Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE

Corso di Laurea in Informatica per il Management

Scrum e Kanban: framework e strumenti di supporto. Prospettive di applicazione nel progetto Agile methods for Agile working.

Relatore:
Chiar.mo Prof.
Davide Rossi

Correlatore:
Chiar.mo Prof.
Angelo Di Iorio

Presentata da:
Matteo Tamari

Sessione II
Anno Accademico 2019/2020

Ad Angela e alla mia famiglia

Non mi giudicate per i miei successi, ma per tutte quelle volte che sono caduto e sono riuscito a rialzarmi.

Nelson Mandela

Introduzione

L'ingegneria del software è l'insieme delle teorie, dei metodi e delle tecniche che si utilizzano nello sviluppo industriale del software. Fin dalla sua nascita l'ingegneria del software si è occupata anche del *processo* di sviluppo del software, questo con l'ausilio di metodologie e framework che sempre piú si adattano si adattano a soddisfare le richieste reali dei committenti, sia in termini di tempistiche sia in termini di qualità.

Lo sviluppo e l'uso di un sistema software coinvolgono molte persone ed è necessario tenere conto dei ruoli, delle esigenze e dei rapporti reciproci per ottenere un prodotto che risponda pienamente alle aspettative. Per questo l'ingegneria del software studia anche gli aspetti sociali e organizzativi sia dell'ambiente in cui viene sviluppato il software sia di quello in cui il software viene applicato.

L'ISO/IEC/IEEE 12207:2017[1] è uno standard internazionale che definisce e descrive il ciclo di vita del software, fornendo una linea guida alle industrie che si occupano dello sviluppo e della produzione. Questo standard contiene processi, attività e obiettivi che devono essere applicati sia durante l'acquisizione di un prodotto o servizio software sia durante la fornitura, sviluppo e manutenzione dello stesso. Possiamo evidenziare i seguenti passi come il *ciclo di vita di un software*:

- Analisi o Specifica
- Progettazione
- Implementazione
- Validazione
- Estensione ed Evoluzione

[1] ISO, *ISO/IEC/IEEE 12207:2017 Systems and software engineering - Software life cycle processes*, Switzerland, 2017, in `https://www.iso.org/standard/63712.html`

Nel corso del tempo sono nati diversi modelli di sviluppo software, partendo dal più comune *Waterfall*, strutturato in una sequenza di fasi, passando per il *Modello a spirale* che approccia ad un sistema iterativo, fino ad arrivare alle metodologie Agili con svariati modelli e framework, che hanno completamente rivoluzionato la produzione del software e la velocità con la quale questo avviene.

Essendo i modelli descritti differenti, è opportuno illustrarne per completezza le rispettive peculiarità:

Modello Waterfall o a Cascata[2] : il primo modello di ciclo di vita del software, strutturato in una sequenza di fasi, le quali prevedono che una fase non possa iniziare se prima non è terminata quella che la precede. Questo modello è stato ampiamente criticato per la sua rigidità e linearità e abbandonato per lo sviluppo software degli anni '80. Le fasi sono così composte:

- Analisi: pianificazione, analisi e specificazione dei requisiti
- Progettazione: progettazione e specificazione del sistema
- Implementazione: programmazione e test di modulo
- Test: integrazione di sistema, test di sistema e test di integrazione
- Esecuzione: rilascio, manutenzione, miglioramento

Modello a Spirale[3] : pensato e sviluppato da Barry Boehm nel 1988, il quale propose per la prima volta un ciclo *iterativo* con un approccio *risk-driven* ovvero basato sul rischio:

> *"The major distinguishing feature of the Spiral Model is that it creates a risk-driven approach to the software process."*[Boehm]

Ancora oggi utilizzato, questo modello consente di rappresentare diversi cicli di vita. I capisaldi sono:

- Pianificazione: valutazione obiettivi associati ai singoli passaggi dello sviluppo del software
- Analisi dei rischi: identificazione dei rischi significativi per l'avanzamento del progetto di sviluppo
- Sviluppo: sviluppo del software con identificazione di una strategia evolutiva del processo
- Verifica: verifica del risultato e pianificazione del ciclo successivo

Modelli Agili : nati con la pubblicazione del *manifesto Agile*[4], si riferiscono ad un insieme di metodi di sviluppo fondati su principi comuni, i quali propongono un approccio meno strutturato e più focalizzato sulla consegna al committente, in tempi brevi e con alta frequenza. Il risultato è la produzione di software tipicamente di alta qualità e funzionante. Si tratta di sistemi basati su un concetto di rilascio *incrementale - iterativo*.

