente "*orientato al risultato"* di completamento di tutte le US presenti in Backlog, superando tutti gli imprevisti e Change Request che possono sorgere durante un progetto IT. Per raggiungere questo obiettivo è necessaria la partecipazione attiva di tutte le risorse a disposizione: è quindi necessario non lasciare nessuno indietro, bisogna essere sempre disposti a dare una mano ai membri meno esperti, soprattutto nelle fasi iniziali del progetto.

Gli Sprint Retrospective si sono rivelati molto utili per monitorare l'apprendimento del team sulle logiche Scrum. Sono state ascoltate le opinioni di tutti coloro che hanno partecipato allo Sprint in analisi e una delle prerogative dello Scrum Master è stata quella di far emergere durante la discussione le motivazioni che hanno rallentato lo svolgimento dello Sprint, per poi intervenire e cercare di migliorare le performance dell'intero team e non quelle del singolo individuo, poiché all'interno di uno Scrum team, cercare di velocizzare il singolo, da benefici molto minori rispetto che a velocizzare il team in ordine di circa 10 a 100.

Qui di seguito, tramite delle tabelle, vengono riportati i risultati delle sessioni di Sprint Retrospective dopo le sessioni relative agli Sprint 3, 4 e 7. Le tabelle, con relativi risultati, sono state successivamente inserite nella sezione wiki di Azure Devops, per tenere traccia degli avanzamenti fatti durante lo svolgimento del progetto. All'interno di questi incontri sono state esposte perplessità anche riguardo la componente tecnica del progetto, ma la nostra analisi sarà incentrata principalmente sulle difficoltà organizzative riscontrate dal team applicando il framework Scrum.

4.1.1 Sprint 3

La prima sessione di Sprint Retrospective è stata svolta proprio alla conclusione dello Sprint 3, infatti i primi due Sprint sono stati utilizzati soprattutto per far ambientare il team alle logiche Scrum, senza registrare grossi avanzamenti. In questo Sprint il team non è riuscito a chiudere gran parte delle US messe in pianificazioni, mostrando difficoltà di adattamento al nuovo metodo di lavoro e comprensione del contesto tecnico-informatico in cui si trovano ad operare.

What slowed us down?	*What powered us up?*	*What are we worried about?*	*I wish I had…*	*Actions*
1. Il passaggio di consegne è un onere	1.Avere obiettivi definiti aiuta. I task sono chiari e anche le scadenze sono chiare. Aiuta anche a darsi delle stime e le proprie capacità. 2.La chiarezza delle user storie 3.Disponibilità del materiale dello IOC (html, css, js) visibile sul Wiki 4.Avere più risorse nel team	1.Sottostima dei task (non essere in grado di stimare correttamente	1.Maggiori informazioni tecniche sulla User Story	1.Prima di iniziare a lavorare una US, analizzare e ragionare sull'obiettivo

Tabella 1. Retrospective Sprint 3

Durante la retrospettiva lo Scrum Master ha cercato di far emergere le difficoltà riscontrate dal team chiedendo ai diretti interessati. L'unico elemento di rallentamento è stato correlato al passaggio di consegne, ovvero al fatto che il Tech A, forte della sua esperienza di implementazione della vecchia versione dello IOC, doveva trasferire la sua conoscenza al team su come agire per svolgere determinate feature. Senza dubbio i passaggi di consegna hanno molto rallentato lo svolgimento dello Sprint, dato che gran parte delle US sono state affidate al Tech A stesso, ma sono stati fondamentali per mettere nelle condizioni di lavorare gli altri membri del team. Curioso il fatto che non siano stati costatati alcun tipo di problema relativo alle logiche Agile, diversamente da quanto individuato dallo Scrum Master e dal PO stesso, i quali dall'alto hanno monitorato l'avanzamento del lavoro giorno dopo giorno, rilevando comportamenti non idonei con le tecniche agili.

Guardando la seconda colonna del *What powered us?* sembra che il team sia contento di lavorare in Scrum, poiché percepiscono il lavoro "semplificato" rispetto a come erano

abituati con il metodo Waterfall. Peccato che, senza la giusta attenzione agli strumenti Agile, gli strumenti di semplificazione messi a disposizione al team non vengono sfruttati al meglio. Ad esempio, la possibilità di avere US chiare e ben definite aiuta sicuramente la comprensione dal punto di vista funzionale e di conseguenza dà un primo input su come agire sul codice per completarla. Non bisogna però dimenticare che la soluzione tecnica pensata deve essere in linea con le tempistiche di chiusura dello Sprint e deve poter centrare gli Acceptance Criteria decisi dal PO. Non a caso, tra le azioni da intraprendere per lo Sprint successivo, lo Scrum Master ha sottolineato l'impegno del team a sforzarsi a comprendere meglio la User Story e gli Acceptance Criteria a essa collegata.

L'unico punto che crea preoccupazione nel team è il processo di stima, per definire l'effort necessario per completare le User Story. In particolare, il team non ha abbastanza conoscenza del contesto per fornire una stima in base all'effort necessario per il completamento. Il tema della difficoltà nelle stime sarà approfondito nel paragrafo 3.1, in quanto individuato come uno dei temi più rilevanti del progetto.

La necessità da parte dei membri del gruppo di avere ulteriori informazioni sulla componente tecnica del progetto è sintomo della difficoltà avuta nei passaggi di consegna, che oltre ad aver rallentato di molto il lavoro è risultato essere anche inefficace. Il principale motivo secondo lo Scrum Master è dovuto ai problemi di comunicazione e di interazione tra il Tech A e i componenti meno esperti, in particolare viene accusato il Dev team di non fare abbastanza domande al Tech A su cosa bisogna applicare per completare le User Story. Si dà per scontato che non sia necessaria un'approfondita conoscenza della materia, in quanto, in caso di problemi sulla programmazione o su eventuali bug, il Tech A sia sempre a disposizione per dare una mano e a risolvere i problemi nel minore tempo possibile. La collaborazione e l'autorganizzazione del lavoro all'interno dello Scrum Team è fondamentale nei progetti Agile, infatti verrà approfondita nei paragrafi successivi, ma questo non implica che ogni membro del team debba essere comunque "indipendente" e lavorare in maniera autonoma ai vari Task che gli vengono assegnati e solo in caso di necessità scomodare un collega per chiedere supporto.

4.1.2 Sprint 4

Lo Sprint 4 ha sancito uno spartiacque per il progetto, in quanto i risultati provenienti dalla Sprint Review hanno evidenziato ancora una volta ritardi nel completamento e nell'esecuzione dello Sprint. Dopo circa due mesi dall'inizio del progetto, il team non è ancora riuscito ad assimilare correttamente le logiche dello Scrum, mostrando ancora una volta degli atteggiamenti controproducenti per la corretta applicazione. Diversamente però da quanto emerso dallo Sprint precedente il team è riuscito ad individuare delle problematiche direttamente collegate ai principi organizzativi di uno Scrum team tradizionale. Questo risulta essere un grosso passo avanti nel percorso di apprendimento di questa nuova metodologia di lavoro in quanto la capacità di poter analizzare con senso critico, cosa non ha funzionato nello Sprint implica che il team sta cominciando ad assimilare ed applicare le nozioni fornitegli dallo Scrum Master, cominciando ad analizzare quello che funziona e quello che bisogna cambiare.

What slowed us down?	*What powered us up?*	*What are we worried about?*	*I wish I had…*	*Actions*
1.Dover gestire parallelamente sia task di test che task di sviluppo 2.Riaprire dei task chiusi precedentemente 3.In alcuni casi le US non sempre molto chiare. Talvolta anche lo sviluppatore non si sforza abbastanza nel leggere e comprendere la US. 4.Poca attenzione durante le fasi di stima e analisi delle US. I requisiti non sono stati compresi nel modo corretto e ha comportato un dover tornarci durante lo sprint. Durante la pianificazione non sono emersi dubbi che sono nati poi solo durante gli sviluppi ma potevano essere anticipati	1. Poter riciclare sw già sviluppato in altri US 2.La granularità dei task in cui viene scomposta la US aiuta ad ottimizzare le risorse e il lavoro. Con l'esperienza si sta imparando a scomporre meglio le US 3.Stimare in ore i task 4.Influenzare poco il gruppo, ad esempio, durante le fasi di stima	1. L'assenza del Tech A 2.Prendere in carico task complessi, fuori dalle nostre capacità, magari senza il supporto dei colleghi	1.Cercare maggiore autonomia e meno dipendenza dal Tech A (limitando le richieste di supporto)	1.Essere più indipendenti dal Tech A, per farlo è necessario migliorare la coordinazione del lavoro tra tutti i membri del team.

Tabella 2. Retrospective Sprint 4

La prima motivazione che ha portato ad un rallentamento dello Sprint si riferisce al tempo impiegato da ciascun sviluppatore per testare le varie modifiche che vengono fatte. Infatti,

una volta che un Task viene completato, prima di passare a quello successivo è buona prassi fare dei test per verificarne il corretto funzionamento. Per poter ovviare a questo problema lo Scrum master ha deciso di nominare un membro del team come *Tester ufficiale,* il suo compito sarà quello di testare sul software stesso tutte le modifiche fatte man mano che gli altri membri del team notificano il completamento di una determinata attività. L'obiettivo è quello di permettere a ciascun componente di continuare a lavorare sui Task che gli sono stati assegnati per cercare di completarli tutti entro la fine dello Sprint. Eventuali bug o malfunzionamenti riscontrati dal Tester verranno notificati principalmente durante i Daily Meeting per poter intervenire direttamente durante lo Sprint in corso. In assenza del Tester, il team molte volte effettuava dei test abbastanza sbrigativi senza entrare nel dettaglio della funzione, questo per poter dedicare la maggior parte del tempo ai Task rimanenti. Ovviamente l'ultima parola per la chiusura definitiva di un Task e della relativa US spetta al Product Owner, che dovrà verificare che tutti gli Acceptance Criteria siano rispettati. Sommando la scarsa attenzione sui test al poco approfondimento delle feature messe in pianificazione il risultato è un certo ammontare di User Story rigettate dal PO con relativi Task riaperti che costituiscono un debito tecnico da reinserire nello Sprint successivo, con conseguenti ritardi sul progetto. Tra le azioni individuate nello Sprint precedente da mettere in atto per migliorare la qualità del lavoro, era stata individuata la necessità di approfondire le US per migliorarne la valutazione dell'effort necessario e la qualità del lavoro svolto. Tale azione, come è possibile vedere in tabella, non è stata intrapresa dal Dev Team che ha ammesso di aver peccato durante le sessioni di analisi per non aver approfondito abbastanza tutti i temi, e di aver iniziato lo Sprint con molti dubbi tecnici e funzionali ancora da chiarire.

Gli aspetti positivi che hanno velocizzato questo Sprint sono per lo più collegati a novità introdotte in fase di Sprint Planning dallo Scrum Master. Da sottolineare la scomposizione delle User Story in Task più granulari per facilitarne la comprensione, ma anche la scomposizione di una singola US in più User story per facilitarne la stima in fase di Planning Poker. Altra modifica molto apprezzata è stata la possibilità di poter definire una stima dei Task in ore. Sostanzialmente è stata data la possibilità di poter fornire una stima dei Task in base al tempo necessario per il completamento. Questa nuovo processo di stima è stata richiesta direttamente dal team, poiché dovrebbe facilitare il processo di scelta del numero adatto di attività da mettere in pianificazione, aumentando la probabilità di poter completare tutti i Task all'interno di uno Sprint. Ultimo elemento positivo è la possibilità

di poter riciclare del software proveniente da vecchie User Story concluse. Il motivo è dato dal fatto che all'interno della maggior parte dei progetti IT alcune User story sono correlate tra loro e sono quindi molto simili da un punto di vista tecnico, ciò significa che eventuali soluzioni possono essere applicate più di una volta, risparmiando così tempo prezioso.

Ora che la preoccupazione del team di sottostimare i Task è stata in parte risolta, sembra che il maggior elemento di preoccupazione sia la probabile assenza del Tech A, in quanto nello sprint successivo potrebbe essere impegnato su altri progetti. Una sua mancanza potrebbe rallentare di non poco l'avanzamento. Il team, non si sente ancora pronto ad affrontare le varie cerimonie senza il suo supporto poiché la sua esperienza risulta essere ancora essenziale per indirizzare al meglio il team nelle attività più complicate.

Ciò nonostante, i membri meno esperti percepiscono di non poter dipendere ancora per molto dal Tech A, hanno voglia di essere più autonomi, limitando le richieste di supporto e cominciando ad affrontare Task sempre più complessi. Il Tech A è sicuramente il componente con più carico di lavoro durante gli Sprint, se si riuscisse a diminuire il numero di richieste di aiuto giorno dopo giorno la produttività avrebbe sicuramente dei miglioramenti non indifferenti. Non a caso lo Scrum Master individua come prossima azione quella di essere sempre meno dipendenti dal consiglio del Tech A. Per farlo è essenziale che le figure meno esperte comincino a collaborare tra loro e comincino ad "auto-organizzarsi" al loro interno per ottimizzare i tempi. L'obiettivo è quello di ritagliarsi all'interno di uno Sprint sia il tempo necessario per completare i Task assegnati direttamente dallo Scrum master sia per essere disponibili ad aiutare i colleghi che presentano delle difficoltà.

4.1.3 Sprint 7

L'ultimo Sprint in analisi è il numero sette. Lo Sprint dove il team ha ottenuto degli ottimi risultati sia in termini di User story portati al termine sia in base alla qualità dell'organizzazione del lavoro. Il team in questo Sprint sembra essersi adattato a pieno alle caratteristiche dell'Agile, sviluppando un senso di collaborazione e unione che lo Scrum considera essenziale per poter raggiungere l'obiettivo.

What slowed us down?	*What powered us up?*	*What are we worried about?*	*I wish I had…*	*Actions*
1.Avere parte del team a disposizione ha limitato la collaborazione tra gli elementi del team 2.Debito tecnico dallo sprint precedente	1.Riciclo del codice IOC 2.Lezione del Tech A su JSON 3.Esperienza sulle US precedente hanno aiutato nella stima 4.Collaborazione e confronto verso la chiusura dello sprint, ha aiutato nella chiusura delle US	1.Poco tempo per supportare i colleghi	1.Maggiore analisi nella valutazione del requisito	1.Dedicare il tempo necessario per analizzare e studiare la struttura dati ed indirizzare il lavoro ed i dubbi nel giusto modo. 2.Anticipare dubbi e domande. 3.Cominciare ad assegnarsi autonomamente i task durante gli Sprint Planning

Tabella 3. Retrospective Sprint 7

In questo Sprint il team non è stato al completo, alcuni dei membri meno esperti non hanno potuto contribuire al lavoro programmato. Ora che il team è più coeso, la presenza del Tech A è importante ma non più essenziale. Si sono sviluppate delle dinamiche interattive tra i membri del team associabili a delle vere e proprie "routine organizzative". Ogni membro si sta specializzando in qualcosa e sa ancor prima dello Sprint planning che tipo di attività andrà a svolgere. Queste routine facilitano la comunicazione e l'ottimizzazione dei tempi, ogni membro sa a cosa lavorano i suoi colleghi in ogni momento della giornata, sa quindi a chi chiedere in caso di necessità nel momento in cui si presentano dei problemi non direttamente riconducibile alla tipologia di attività che ha approfondito, evitando di perdere tempo ad aspettare che qualcuno si faccia vivo per risolvere il problema. Tali routine possono essere facilmente spezzate una volta che una risorsa viene a mancare, per questo il team ha individuato la mancanza di alcuni componenti come uno dei motivi di rallentamento.

La maggiore consapevolezza del team è visibile anche analizzando i punti segnalati come positivi durante lo Sprint. Il team ha percepito il benessere derivante dalla collaborazione e dal confronto e vorrebbe dedicare più tempo ad aiutare il prossimo. Adesso tutte le imposizioni dello Scrum come la durata non modificabile dello Sprint, la quantità di Story Point, la necessità di approfondire le US per dare una stima coerente e la responsabilità di dover portare correttamente a termine i Task assegnati, non sono più visti come ostacoli ma come dei riferimenti molto utili per orientare il team al risultato. Ovviamente questi

miglioramenti non sono riconducibili esclusivamente ad un cambio di mentalità da parte del team che ha deciso di sposare del tutto le logiche Scrum, ma sono dovute anche dall'esperienza e dalla continua iterazione del progetto. Infatti, la possibilità di poter ripetere settimanalmente tutte le cerimonie con le stesse modalità ha permesso al gruppo di accumulare informazioni e di approfondire la loro conoscenza sia del framework Scrum che del contesto generale del progetto. Una delle conseguenze dovute all'aver accumulato un certo numero di presenze alle varie cerimonie è il miglioramento percepito durante la sessione di stima delle US, potendo ora fare un confronto con le stime date nelle scorse sessioni. La valutazione può basarsi quindi su più elementi, dando vita a Story Point sempre più coerenti tra le varie US del backlog di progetto.

Tra le migliorie, per la prima volta il team desidera ritagliare maggior tempo per l'analisi funzionale della US. Tutti i membri finalmente riconoscono che un approfondimento dei Task, prima e dopo averle messe in pianificazione, permette di migliorare le sessioni di Planning poker e la qualità del lavoro dello Sprint in corso. Invogliare il team ad approfondire la fase di analisi dei Task, anticipando eventuali domande e dubbi, è stata una delle prerogative dello Scrum Master durante lo svolgimento del progetto. Vedere che ora è il team stesso a voler dedicare maggior tempo a questo tipo di attività è sintomo della maturità raggiunta e un altro grosso risultato per il team, oltre alla capacità di autorganizzazione citata prima.

Proprio perché il Dev team ha dimostrato una forte propensione all'autorganizzazione a loro interno, lo Scrum master per i successivi Sprint ha deciso di affidare il compito di scelta e assegnazione dei Task al gruppo stesso. Questo vuol dire che durante gli Sprint Planning il PO indicherà quelli che sono le US prioritarie da mettere in pianificazione, ma l'ammontare di Story Point da inserire verrà deciso dal team, come anche l'assegnatario di ciascun Task. Da questa fase del progetto in poi lo Scrum Master ripone molta fiducia nelle capacità valutative del team, avendo la garanzia che il Tech A potrà intervenire in qualsiasi momento per riportare il team sulla strada giusta, poiché le scelte fatte dovranno, comunque, sempre essere in linea con il pensiero del PO.

4.2 Stima delle User Story

Il primo dei tre punti, su cui si è deciso di approfondirne le dinamiche, è inerente al processo di stima delle US, ovvero alla definizione dell'effort necessario per il completamento. Come già anticipato nei capitoli precedenti, il processo di stima delle US avviene mediante un processo di votazione da parte del Dev Team durante la cerimonia dello Scrum Poker, utilizzando la serie di Fibonacci. Una corretta stima delle User Story è fondamentale per poter creare Sprint con carichi di lavoro idonei, in modo tale che il team possa portali alla conclusione nei tempi previsti. Per poter fornire delle stime il più coerenti possibili con quella che è la difficoltà reale di esecuzione delle US, è necessario che il Dev team abbia bene in mente quali sono tutti i Task e le applicazioni tecniche (codice, query, ecc.) necessarie. Per far questo è essenziale la presenza del PO, che dovrà spiegare la US dal punto di vista funzionale per poi introdurre gli Acceptance Criteria stabiliti con il cliente.