Questi ultimi saranno per l'appunto oggetto di questa tesi.

Un interessante studio compiuto dalla State of Agile[5] (facente parte della CollabNet VersionOne), azienda che ogni anno produce un survey estremamente dettagliato sul mondo Agile, offre ai professionisti del settore una visione approfondita delle tendenze Agili, delle migliori pratiche e delle lezioni apprese per aiutarli a raggiungere il successo con le implementazioni che più si prestano all'impiego previsto.

Le analisi compiute dal survey sono tante e di svariato genere; per quello che interessa al tema della tesi possiamo estrarre cinque punti estremamente importanti e descrittivi. Essi ci aiutano ad inquadrare fin da subito la dimensione nella quale ci troviamo e capire l'importanza che oggi hanno i modelli Agili nelle industrie software:

- Il 95% dei rispondenti ha dichiarato che la loro organizzazione utilizza metodologie di sviluppo Agili.
- L'81% dei rispondenti ha dichiarato che l'organizzazione ha team Agili che non lavorano nello stesso luogo e sono distribuiti geograficamente.
- Il 58% dei rispondenti ha dichiarato che utilizza Scrum come framework di sviluppo.
- Il 71% dei rispondenti ha dichiarato che la ragione dell'adozione da parte della organizzazione di metodologie Agili è perché consentono di accelerare in modo consistente la consegna del prodotto al committente.
- Il 67% dei rispondenti ha dichiarato che utilizza Atlassian Jira come strumento a supporto dello sviluppo e il 78% di questi ha dichiarato che lo raccomanderebbe ad altri in base alla sua esperienza.

In base a quanto elencato viene naturale chiedersi quale sia il motivo di tutto questo successo e perché risulti così tanto funzionale.

[4]K. Schwaber, J. Sutherland, K. Beck e altri, *Manifesto for Agile Software Development*, USA, 2001, in `https://agilemanifesto.org/iso/it/manifesto.html`

[5]State Of Agile, *14th Annual State of Agile Report*, USA, 2020, in `https://explore.digital.ai/state-of-agile/14th-annual-state-of-agile-report`

Spiegarne le ragioni sarà oggetto particolarmente importante della tesi; tuttavia appare preliminarmente opportuno illustrare il quadro generale che sostanzia il modello, attraverso l'esame dei dodici principi enucleati nel manifesto *manifesto agile*, che così recita:

- La nostra massima priorità è soddisfare il cliente rilasciando software di valore, fin da subito e in maniera continua.
- Accogliamo i cambiamenti nei requisiti, anche a stadi avanzati dello sviluppo. I processi Agili sfruttano il cambiamento a favore del vantaggio competitivo del cliente.
- Consegnamo frequentemente software funzionante, con cadenza variabile da un paio di settimane a un paio di mesi, preferendo i periodi brevi.
- Committenti e sviluppatori devono lavorare insieme quotidianamente per tutta la durata del progetto.
- Fondiamo i progetti su individui motivati. Diamo loro l'ambiente e il supporto di cui hanno bisogno e confidiamo nella loro capacità di portare il lavoro a termine
- Una conversazione faccia a faccia è il modo più efficiente e più efficace per comunicare con il team e all'interno del team.
- Il software funzionante è il principale metro di misura di progresso.
- I processi Agili promuovono uno sviluppo sostenibile. Gli sponsor, gli sviluppatori e gli utenti dovrebbero essere in grado di mantenere indefinitamente un ritmo costante.
- La continua attenzione all'eccellenza tecnica e alla buona progettazione esaltano l'agilità.
- La semplicità - l'arte di massimizzare la quantità di lavoro non svolto - è essenziale.
- Le architetture, i requisiti e la progettazione migliori emergono da team che si auto-organizzano.
- A intervalli regolari il team riflette su come diventare più efficace, dopodiché regola e adatta il proprio comportamento di conseguenza.