Detto ciò, non avendo mai partecipato ad una sessione di Scrum Poker, il Dev Team ha bisogno di essere guidato e di esercitarsi ad assegnare dei voti che possono prendere i valori di 3-5-8-13-21 dove 21 vuol dire difficoltà massima. Nella prima sessione di Scrum Poker viene svolta un esercizio per prendere dimestichezza con la scala di Fibonacci e per capire come si svolgono le sessioni di votazione. Questo esercizio consiste nel dare dei voti riguardo la grandezza degli animali (es: 3 la formica e 5 il cane), per ogni animale proposto vengono svolti due sessioni di votazioni. Tra queste due sessioni si ha un momento di confronto tra i votanti, per spiegare i motivi che hanno portato ciascun membro a fornire quel voto. Il secondo giro di votazione serve per far capire al team come, dopo la discussione, qualcuno possa appoggiare il pensiero di un altro collega e aggiustare il suo voto. Viene fuori quindi una pillola dello Scrum, "il confronto e la cooperazione per trovare una soluzione che metta d'accordo tutti in modo definitivo". Infatti, possono essere svolte anche più sessioni di votazione, l'importante è che il team abbia capito realmente la US e che il voto dato sia accettato da tutti poiché una volta deciso non sarà più possibile cambiarlo.

Dopo l'esercitazione, si passa subito a stimare le prime US indicate dal PO, applicando le regole viste durante la simulazione. Interessante è valutare le dinamiche del primo confronto post-votazione. Il membro più esperto (Tech A), cerca di guidare la discussione per indirizzare il voto nella direzione corretta, avendo già lavorato alla versione precedente dello IOC, conosce i rischi e le difficoltà tecniche che possono sorgere da questo progetto,

per questo il suo parere e il suo voto influenzerà tanto il parere dei meno esperti. La presenza del Tech A crea sicuramente serenità e sicurezza tra i membri del team, che avranno "meno paura di sbagliare". Questo però rischia di alienarli dall'obiettivo della sessione, ovvero la corretta comprensione del requisito. I membri meno esperti, più che farsi convincere, devono puntare a capire a fondo le varie US, poiché saranno loro a lavorarci durante i vari Sprint. Bisogna quindi invogliare il team ad essere più proattivi e a fare più domande, sia al PO che al Tech A e non dare a priori per buono ciò che questi decretano. Proprio per evitare la grande influenza del Tech A, lo Scrum Master decide che per le prime sessioni questo non parteciperà ai momenti di confronto post-votazione. Ciò creerà delle grosse difficoltà di analisi tra i membri meno esperti, dando vita a una sovrastima o sottostima delle US, ma è un modo per assicurarsi che il team si sforzi a capire ed eventualmente ad arrivare a delle conclusioni errate che fanno parte del percorso di apprendimento. Dopo la prima sessione di votazione, è possibile delineare un *"punto di riferimento"* per le sessioni successive, infatti, guardando gli Story Point dati a ciascuna US, viene definito qual è il valore di questo (La Creazione delle operations, view, layout, screen è considerato dal team come attività "facile", che vale 1 Story Point). Questo permetterà di uniformare il voto del team avendo dando a tutti lo stesso riferimento.

La prima sessione di Scrum Poker non ha avuto molto successo, durante il processo di Sprint Review inerente alla conclusione del primo Sprint sono state chiuse pochissime attività rispetto a quelle pianificate. Il Dev Team ha reso noto allo Scrum Master che i motivi non sono legati tanto alle difficoltà di implementazione ma alla *"mancanza di tempo"*, credono che uno Sprint di due settimane sia troppo poco per chiudere un numero consistente di attività. Nella metodologia Agile la durata degli Sprint viene decisa in base alle esigenze del team ma soprattutto in base alle richieste del cliente: come detto prima uno dei capisaldi dell'Agile è quello di consegnare dei pezzi funzionanti di software alla conclusione di ogni Sprint. Detto ciò, una volta deciso che gli Sprint dureranno due settimane, il PO si aspetta di vedere degli avanzamenti ogni due settimane e tale arco temporale stabilito non è modificabile, salvo emergenze o gravi motivazioni. La grande quantità di debito tecnico accumulatasi nel primo Sprint è dovuto alla *"sottostima dei Task"* che scaturisce da:

- *Inesperienza*: Il team sulle ali dell'entusiasmo ha assegnato degli story point troppo bassi che non rispecchiano il reale effort necessario

- *Scala di Fibonacci*: Tale scala risulta essere molto insidiosa, poiché essendo una successione di numeri interi in cui ciascun numero è la somma dei due precedenti (eccetto i primi due che sono, per definizione, 0 e 1). Risulta difficile passare ad un valore inferiore o superiore una volta data una stima iniziale, ad esempio: il Team ha riscontrato che durante le stime con voti 3-5-8-13-21, fa molto riferimento alla distanza tra i valori. Il passaggio da 3 a 5 non è banale perché vi è un aumento di quasi il doppio del valore, indica quindi che si sta considerando l'attività mediamente il doppio più difficile rispetto alla stima iniziale. Questo porta a non modificare il voto e a sottostimare la User Story. Il team deve abituarsi a stimare considerando tutti i valori all'interno della scala e non soffermarsi tanto sulla distanza tra i valori.
- *Comprensione delle US:* Le US vengono scritte solitamente dal PO utilizzando un linguaggio più semplice possibile, nonostante ciò, il team ha avuto notevoli difficoltà di comprensione e accusa che ogni US racchiude una *"quantità di requisiti troppo elevata"*. Per alcune di queste, infatti, sarebbe necessario dividerle in più US durante l'incontro di refinement, per poter definire meglio i vari Task che la compongono. Lo split delle US può generare un duplice effetto uno positivo e uno negativo: come accennato prima, permette una più rapida comprensione della feature, d'altro canto splittare le US potrebbe farle risultare ancora più semplici agli occhi del team alimentando il rischio di sottostima che ha colpito la prima fase di votazione.

Questi tre problemi sono stati individuati dallo Scrum Master e successivamente notificati in fase di Sprint Retrospective. Tutto il team dovrà tenere a mente cosa emerso durante queste prime sessioni di Planning poker e cominciare a fornire delle stime più coerenti con quelle che sono le capacità attuali del team.

Con l'avanzare del progetto il team acquisisce esperienza e consapevolezza con il processo di stima, facendo tesoro degli insegnamenti delle sessioni precedenti. Gli Sprint cominciano a contenere un numero di Story Point tali da consentire il corretto completamento di questo e si comincia a rilasciare qualche nuova funzionalità osservabile dal cliente. Il parere del Tech A non viene più considerato come "verità assoluta", ma come ottimo punto di partenza su cui basare un dibattito costruttivo che avrà come output un effort idoneo alla reale difficolta delle US. In generale, con l'avanzare del progetto si nota come alle US venga assegnato un effort sempre minore, questo è dato dal team che, dopo

aver portato a termine alcune US, mette in atto il cosiddetto *"riciclo"* ovvero la possibilità di riutilizzare query e pezzi di codici più volte, tra User Story diverse, rendendo di fatto più rapida l'esecuzione.

L'entusiasmo del team che ha inizialmente portato a sottostimare le US nel corso del progetto rimarrà costante, in quanto il team, voglioso di aumentare la produttività di ogni Sprint, chiede in fase di Planning Poker di provare a mettere un effort iniziale e vedere se effettivamente è idoneo o è troppo alto per quelle che sono le attuali capacità del team. Si chiede, sostanzialmente, di cambiare durante uno Sprint in corso gli effort di una User Story nel momento in cui a fronte di ulteriori valutazioni questa risulta essere più facile del previsto. Tutto ciò entra in contrapposizione con quanto affermato da Scrum: in fase di Planning Poker bisogna uscire con un effort definitivo e durante lo Sprint non è possibile cambiare l'effort in corso, poiché, come detto precedentemente, le attività inserite nello Sprint sono scelte anche in base alla quantità di effort massimo, solitamente intorno a 35. Cambiare l'effort di una US in corso d'opera equivale, durante lo Sprint Planning, a sovrastimare o sottostimare il numero di US inserite.

Vista l'ottima risposta da parte del team alla metodologia, verso la parte conclusiva del progetto lo Scrum Master invece che aumentare la produttività di ciascuno Sprint aumentando il numero di US in pianificazione, propone al team due cambiamenti che potrebbero portare benefici sia nella produttività che nella qualità del lavoro svolto, queste proposte consistono in:

- Cambiare la scala che definisce uno Story Point, passare da definire non più l'operation come effort pari a 1, bensì la creazione dell'HTML che risulta essere una attività ben più complicata della precedente. Così facendo, mantenendo inalterato il numero di Story Point medio eseguibili ad ogni Sprint, si sta indirettamente aumentando la produttività, poiché adesso uno Story point ha una valenza maggiore. Questa proposta sarà però accantonata in quanto si prevede l'ingresso di nuovi membri all'interno del Dev Team che, avendo ovviamente meno esperienza e conoscenze del contesto, troverebbe delle difficoltà nell'avere come riferimento per uno Story Point una attività ben più complicata rispetto a quella indicata inizialmente.
- Cominciare ad eseguire un approfondimento del requisito durante le sessioni di Planning Poker per proporre nuove soluzioni tecniche e per fare una analisi del

rischio alla soluzione attuale. Ora che il team possiede un bagaglio culturale sugli aspetti tecnici maggiore, una volta compresa la nuova US, gli verrà chiesto di non limitarsi a fornire una semplice stima, ma di individuare e rendere noti nei momenti di confronto quali possono essere i rischi e migliorie che potrebbero sorgere applicando la soluzione tecnica proposta nella US. Lo Scrum Master, per l'analisi del rischio, chiede quindi di effettuare una vera e propria analisi di ciascuna attività, evidenziandone "l'impatto" e la "soluzione tecnica", per poi aggiungerle direttamente in backlog e trattarli come veri e propri Task associabili alla US. Per quello che riguarda le nuove proposte, queste devono prima essere validate dal PO, che dovrà assicurarsi che la nuova soluzione tecnica porti al raggiungimento del risultato, ovvero al completamento degli Acceptance Criteria.