Di rilevante importanza è come questi dodici principi (e quindi lo stesso manifesto) siano completamente rivoluzionari rispetto alla corrente di pensiero che sta dietro alle metodologie tradizionali, come ad esempio il *modello a Cascata*. Esso, fino a prima del 2001, era quello di riferimento e basa la sua struttura non solo attraverso delle fasi scalari organizzate, ma anche sotto forma di gerarchie organizzative che non permettono una comunicazione reale sia col top management sia con il committente stesso.

A fronte di questo è evidente che nel *modello Agile* non vi siano riferimenti a strutture gerarchiche o organizzazione complessa interna (cosa che nel modello a cascata era ben chiarita e disposta), ma bensì vi sia una tendenza totale alla soddisfazione del cliente o committente a scapito di gerarchie e complessità, attraverso una rapida risposta al cambiamento laddove questo sia necessario, con una enfasi alla collaborazione e comunicazione da parte di tutte le parti interessate al successo del progetto in tutta la fase di sviluppo, dall'inizio alla fine.

> *"Agility is the ability to adapt and respond to change ... agile organizations view change as an opportunity, not a threat."*[Jim Highsmith]

Queste metodologie che abbracciano le persone prima degli strumenti, la comunicazione prima di una documentazione esaustiva, il rilascio incrementale piuttosto che uno unico alla fine, hanno concesso alle organizzazioni di migliorare non solo la produzione, ma anche il benessere interno relativo ai collaboratori (di qualsiasi genere e occupazione) e la comunicazione con il committente; egli non si trova più alla consegna con un prodotto superato e non desiderato, ma viene coinvolto nello sviluppo e collabora per valutare eventuali cambiamenti sia di mercato sia di scope[6], e correggere la rotta immediatamente nella iterazione successiva.

Tutto ciò, al contrario di quello che si può pensare, non significa non documentare e non avere regole che gestiscano lo sviluppo e la collaborazione, ma bensì a produrre documentazione più concisa e contenente le informazioni necessarie che potranno anche mutare nel tempo a seconda della variazione dello scope.

Nei prossimi capitoli analizzeremo due delle metodologie più conosciute e utilizzate nel mondo Agile, *Scrum* e *Kanban*, che presentano approcci differenziati e gli strumenti a supporto, con particolare riferimento ad Atlassian Jira, i quali offrono ai team un'esperienza che molto si avvicina a quella di lavorare all'interno dello stesso ufficio e sono oggi giorno di particolare importanza per consentire uno sviluppo continuo e collaborativo.

[6]Con *scope* si intende tutta e sola la quantità di lavoro necessaria a raggiungere lo scopo del progetto; si tratta di un'attività da eseguire, assieme ad altre, per concorrere alla realizzazione del risultato (cioè dello scopo).

Indice

Elenco delle figure

Capitolo 1

Scrum

Uno dei framework Agili che ha sicuramente riscosso più successo in tutto il mondo nello sviluppo software è Scrum.

Un po' di storia

Le origini di questo framework sono estremamente radicate.

Nel 1986 due noti economisti e professori di origine giapponese, Hirotaka Takeuchi *(Harvard Business School)*, e Ikujiro Nonaka *(Hitotsubashi University)* pubblicarono un articolo sulla Harvard Business Review intitolato *"The new new product development game"*[7].

L'articolo si strutturava con una prima analisi incentrata sulla strategia di produzione dei nuovi prodotti e, successivamente, con l'esame delle c.d. *best practices* di aziende di grande spicco come Toyota, 3M, IBM e Honda. Ne emerse che l'elemento accomunante tutti questi colossi era l'adozione di un processo di sviluppo sovrapposto, anche detto *overlapping development process*[8], piuttosto che il vecchio approccio sequenziale. I due economisti dunque paragonarono i team che vi lavoravano all'interno ad una squadra di Rugby che, come il gioco vuole, avanza passandosi la palla avanti e indietro utilizzando tattiche diverse a seconda dell'impiego. Questo passarsi la palla avanti e indietro può sembrare superfluo, ma è la chiave di volta per capire le dinamiche di Scrum e la filosofia di pensiero che vi soggiace.