Nonostante la prima proposta venga declinata per motivi organizzativi del team, la proposta relativa all'approfondimento delle US verrà presa in considerazione, provando ad applicare quanto richiesto dallo Scrum Master se pur con scarsi risultati.

4.3 L'autorganizzazione

In questo paragrafo il focus è centrato sul passaggio al concetto di autorganizzazione, ovvero passare da una metodologia in cui i membri del team di sviluppo erano abituati a un'assegnazione di Task da eseguire con dei periodi di consegna variabili, a dover organizzarsi il lavoro al loro interno per portare a compimento lo Sprint.

Abbiamo visto come in fase di Sprint Planning, una volta scelte le attività da mettere nello Sprint, si passa con la definizione dell'assegnatario, ovvero colui che sarà responsabile della corretta esecuzione della US e di tutti i Task collegati ad esso, questo però non implica il fatto che anche gli altri membri del gruppo sono coinvolti nell'implementazione di quella determinata attività. Infatti, il concetto di *"mischia"* si basa proprio sulla pressione che il team esercita verso un unico obiettivo, che è appunto il completamento dello Sprint. Tutti sono responsabili dei successi e degli insuccessi: è essenziale che il team nelle due settimane di svolgimento dello Sprint abbia un piano di massina per poter completare tutte

le attività in tempo, concordando su quali attività ogni membro dovrà intervenire indipendentemente che sia l'assegnatario di quella US oppure no.

La pianificazione del lavoro si ha a grandi linee durante lo Sprint Planning ma viene raffinata giorno dopo giorno attraverso i Daily Meeting e le continue interazioni tra il team. I membri del Dev Team si troveranno molte volte a lavorare su diversi Task interdipendenti tra loro ma appartenenti alla medesima US, per questo uno degli elementi critici diventa la comunicazione che dovrà essere chiara e soprattutto proattiva per non demoralizzare i colleghi qualora si presentassero problemi.

Nella fase iniziale del progetto le difficoltà riscontrate nel completamento del primo Sprint, oltre che hai problemi legati alla sottostima delle US viste nel paragrafo precedente, erano legati anche alla mancata collaborazione tra i membri, che per insicurezza (e forse anche un po' d'orgoglio), tendevano a svolgere tutte le attività assegnate in maniera autonoma senza chiedere alcun tipo di sostegno sia riguardo la parte funzionale che tecnica. Trattandosi delle fasi iniziale del progetto, a nessuno dei membri meno esperti era chiarissimo il contesto in cui si agiva; perciò, era più che opportuno chiedere aiuto al Tech A e al PO durante gli incontri giornalieri. Come accennato prima, durante lo svolgimento del primo Sprint Retrospective, è emerso che uno dei motivi principali del perché i membri meno esperti non hanno richiesto il supporto necessario, era legato al fatto che non volevano rallentare il lavoro degli altri, soprattutto quello del Tech A, che essendo quello più esperto gli risultano assegnati un numero maggiore di Task. Questo disguido riguardo la sensazione di intralcio del lavoro da parte dei membri meno esperti è stato risolto con tempestività da parte dello Scrum Master, invogliando il team a collaborare e a fare domande anche banali.

Con l'avanzare del progetto lo Scrum Master capisce come ogni membro del team si sta specializzando su un'attività tecnica in particolare, diventando esperto di specifiche modifiche da effettuare. Una volta che un membro comincia a lavorare su determinate funzionalità simili tra loro, allora continuerà a lavorare su quelle direttamente collegate o che da un punto di vista tecnico sono simili, piuttosto che spostarsi su un altro Task completamente diverso. Questo evita ogni volta di dover eseguire i "passaggi di consegna" che costano tempo e fatica per spiegare come svolgere un determinato Task. In particolare, i passaggi di consegna avvengo sia a voce sia tramite creazione di documenti ad hoc inseriti

poi nella sezione wiki di Azure Devops, in modo tale che tutti possono accedervi in qualsiasi momento. Tali documenti vengono creati principalmente dal Tech A e dal PO.
Insieme alla crescente coesione del Dev Team, che ha portato all'instaurazione delle routine organizzative citate prima, la possibilità di evitare i passaggi di consegna è uno dei motivi principali che ha spinto lo Scrum Master a lasciare sempre più libertà al team di organizzarsi in maniera autonoma il lavoro, permettendo a ciascuno di poter assegnare in maniera autonoma i vari Task da completare. Questa ulteriore responsabilità ha portato il team ad effettuare una nuova operazione di stima, ovvero il tempo necessario per il completamento di un Task. Questo agli occhi del Dev Team dovrebbe facilitare l'organizzazione del lavoro in fase di Sprint Planning agevolando la stima e la scelta delle attività da inserire in pianificazione.

Sempre per migliorare l'autorganizzazione del team, è essenziale che questo partecipi in maniera attiva alle sessioni di Sprint Review analizzando quali e quante attività sono andate a buon fine e quali no. In particolare, uno strumento importante che il team ha avuto la possibilità di analizzare per capire se uno Sprint ha avuto successo è lo Sprint Burndown Chart. In questo caso è stato preso come riferimento lo Sprint 3, che come accennato prima, ha sancito un po' un punto di svolta nell'esecuzione del progetto. Grazie a questo Sprint il team si è reso conto che le cose non stavano andando per il verso giusto e che c'era bisogno di un cambiamento nell'atteggiamento nei confronti dello Scrum, cominciando ad essere più cooperativi l'uno con l'altro.

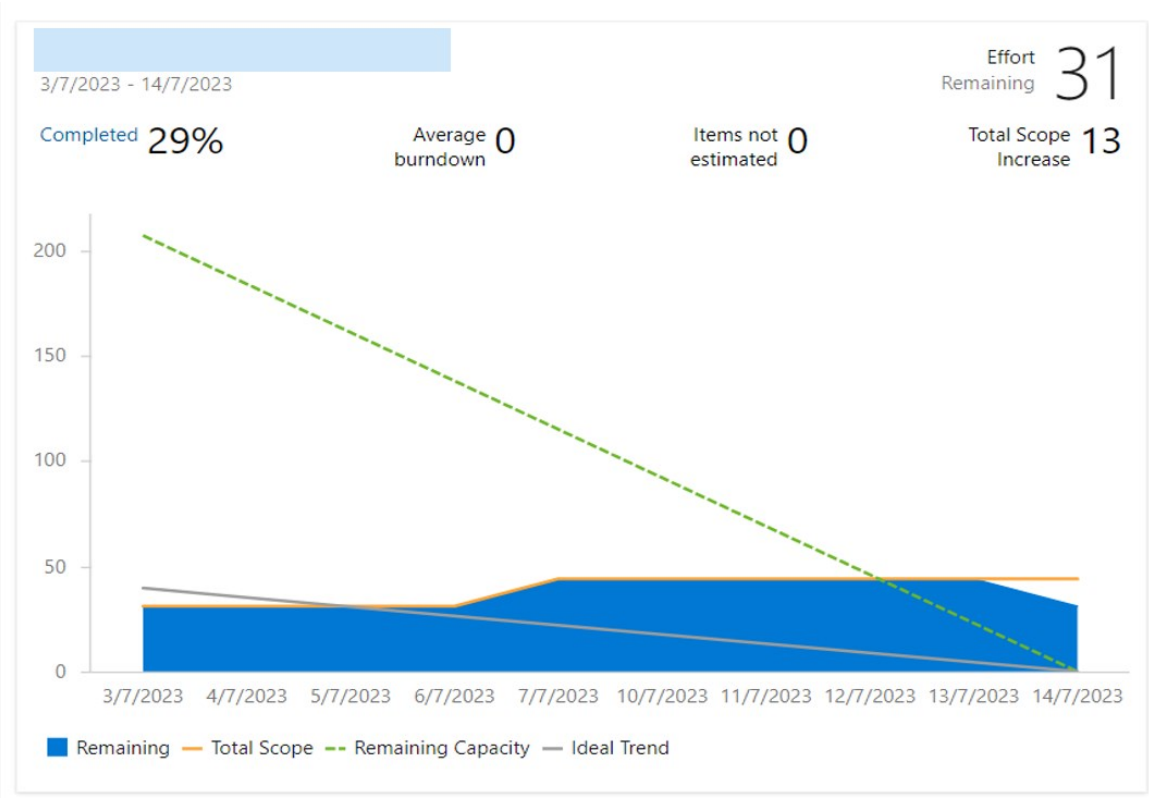

Figura 11. Sprint burndown chart "Sprint 3"

Nello Sprint appena concluso sono rimasti 31 Story Point: la linea grigia indica il trend ideale di chiusura di tutte le User Story, mentre l'area in blu le evoluzioni degli Story Point conclusi nelle due settimane dello Sprint. Osservando il total scope dello Sprint, che in data 6/07 ha subito un innalzamento, notiamo come siano stati aggiunti degli Story Point che hanno portato un conseguente aumento di volume dell'area blu. In questo caso l'aumento di Story Point è dovuto all'aggiunta di una User Story, inserita dal PO in accordo con il team di sviluppo. Le motivazioni legate a questa scelta sono date dal fatto che durante lo svolgimento dello Sprint, il team avrebbe avuto il supporto di membri inizialmente non presi in considerazione durante lo Sprint Planning. Di fatto questo è osservabile dalla linea grigia, ovvero l'ideal trend che avrebbe dovuto avere lo Sprint considerando l'effettivo numero di risorse a disposizione. Gli Story Point passano di fatto a 44, considerato un numero idoneo seppur abbastanza elevato rispetto alla media di progetto. Nonostante il team fosse al completo, e che quindi ci fossero i presupposti per instaurare un'organizzazione stabile per cominciare a mettere in atto delle routine organizzative, il team è riuscito a chiudere solo il 29% degli story point pianificati, anche se in questo grafico non viene detto che le US non chiuse sono comunque state abbondantemente lavorate, ma

nello Scrum non c'è la via di mezzo, le attività o sono chiuse o costituiscono un debito tecnico per il prossimo Sprint.