[7] H. Takeuchi, I. Nonaka, *The New New Development Game*, in *Harvard Business School*, 1986, in `https://hbr.org/1986/01/the-new-new-product-development-game`

[8] Overlapping development process è un modello progettuale studiato per ridurre i tempi di completamento del progetto nello sviluppo del prodotto. L'efficacia differisce dalla capacità dell'organizzazione di risolvere l'incertezza nelle prime fasi del processo. I progetti traggono maggiori vantaggi dalla sovrapposizione se sono in grado di risolvere precocemente l'incertezza.

La nascita e l'evoluzione di Scrum

Nel 1995 Ken Schwaber e Jeff Sutherland presentano Scrum per la prima volta alla conferenza di OOPSLA ad Austin, Texas.

Siamo difronte ad un framework e non ad una metodologia, il quale si fonda sullo studio che abbiamo visto di Takeuchi e Nonaka. Il termine Scrum[9] è stato coniato dal Rugby ed il significato che i creatori volevano darne era proprio il raggruppamento degli elementi in un unico punto, compiendo uno sforzo collettivo per il raggiungimento dell'obiettivo.

L'idea espressa è quella di creare una squadra fortemente adattiva, che impara attraverso l'intelligenza sociale e la condivisione, riuscendo a risolvere problemi complessi e al tempo stesso rilasciare prodotti in modo efficace e creativo del più alto valore possibile.

Come recita la *Scrum Guide*[10]:

> "Scrum is not a process, technique, or definitive method. Rather, it is a framework within which you can employ various processes and techniques. Scrum makes clear the relative efficacy of your product management and work techniques so that you can continuously improve the product, the team, and the working environment."

Scrum quindi si concentra sulla teoria del controllo empirico dei processi, la quale afferma che la conoscenza deriva dalla esperienza e che le decisioni si basano su ciò che si conosce.

Questo framework utilizza un approccio *iterativo ed incrementale* per ottimizzare la prevedibilità ed il controllo del rischio.

Sono quindi 3 i pilastri che sostengono il controllo empirico:

- Trasparenza
- Ispezione
- Adattamento

Quando i valori come impegno, coraggio, concentrazione, apertura e rispetto sono incarnati in un team Scrum, allora i pilastri elencati prendono vita e creano fiducia per tutti.

[9] Con il termine Scrum si intende una situazione di gioco che si crea per ordine dell'arbitro - al fine riprendere il gioco quando esso è stato interrotto per qualche irregolarità - nella quale i giocatori si amalgamano formando un unico grande cerchio al fine di prendere possesso della palla

[10] K. Schwaber, J. Sutherland, *Scrum Guide*, USA, 2017, p. 3

1.1 Il modello e il cuore di Scrum

> *"I offer you Scrum, a most perplexing and paradoxical process for managing complex project."*[Ken Schwaber]

Scrum focalizza tutte le sue pratiche su un processo di tipo *iterativo ed incrementale* mostrato nella Figura 1.1.

Il cerchio inferiore rappresenta un'iterazione delle attività di sviluppo che si verificano l'una dopo l'altra, l'output di ogni iterazione rappresenta un incremento del prodotto, detto anche valore rilasciato. Il cerchio superiore, invece, rappresenta quella che definiamo ispezione giornaliera o meglio conosciuta come *Daily Scrum* che si verifica durante l'iterazione (tipicamente una volta al giorno),durante la quale i singoli membri del team si incontrano per ispezionare le reciproche attività e apportare gli adattamenti appropriati. Questo ciclo si ripete fino a quanto il progetto non è concluso.

Figura 1.1: Lo scheletro di Scrum

In linea generale e per chiarire subito difronte a che cosa ci troviamo, possiamo descrivere il funzionamento in questo modo: all'inizio della iterazione, il team rivede cosa deve fare. A questo punto seleziona ciò che ritiene di poter trasformare in un incremento di funzionalità potenzialmente consegnabili entro la fine della iterazione.

Alla fine della iterazione, il team presenta l'incremento delle funzionalità prodotte in modo tale che gli stakeholders[11] possano ispezionarne le funzionalità aggiuntive intro-

[11]Con il termine *stakeholders* si vuole descrivere individui o gruppi di individui portatori di interessi nel progetto che dipendono dall'organizzazione di riferimento per la realizzazione dei loro obiettivi: ad esempio, soci di maggioranza e minoranza, lavoratori dipendenti, collaboratori autonomi, clienti,

dotte ed eventualmente richiedere adattamenti tempestivi che potranno essere apportati anche alla iterazione successiva.

Diventa alquanto evidente che il cuore di Scrum stia proprio *nell'iterazione*.

Il team analizza ed esamina i requisiti e tenendo in considerazione la tecnologia e la conoscenza della quale dispone, valuta le proprie capacità. Questa valutazione è collettiva o di gruppo; infatti in Scrum, come vedremo, non esiste il singolo individuo, ma il team.

Una volta compiuta l'analisi, determinati gli strumenti e conoscenze, il team inizia l'implementazione, modificando ed adattando anche quotidianamente il suo approccio qualora incontri nuove complessità, difficoltà e sorprese. Le conoscenze che ogni membro possiede sono condivise ed è proprio grazie a questo che il team sa come deve operare e trova la strada migliore per farlo. Questo processo creativo e rivoluzionario è il cuore della produttività di Scrum, ovvero la base necessaria per comprendere a fondo tutto il processo che andremo a descrivere nei prossimi paragrafi.

1.2 Il processo

Un progetto in Scrum parte con una visione del sistema che deve essere sviluppata.

Inizialmente questa visione potrebbe presentarsi vaga e non cristallina ed è del tutto previsto e normale; con il passare del tempo e l'avanzamento del progetto essa diverrà chiara.

Per spiegare al meglio il processo che Scrum implementa andremo a descrivere brevemente in questo paragrafo anche ruoli e artefatti che in seguito definiremo nel dettaglio.

Partendo dal principio, in Scrum abbiamo una serie di artefatti e ruoli che supportano tutto il processo di sviluppo e aiutano ad aumentare la produttività e la velocità del team di sviluppo: il Product Backlog è uno di questi. Si tratta di una lista di requisiti con priorità che offrono una visione sulle feature da implementare; quando parliamo di priorità intendiamo dire che la lista ha un ordine di preferenza che si basa tipicamente sulle richieste ed esigenze che il committente presenta.

Il Product Owner si occupa della gestione del Procuct Backlog insieme agli stakeholders.

Il processo quindi inizia con una fase di raccolta dei requisiti iniziali e con una visione del progetto, che con ogni probabilità varierà nel futuro. Tutto parte dallo Sprint.

---

fornitori, parti sociali, finanziatori. Tipicamente esistono due tipologie di stakeholders: interni ed esterni. Dipendenti, azionisti, manager dell'azienda vengono identificati come stakeholders interni, mentre le istituzioni e i governi, i fornitori e i clienti, le associazioni di imprenditori, i sindacati e altri attori sociali che operano nelle comunità sono considerati stakeholders esterni.