Ultimo tema legato all'organizzazione è la presenza o meno dei membri del team di sviluppo durante gli Sprint. Essendo una società di consulenza, i dipendenti di Aubay Italia, possono trovarsi nella condizione di poter lavorare contemporaneamente su progetti diversi, questo implica che in determinati periodo un membro potrebbe risultare assente o parzialmente disponibile a lavorare sul progetto. Risulta quindi fondamentale durante lo Sprint Planning accertarsi quali siano i membri del team presenti, e se presenti quanto tempo possono dedicare a questo. Conoscendo questo particolare in più, eventuali assenze e imprevisti da parte di un membro possono essere colmate da una stretta collaborazione all'interno del team, qualora le risorse a disposizione per l'esecuzione dello Sprint siano scarse è possibile diminuire l'ammontare di Story Point da inserire in fase di Sprint Planning, se invece le assenze dovessero verificarsi durante lo Sprint in corso sarà compito dei membri del team collaborare per cercare di coprire il più possibile il collega assente e preservare le routine organizzative essenziali per lo svolgimento delle attività.

4.4 La relazione con il Product Owner interno

Come già accennato più volte, l'intero Scrum Team è completamente composto da dipendenti di Aubay Italia, compreso il PO, figura che cerca di massimizzare l'utilità del cliente rendendo note al team di lavoro quali sono le sue aspettative sull'esecuzione delle attività nel Product Backlog. In uno Scrum Team tradizionale, il PO entra in contatto con il team di sviluppo in fase di Refinement delle User Story e durante le cerimonie di Sprint Planning e di Sprint Review, lasciando delle proprie opinioni riguardo ciò che va messo in pianificazione e ciò che può essere considerato concluso. Essendo in questo caso una figura a stretto contatto con tutto il team, la sua posizione risulta essere molto delicata fin dall'inizio, dà un lato è evidente la sua volontà di aiutare il team a performare al meglio, mentre dall'altro sarà costretto a mettere dei paletti per assicurarsi il rispetto di quanto concordato con il cliente.

Il comportamento del PO e le relazioni con il team hanno subito una forte evoluzione durante lo svolgimento del progetto, complice anche il fatto che lui stesso ha avuto delle difficoltà ad approcciarsi a questo ruolo un po' anomalo, difficile da interpretare.

All'inizio del progetto, con il team completamente estraneo alle logiche Agile, in accordo con lo Scrum Master, il PO ha preso le veci di una figura autoritaria, intervenendo per spiegare i requisiti nel dettaglio e per manifestare disappunto nel momento in cui le stime date alle US e le priorità individuate non erano in linea con il suo pensiero. Questo approccio secondo lo Scrum Master era utile per far comprendere in fondo le dinamiche dello Scrum, per attuare quel processo di formazione essenziale per il Dev Team. Purtroppo, gli insuccessi dei primi tre Sprint sono legati da questo distaccamento del PO, è vero che l'applicazione delle vere dinamiche comunicative tra il PO e il team sono state utili per dare una panoramica completa di cosa vuol dire lavorare in Agile, ma è anche vero che le difficoltà iniziali legati alla corretta comprensione delle US e alla quantità enorme di Task da lavorare sono correlate anche al formale comportamento del PO. Proprio nelle fasi iniziali, in cui la comprensione del contesto del software da un punto di vista funzionale è fondamentale, limitarsi a fornire poche informazioni a riguardo nelle mere sessioni di Planning Poker non è stato abbastanza per aiutare il team. Inoltre, il Product Owner ha premuto fin da subito per inserire all'interno degli Sprint una quantità idonea di Story Point, invece di andare incontro a quanto espresso dal Dev Team che ha comunque comunicato di voler partire con un numero abbastanza basso intorno ai 20-25, consapevole proprio della sua inesperienza. Ricordiamoci che il framework Scrum prevede che l'ultima parola in merito alla quantità di Story Point da mettere in pianificazione spetta proprio alla componente tecnica del team, quindi al Dev Team. Nel nostro caso proprio a fronte dell'inesperienza dei partecipanti, la decisone riguardo la quantità di Story Point è decisa dallo Scrum Master che ovviamente sentirà l'opinione del PO che cercherà di influenzare il risultato a suo favore. Come descritto nel paragrafo precedente, questa regola verrà cambiata a partire dallo Sprint 7.

Per dare una scossa al progetto, oltre i provvedimenti presi per migliorare l'autorganizzazione e la stima delle US visti nei due paragrafi precedenti, lo Scrum Master decide di cambiare il ruolo e i compiti del PO. Il grande cambiamento prevede la *partecipazione del Product Owner ai Daily Meeting*, una decisione molto importate a detta di tutto il team che porterà dei grossi benefici. In realtà questa decisione avrà un duplice effetto sul proseguo del progetto.

La presenza del PO ai daily ha permesso al team di approfondire le User Story. La sua figura permette al team di esporre dubbi e perplessità riguardo le feature da implementare. Capire al meglio le User Story dal punto di vista funzionale facilita la ricerca alla soluzione tecnica più idonea, processo ancora abbastanza complicato in quanto a inizio progetto non si ha ancora la possibilità di poter riciclare il codice da vecchie US. Altro aspetto positivo è la prioritizzazione dei Task da completare, nel caso in cui il team capisse di non riuscire ad arrivare a completare tutte le attività pianificate. In quel caso sentire il parere del PO su cosa focalizzarsi è molto importante poiché sarà lui a dover poi fare il resoconto dello stato dell'arte del progetto direttamente agli Stakeholders, qualora gli venisse chiesto.

D'altro canto, avere il PO presente a tutti i Daily Meeting, vuol dire anche fornirgli degli aggiornamenti su cosa ci si occuperà oggi e cosa è stato fatto ieri. Ovviamente il suo obiettivo sarà quello di arrivare alla Sprint Review con tutte le US messe in pianificazione che rispecchiano gli Acceptance Criteria definiti. Per fare ciò il PO non si farà problemi a mettere pressioni sul team per velocizzare il lavoro, pretendendo comunque degli standard qualitativi elevati.

La partecipazione del PO ai daily sarà una variabile permanente fino alla conclusione del progetto, con l'avanzare dei lavori si instaurerà un rapporto di fiducia tra il team e il PO che cercherà di rendersi utile anche per facilitare la coordinazione all'interno del Team.

5. Aspetti positivi e negativi

Il framework Scrum ha dato la possibilità al gruppo di scoprire un modo nuovo di approcciare i progetti IT e di applicarlo correttamente per la riuscita del progetto.

Come emerso dai paragrafi precedenti durante lo svolgimento sono stati fatti dei cambiamenti in corso d'opera per poter migliorare concretamente la qualità del lavoro e mettere nelle condizioni il team di prendere le decisioni corrette. È possibile quindi affermare che la scelta di gestire il progetto in modalità Agile ha portato i risultati sperati. In questo capitolo verrà effettuata una analisi critica sulle decisioni e dinamiche che non hanno permesso di massimizzare i benefici portati dallo Scrum e quelli che invece hanno facilitato l'applicazione di questa metodologia. Per comodità sono stati divisi tra aspetti positivi e negativi di progetto.

5.1 Aspetti negativi

Come emerso dai paragrafi precedenti, durante lo svolgimento del progetto sono stati fatti dei cambiamenti in corso d'opera per poter migliorare concretamente la qualità del lavoro e rimediare quindi a dinamiche controproducenti alla corretta applicazione dello Scrum e per il processo di apprendimento del team. Oltre che a descrivere i motivi per il quale questi aspetti sono stati individuati come "negativi", verranno fornite anche possibili soluzioni e approcci alternativi che avrebbero potuto giovare l'applicazione dello Scrum.

5.1.1 L'influenza dello Scrum Master e del Tech A

A causa dell'inesperienza del Dev Team, all'inizio del progetto lo Scrum master e il Tech A sono state delle figure molto presenti per il team per motivi diversi. Lo Scrum Master per motivi prettamente legati alla corretta applicazione della metodologia Agile mentre il Tech A per aiutare il gruppo alla corretta valutazione delle User Story e per indirizzare il team a prendere le soluzioni tecniche migliori per il completamento dei Task. Le forti influenze di queste due figure hanno sicuramente aiutato il team a finire il progetto nei tempi e nei modi

corretti, ma in certi aspetti sono quasi stati di intralcio per il processo di apprendimento del team sul framework Scrum, ma anche sulla parte di sviluppo coding del progetto. Tutte e due le figure hanno controllato nel dettaglio le attività e le scelte organizzative del team intervenendo nell'immediato per evitare eventuali escalation di problemi che avrebbero potuto impattare il progetto. Tutto ciò non ha permesso di effettuare il processo del *learning by doing,* ovvero dare la possibilità al team di sbagliare più volte per poi trovare la quadra giusta per approcciare il progetto al meglio. I motivi della forte influenza da parte di queste tre figure sono:

- *Scrum Master*: Non ha permesso al team di auto-organizzarsi fin dall'inizio in maniera autonoma, poiché in fase di Sprint Planning le scelte relative agli assegnatari dei vari Task e alla quantità di Story point inseriti all'interno di uno Sprint sono state fortemente influenzate dallo Scrum Master ritenendo il team non capace di prendere decisioni di questo genere, almeno all'inizio del progetto.
- *Tech A*: Come già citato nei capitoli precedenti, troviamo una forte influenza da parte di questa figura nel processo di valutazione delle User Story durante la fase di Planning Poker, cercando di influenzare il parere dei meno esperti sull'effort a suo avviso più idoneo.

Dare la possibilità al team di effettuare in maniera autonoma delle decisioni così complicate e delicate fin dall'inizio avrebbe da un lato generato dei risultati ancor più negativi sui primi Sprint pianificati, compromettendo di non poco l'avanzamento del progetto, ma avrebbe diminuito il periodo necessario per l'apprendimento delle logiche Scrum, in quanto avrebbe dato una percezione migliore degli errori fatti dal team durante le sue analisi.

5.1.2 Assenza di pre-analisi del Tech A

La grande conoscenza del Tech A sul contesto tecnico del progetto è stata sfruttata per supportare i membri meno esperti durante lo svolgimento delle attività e per la stima delle User Story. Il Tech A, durante i Daily Meeting, era sempre disposto a intervenire nei momenti di difficoltà nell'eventualità che le soluzioni adottate dal Dev Team per il completamento dei Task non fossero idonee. Lo stesso accadeva in fase di Planning Poker: abbiamo visto come il Tech A cercava di indirizzare il voto del Dev Team sull'effort più adatto per lo svolgimento delle US. In entrambi i casi, l'intervento del Tech A avveniva dopo una prima analisi da parte del Dev Team, che si basava sulle informazioni riportate nelle US e sugli approfondimenti fatti con il Product Owner. Solo dopo che il Team aveva maturato una prima considerazione sul lavoro da eseguire il Tech A faceva delle valutazioni personali e interveniva qualora lo rendesse opportuno.

Questa tipologia di supporto dato dal Tech A è stata sicuramente d'aiuto, ma non ha massimizzato a pieno le competenze messe da lui stesso a disposizione. Effettuare degli interventi correttivi posteriori l'analisi del team è sicuramente un approccio corretto e utile, ma per far fronte all'inesperienza del Team a inizio progetto sarebbe stato opportuno che il Tech A effettuasse una analisi delle US presenti in backlog prima che il progetto partisse. Il Tech A avrebbe potuto svolgere una vera e propria analisi del rischio dal punto di vista tecnico, aggiungendo le sue considerazioni sul ticket stesso della User Story. In questo modo nel momento in cui un membro del Dev Team si approcciava per la prima volta ad una nuova US, oltre che a trovarsi una descrizione dal punto di vista funzionale, avrebbe trovato già degli spunti da cui partire per abbozzare una possibile soluzione, con annessi rischi che possono emergere.

Bisogna sottolineare che la proposta di far eseguire una pre-analisi delle attività non è prettamente Agile ma una soluzione che, a mio avviso, avrebbe aiutato nel concreto il team a superare le difficoltà iniziali. Una corretta applicazione del framework Scrum prevede che sia il team nella sua interezza ad effettuare questo tipo di analisi e non di affidare questo compito ad un'unica figura, infatti nel paragrafo 3.2, dopo che il team ha maturato maggiore esperienza riuscendo a specializzarsi su certe tipologie di attività, lo Scrum Master ha chiesto espressamente al team di cominciare ad eseguire una analisi del rischio di ciascuna User Story che veniva analizzata. Infine, non bisogna dimenticare che ci troviamo di fronte ad un progetto atipico, in quanto, trattandosi di un progetto di *"migrazione"* e non di puro

"sviluppo software", il backlog è sostanzialmente già ben definito con *Change Request* praticamente nulle da parte del cliente. Nel caso in cui ci fossimo trovati di fronte ad un progetto con User Story ancora da definire, è facile intuire che una pre-analisi da parte del Tech A sarebbe stata infattibile, poiché le funzionalità sarebbero state concordate e definite nei vari refinement svolti durante il progetto.

5.1.3 La variabilità numerica del team

Per tutta la durata del progetto il Dev Team è stato composto da un numero variabile di componenti che si aggiravano tra i tre e i sei partecipanti. La causa è legata al fatto che alcuni membri non erano completamente allocati sul progetto di migrazione dello IOC, ma avevano l'obbligo di prestare servizio su altri progetti che venivano svolti in parallelo. I membri del Dev Team che lavoravano "full" al progetto erano tre, tra cui uno era proprio il Tech A. La restante parte a seconda delle indicazioni che riceveva dal proprio superiore comunicava la propria disponibilità allo Scrum Master per le due settimane inerenti allo Sprint in partenza. Nel corso del progetto si sono comunque verificati dei casi in cui alcuni membri hanno avuto delle urgenze sugli altri progetti proprio in mezzo ad uno Sprint in corso, mettendo in difficoltà l'intero gruppo costretto a rivedere la propria organizzazione interna per cercare di coprire l'assenza di un collega.

Non avere un numero di partecipanti fissi durante lo svolgimento del progetto è stata una variabile che sia lo Scrum Master che il Dev Team hanno dovuto gestire effettuando delle variazioni in corso d'opera per poter salvaguardare la riuscita del progetto. Le principali criticità emerse sono:

- *Perdita di tempo per i passaggi di consegna e per la spiegazione delle US*: L'incostante presenza di alcuni membri ha obbligato il team fare delle riunioni ad hoc per far sì che tutti membri fossero a conoscenza di quanto emerso nelle cerimonie a cui non hanno partecipato.
- *Difficoltà nell'instaurare delle routine organizzative*: Abbiamo già sottolineato come avere delle routine solide all'interno di uno Scrum Team è necessario per migliorare la produttività e la qualità del lavoro. Per instaurare dei rapporti del genere è necessario un periodo di adattamento, il

team ha bisogno di conoscersi e di sviluppare delle conoscenze sulla materia, ma l'assenza sporadica di alcuni membri ha ritardato di molto questo processo non potendo coinvolgere tutti.

- *Accumulo di Debito Tecnico*: Molti Task che vengono messi in pianificazione in uno Sprint sono interdipendenti tra di loro con legami del tipo Finish to Start. Questo vuol dire che nel momento in cui il responsabile di una determinata attività, il cui completamento è necessario per l'inizio di un'altra, è assente, non sarà possibile finire tutte le attività pianificate generando così del debito tecnico da inserire nello Sprint successivo.
- *Determinare il numero di Story Point in uno Sprint*: L'incertezza delle risorse a disposizione ha portato a sua volta incertezza nel determinare quanto lavoro va assegnato in uno Sprint. Nelle due settimane i membri possono aumentare ma anche diminuire quindi vuol dire che l'ammontare di Story Point inseriti possono risultare: troppi nel caso in cui alcuni componenti non riescono a lavorare i Task assegnategli o pochi nel caso in cui alcuni componenti non considerati durante lo Sprint Planning riescono a staccarsi dagli altri progetti e a supportare il Dev team.

L'unica soluzione per far fronte al problema della variabilità del team è quello di basarsi sulle sole risorse a disposizione in fase di Sprint Planning e pianificare il lavoro su quelle. Inoltre, bisogna cercare di variare il meno possibile il pianificato anche a fronte di assenze/presenze impreviste, poiché come visto per lo Sprint 3, l'aggiunta di US allo Sprint iniziale, dato l'aumento di componenti disposti a lavorare, ha mandato in confusione l'intero gruppo portando ad un risultato disastroso.

5.2 Aspetti positivi

Tra gli aspetti positivi troviamo, invece, tre caratteristiche del progetto che hanno facilitato l'applicazione della metodologia Agile e che si sono rivelati dei punti a favore per il team. Si fa riferimento sia a decisioni prese dallo Scrum Master stesso per agevolare il lavoro, sia a dinamiche casuali di contesto che hanno facilitato l'applicazione della metodologia Agile.

5.2.1 Livello di conoscenza del Dev Team

Tutti i componenti del Dev Team all'inizio del progetto presentavano una conoscenza di base sul framework Scrum pari a zero, poiché abituati a lavorare in progetti con un approccio simile a quello a cascata. Come già descritto in precedenza lo Scrum Master ha dovuto effettuare delle lezioni preliminari per insegnare al team il framework Scrum per assicurarsi che il team arrivasse con un minimo di preparazione all'inizio del progetto. L'Agile oltre che ad essere un metodo ben definito che si avvale dell'utilizzo di strumenti come gli artefatti e le cerimonie, è anche una vera e propria *filosofia di lavoro* che per essere assimilata al meglio da parte di un gruppo è necessario che sia provata e aggiustata dal gruppo stesso, sperimentando approcci diversi per poi trovare quello che più si addice alle caratteristiche degli individui.

Nel progetto di migrazione dello IOC è emerso che avere un gruppo di lavoro con esperienza nulla sull'Agile, dà la possibilità a tutti i componenti di poter eseguire un percorso di approfondimento di questa filosofia, trovando insieme un approccio che sia approvato e condiviso da tutti. Nessun componente del gruppo ha avuto delle esperienze pregresse sul framework Scrum che avrebbero potuto intaccare il processo di apprendimento, magari imponendo al team le logiche maturate in altri progetti che non per forza possono garantire lo stesso risultato. Ogni team ha bisogno di sviluppare le proprie routine e la propria concezione di Agile e questo è facilitato se tutti i componenti hanno lo stesso livello di preparazione. Il riciclo di dinamiche già utilizzate in altre organizzazioni non può funzionare in quanto il contesto del progetto e gli individui sono diversi. Anche nel caso in cui il Dev Team tra un progetto e l'altro rimanesse lo stesso, questo dovrà comunque adattarsi alle richieste dello Scrum Master e del PO e al contesto tecnico del progetto.

5.2.2 Libero accesso ad Azure Devops

Il libero accesso a tutti i membri del team ad Azure Devops è risultato essenziale per il completamento del progetto, un plus non indifferente e per niente scontato messo a disposizione dallo Scrum Master. I tre principali strumenti che si sono rivelati d'aiuto per il team sono:

- *User Story e Task*: La possibilità di avere a disposizione le User story e i Task catalogati in ordine di importanza con stati di avanzamento aggiornati in tempo reale ha permesso di tenere traccia del livello di completamento di ciascuno Sprint e ha facilitato la pianificazione di quelli successivi, oltre che ad essere stata una fonte di informazioni per comprendere al meglio i requisiti descritti all'interno. Da sottolineare che la scomposizione delle US e i cambi di stato potevano essere eseguiti rispettivamente dal Product Owner e dallo Scrum Master nelle sessioni dedicate.
- *Artefatti*: Gli Artefatti come lo Sprint Burndown Chart e il Burnup hanno dato una visione delle performance del team all'interno di un singolo Sprint e durante tutto il progetto, permettendo di avere informazioni in più per valutare eventuali migliorie da apportare per migliorare la qualità del lavoro.
- *Wiki*: Avere un contenitore di manuali riguardo le basi teoriche dell'Agile, i codici da utilizzare e la definizione di story point, ha semplificato molto l'accesso ad informazioni essenziali, permettendo a tutti i componenti del team di trovare risposte e soluzioni nell'immediato.

5.2.3 Product Backlog statico

Possedere un Product Backlog definito e statico è abbastanza atipico per un progetto gestito in Agile, poiché una delle fondamenta di questa metodologia è la continua interazione col cliente per gestire il cambiamento dei requisiti, per poi andare ad aggiornare o aggiungere nuove User Story.

Questa variabile ha permesso al Product Owner di evitare sessioni di scrittura delle US e di definizione del requisito insieme al cliente diretto, limitandosi solo a dover scomporre le User Story sotto le direttive del Dev Team per renderle più comprensibili.

Il team di sviluppo ha la possibilità di concentrarsi totalmente sulle attività fornitegli all'inizio del progetto, potendo cominciare ad approfondire fin da subito le US, forti del fatto che non subiranno variazioni. Anche la pianificazione degli Sprint è stata notevolmente semplificata, sapendo che tutte le attività da lavorare erano già presenti in backlog è stato più semplice organizzare il lavoro in modo tale da completare tutte le attività

nei tempi previsti. Eventuali Change Request e nuove US non preventivate generano inevitabilmente un aumento della durata del progetto stravolgendo quanto pianificato, costringendo lo Scrum Master ad aggiungere nuove attività in pianificazione anche per gli Sprint in corso, che porteranno verosimilmente alla creazione di debito tecnico.

6. Considerazioni finali

Attraverso il *Burn up* di figura 12, è possibile intravedere come Scrum abbia avuto degli effetti positivi sul progetto, portando il team ad aumentare nel tempo la percentuale di User Story completate, partendo da un valore molto basso fino ad arrivare allo Sprint 7 dove le performance ottenute durante lo Sprint (Burn-up, linea grigia) sono in linea con quanto aggiunto in pianificazione da Sprint a Sprint (Total Scope, linea arancione), per arrivare allo Sprint 9 con un quantità di debito tecnico al minimo storico (Area blu).

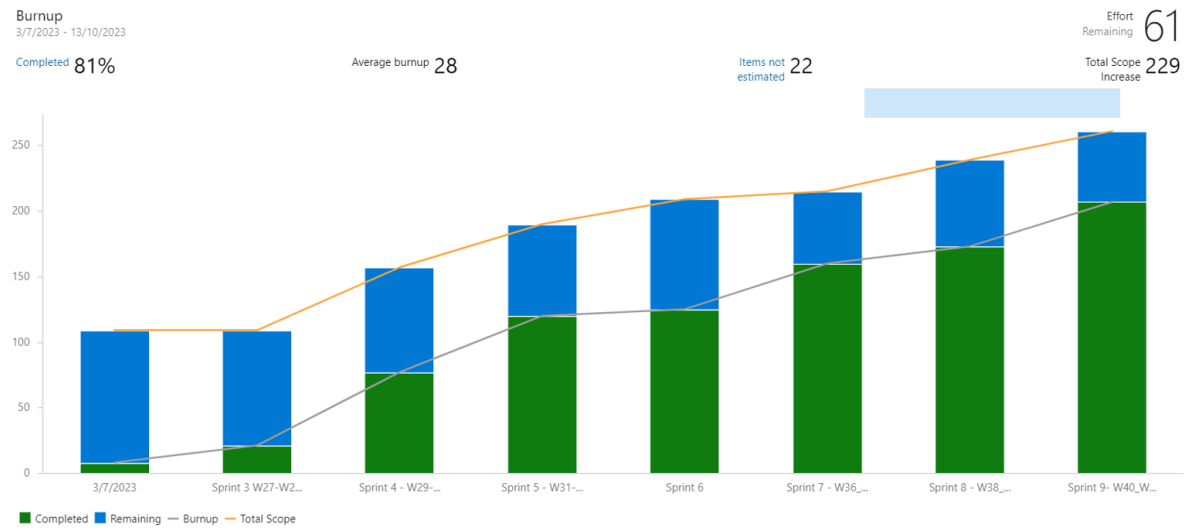

Figura 12. Burn up di progetto

Guardando il graduale aumento di performance del team, è lecito dire che la scelta di gestire il progetto in modalità Agile ha portato i suoi frutti, portando notevoli vantaggi al team soprattutto riguardo l'organizzazione del lavoro. I vantaggi percepiti dal team sono già stati citati nei paragrafi precedenti, ma possono essere racchiusi nei seguenti macro-temi:

- Autorganizzazione del lavoro da svolgere
- Collaborazione tra tutti i membri del Dev Team
- Mole di lavoro da svolgere ben definita nelle due settimane di Sprint
- Monitoraggio continuo dello stato avanzamento lavori

Tuttavia, come detto, più volte, ci troviamo di fronte ad un progetto atipico come quello di migrazione di feature di un software, da un ambiente di partenza (Cloud View) ad uno di destinazione (Apriso), con requisiti già ben definiti e statici. Proprio questa peculiarità del

progetto farà emergere quella che è stata la limitazione principale dovuta all'applicazione del framework Scrum a questo progetto, che porterà alla descrizione di un approccio alternativo che a mio avviso avrebbe potuto migliorare il già ottimo risultato del progetto.

6.1 Lo svantaggio degli Sprint bisettimanali

La grande limitazione data dall'applicazione del framework Scrum, per questo progetto, nasce da uno dei capisaldi di questa metodologia ovvero la creazione di intervalli di lavoro di durata ben definita, in questo caso di due settimane, dove il team dovrà svolgere una quantità fissa di attività entro le scadenze previste, stiamo parlando degli Sprint.

Gli Sprint sono essenziali per applicare al meglio le logiche Agile aiutando il team a organizzare il lavoro nel migliore dei modi in periodi ben definiti. Allo stesso tempo, permettono il rilascio periodico di software funzionante utile per mostrare al cliente qualcosa di utilizzabile da lui stesso, per poter poi ricevere indicazioni ulteriori sulle funzionalità successive che si aspetta di vedere alla conclusione del prossimo Sprint. Lo Sprint è quindi anche utilizzato come uno strumento finalizzato alla raccolta del requisito, un vero e proprio strumento di progettazione dell'output finale.

Detto ciò, trovandoci di fronte ad un progetto dove i requisiti sono già ben definiti, risulta non essenziale mostrare degli avanzamenti periodici, per poter ricevere dei feedback utili da parte del cliente sugli step successivi da fare per il completamento del progetto. Tutte le US e i rispettivi Task contengono già tutti i requisiti necessari per ottenere un prodotto in linea con le richieste del cliente, con l'ulteriore possibilità di approfondire i vari temi grazie all'aiuto del Tech A e del PO che conoscono bene il contesto, in quanto hanno già lavorato alla creazione della vecchia versione dello IOC.

Tutto ciò, per poter sottolineare come il dover rispettare delle scadenze ben definite, per il completamento di quanto pianificato, è risultato essere un obbligo evitabile che ha apportato al gruppo apprensione di non completare le attività assegnategli. Gli effetti delle scadenze da dover rispettare hanno impattato negativamente le performance del Dev Team per tutta la durata del progetto, soprattutto nella parte iniziale, quando oltre alle scadenze imposte si aggravavano difficoltà relative alla collaborazione all'interno del team, alla

valutazione dell'effort necessario per il completamento delle US e alla comprensione del contesto tecnico-funzionale, difficoltà poi superate grazie all'esperienza maturata nel proseguo del progetto.

Lavorare con delle scadenze così stringenti non ha aiutato un team giovane e inesperto come quello preso in analisi a performare al meglio. Una soluzione possibile per cercare di mantenere la logica per Sprint sarebbe stata quella di allungare la durata di ogni Sprint ad almeno un mese, questo avrebbe però comportato un conseguente aumento di US messe in pianificazione, non risolvendo di fatto del tutto il problema.

Nel paragrafo successivo si vuole proporre una soluzione che prevede un cambio di approccio che si distacca in parte dalle logiche Agile ampiamente descritte, ma che a fronte delle analisi effettuate, potrebbe risultare più adatto al progetto di migrazione dello IOC.

6.2 L'approccio alternativo

La soluzione alternativa pensata è maturata durante lo svolgimento del progetto, applicando le nozioni teoriche presenti in letteratura e ciò che ho appreso durante la mia esperienza di tirocinio come Scrum Master. Tale proposta punta a massimizzare le performance del team di sviluppo in relazione alle principali difficoltà emerse e al contesto in cui si è trovato ad operare.

Partiamo col dire che il nuovo approccio proposto può essere associato ad una vera e propria soluzione ibrida tra il metodo Agile e il metodo tradizionale, applicando delle modifiche al classico framework Scrum utilizzato per questo progetto.

L'approccio si basa sull'esecuzione di tre fasi preliminari da dover eseguire in serie, ogni fase corrisponde ad una attività che il team dovrà svolgere insieme e nel minor tempo possibile. Una volta concluse le tre fasi che chiameremo *fasi di pianificazione*, il Dev Team potrà cominciare a lavorare alle varie User Story. La struttura del team rimarrà la medesima vista durante il progetto: lo Scrum Master manterrà i suoi poteri e la sua influenza sul team monitorando l'avanzamento del progetto, il Product Owner continuerà a fornire supporto sull'analisi dei requisiti e il Dev Team si concentrerà sulle attività di stima e di implementazione. Le tra fasi di pianificazione consistono in:

- *1° fase:* Si inizia con una vera e propria analisi dei requisiti già descritti in tutte le User Story presenti in backlog. In questa fase il Dev Team esegue degli approfondimenti su tutte le feature che dovrà implementare con l'aiuto del PO che sarà centrale in questa fase. L'output derivante dalle analisi svolta dovrà costituire in una piena comprensione dei requisiti da parte dei membri del team, oltre che ad una bozza sulle soluzioni tecniche da poter mettere in atto per lo svolgimento con annessi rischi che possono emergere.
- *2° fase:* La seconda prevede la suddivisione delle US in Task e l'individuazione di eventuali correlazioni e precedenze tra le varie User Story analizzate, attività fondamentale per lo svolgimento della fase tre.
- *3° fase:* Risulta essere la fase più delicata, il team è chiamato a creare dei *pacchetti di attività* contenenti tutti i Task individuati. Questi pacchetti sono riconducibili a dei veri e propri Sprint con la differenza che tra un pacchetto e l'altro la durata può variare in base alle giornate stimate dal team per il completamento. Altra peculiarità di questi pacchetti sono le interdipendenze che dovranno avere, ogni pacchetto conterrà un ammontare di Task propedeutici ai Task del pacchetto successivo, obbligando così il team a completare le attività del pacchetto in lavorazione per passare a quello successivo.

Le tra fasi iniziali sono orientate alla preparazione del lavoro, cercando di concentrare l'attenzione su alcuni aspetti che si sono rivelati critici durante lo svolgimento del progetto, come l'approfondimento dei requisiti, l'analisi del rischio e la stima delle US. Emergono inoltre delle sostanziali differenze rispetto al framework Scrum. Infatti, non verranno più svolti degli Sprint con una durata prefissata di due settimane, che, come abbiamo visto, si sono rivelati non ottimali al progetto in questione, ma dei pacchetti di lavoro di durata variabile, stimati dal team stesso in base alle giornate necessarie per il completamento e non in effort necessario. Importante sottolineare che la stima, oltre a comprendere il tempo necessario per l'implementazione delle US, dovrà comprendere anche la fase di testing svolta sempre dal tester ufficiale nominato dallo Scrum Master. La dipendenza tra i vari pacchetti di lavoro costringe il team a completare tutte le attività del pacchetto precedente per poter passare al successivo generando così zero debito tecnico. Sarà fondamentale per loro completare tutte le attività nelle giornate stimate per non ritardare di troppo il progetto. Potendo dividere l'intero progetto in pacchetti di durata definita, si dà la possibilità allo Scrum Master di poter definire la data di completamento del progetto, che dovrà aggirarsi

attorno a fine dicembre. Detto ciò, tenendo conto dell'analisi del rischio fatta nella fase 1 e della possibile variazione dei membri del team in corso d'opera, lo Scrum Master potrà inserire, ove lo ritenesse opportuno, eventuali *buffer temporali* che permettono al team di avere tempo extra per il completamento del pacchetto, rimanendo in linea con la data di fine progetto prevista.

Fino a qui la gestione del progetto può essere associato ad un vero e proprio metodo a cascata, sia per la natura delle tre fasi di pianificazioni, sia per la possibilità di poter fornire una schedulazione del lavoro da svolgere a priori potendo anche definire una data di fine progetto. Tutto questo è possibile grazie alla possibilità di poter accedere fin da subito a tutti i requisiti essenziali per svolgere la migrazione. Per quello che riguarda l'organizzazione all'interno del team, questa continuerà ad essere gestita facendo affidamento alle tecniche Agile, in modo tale che vengano applicate le logiche Scrum di orientamento verso un unico risultato. Per permettere un confronto continuo sul lavoro svolto e sull'evoluzione delle dinamiche organizzative del team, è possibile effettuare delle sessioni di Retrospective e Review al completamento di ogni pacchetto di lavoro, nelle stesse modalità e tempi in cui vengono svolte col framework Scrum.

Elementi Waterfall	Elementi Agile
• Approfondimento e analisi di tutti i requisiti prima dell'inizio del progetto. • Individuazione di Task e interdipendenze tra essi. • Definizione di pacchetti di lavoro • Stima in giornate necessarie per il completamento • Definizione della durata del progetto • Esecuzione in serie	• Organizzazione del team attraverso le logiche Scrum • Esecuzione di Retrospective e Review al completamento di ogni pacchetto

Tabella 4. Elementi Waterfall e Agile dell'approccio alternativo

Conclusioni

In conclusione, l'applicazione della metodologia Agile, in particolare del framework Scrum, nel progetto di migrazione dello IOC ha portato a risultati significativi. La flessibilità e la trasparenza introdotte da Scrum hanno migliorato la comunicazione all'interno del team, consentendo una gestione più efficiente delle priorità del progetto. Tuttavia, i risultati indicano che, sebbene Scrum abbia facilitato il completamento del progetto con successo, potrebbe non essere l'approccio ottimale per progetti di migrazione software.

Le peculiarità dei progetti di migrazione, caratterizzati solitamente da requisiti già definiti ma che richiedono una profonda comprensione, una pianificazione accurata e una gestione dei rischi, potrebbero richiedere un adattamento delle pratiche Agile. L'agilità e il focus sulla consegna rapida potrebbero non sempre allinearsi con le esigenze specifiche di questi progetti, che richiedono una maggiore attenzione alla pianificazione dettagliata e alla comprensione approfondita del contesto tecnico.

L'analisi dei risultati ha portato a considerare l'adozione di un approccio ibrido, combinando elementi delle metodologie agili con quelle del modello tradizionale a cascata. Questo approccio mira a sfruttare i vantaggi della flessibilità offerta dalle metodologie agili, mantenendo al contempo una struttura più definita e controllata nelle fasi di analisi e di pianificazione del lavoro.

In conclusione, l'esperienza pratica ha evidenziato che non esiste una soluzione universale per tutti i progetti in ambito IT. La scelta tra un approccio completamente Agile, Waterfall o un approccio ibrido dovrebbe essere basata sulle caratteristiche specifiche del progetto e del team di sviluppo. L’elaborato in questione, oltre che ad analizzare gli effetti dello Scrum su un progetto di questo genere, spera di fornire un chiaro esempio di come sia importante analizzare nel dettaglio il contesto, per poi proporre l’approccio migliore finalizzato alla massimizzazione del risultato.

Bibliografia

Manifesto for Agile Software Development, 2001.

Cantamessa, M, Cobos, E., Rafele, C. (2017). Il project management. Un approccio sistemico alla gestione dei progetti. ISEDI.

Canty D, Agile for Project Managers, CRC Press, Stati Uniti, 2015.

Cobb C.G, The Project Manager's Guide to Mastering Agile: Principles and Practices for an Adaptive Approach, Wiley Publishing, Stati Uniti, 2015.

Corbucci D, Agile Project Management. Overview delle principali metodologie Agile, Agile Mindset e guida all'esame di certificazione PMI-ACP, FrancoAngeli, Milano, 2019.

De Marco A, Project Management Theory Book, electronic source, 2006

Falchi, Manufacturing execution system (Mes): cosa sono e perché il manifatturiero non può farne a meno, 2023.

Julia Martins, Cos'è Scrum? Che cos'è e perché funziona così bene, 2023.

Ken Schwaber e Jeff Sutherland, La Guida Definitiva a Scrum: Le Regole del Gioco, 2020.

Lean Management Academy, Agile Project Management: The Complete Step-byStep Guide to Learn Project Management from Beginning to End, 2019.

Marino A, International Project Management Standard: Guida all’acquisizione delle credenziali PMP e CAPM del PMI, 2018

Mosca, MES software: 10 piattaforme da scegliere nel 2023, 2023.

Protto S, Concetti e strumenti di Project Management, FrancoAngeli, Milano, 2009.

Sutherland J, Fare il doppio in metà tempo. Puntare al successo con il metodo Scrum, Rizzoli, Milano, 2015.

Schwaber K, Beedle M., Agile Software Development with Scrum, Prentice Hall, Stati Uniti, 2001.

Taiichi Ohno, Toyota Production System, Productivity Press, Stati Uniti, 2005.

Thompson, G, Agile Testing: A Mixed Approach to System Development and Testing: Parallel Agile and Waterfall Approach Streams within a Single Project, 2009

Wikipedia, DevOps, Wikimedia Fundation

Wikipedia, Waterfall Model, Wikimedia Fundation

Wikipedia, Scrum (software development), Wikimedia Foundation