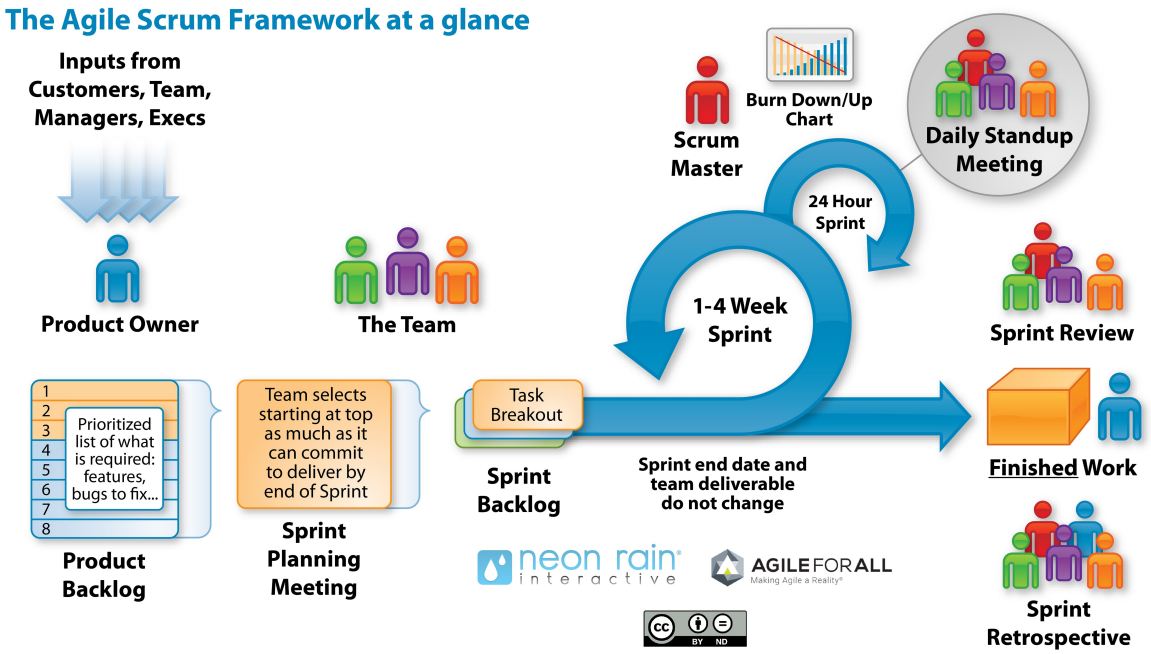


Figura 1.2: Il processo Scrum

Gli Sprint rappresentano iterazioni che variano dalle due alle quattro settimane (dipende dal tipo di progetto e dalla dimensione). Ogni Sprint ha il suo inizio con lo Sprint planning meeting, dove il Product Owner e il Team si ritrovano per decidere insieme cosa dovrà essere fatto nella Sprint successivo. Selezionando dal Product Backlog gli elementi con più alta priorità, il Product Owner comunica al team ciò che ha più urgenza di essere implementato nel prossimo Sprint, esigenze che vengono chiaramente dettate dagli stakeholders.

Una volta selezionati, gli elementi vengono spostati nello Sprint Backlog la cui funzione è per lo più transitoria poiché serve come appoggio in attesa della partenza dello Sprint. Concluso lo Sprint planning ha così inizio lo Sprint. Ogni giorno il team si ritrova per 15 minuti in un meeting detto Daily Scrum; in questo meeting ogni membro del team risponde a tre domande: "Che cosa hai fatto ieri?" " Che cosa farai oggi?" "Che impedimenti pensi di poter trovare?" L'obiettivo è cercare di sincronizzare il lavoro di tutto il team e pianificare eventuali meeting straordinari che si rendano necessari per compiere il lavoro. Questo viene ripetuto iterativamente per tutta la durata dello Sprint.

Quando uno Sprint si conclude, si tiene la Sprint review.

Si tratta di un meeting nel quale il team presenta il lavoro svolto e l'incremento prodotto agli stakeholders e al Product Owner. Una volta concluso il meeting, lo Scrum Master terrà la Sprint Retrospective con solo il team di sviluppo. Si tratta di un mo-

mento di condivisione nel quale tutti i membri del team espongono i propri pensieri, positivi o negativi, relativi allo Sprint trascorso. Per vedere il processo di Scrum si faccia riferimento alla Figura 1.2.

1.3 I ruoli

Come in ogni framework ben strutturato, i ruoli che si associano alle figure che prendono parte a Scrum sono ben definiti, con compiti chiari e precisi. Andremo quindi a descrivere le loro responsabilità e come si posizionano all'interno del framework.

Esistono tre ruoli fondamentali all'interno del framework Scrum:

- Product Owner
- Il Development Team
- Scrum Master

Tutti questi ruoli gestiscono e rispondono a responsabilità differenti.

1.3.1 Il Product Owner

Il *Product Owner* (che chiameremo PO) è il responsabile nel rappresentare gli interessi di tutti coloro che hanno commissionato il progetto; tipicamente si parla di stakeholders e committenti. Questi interessi si riflettono nel successo del progetto commissionato, rispettando valori e scope.

Tipicamente il PO si preoccupa anche di ottenere gli investimenti iniziali e continui per tutto l'avanzamento del progetto, creando gli stessi requisiti generali per il progetto basati sulla massimizzazione del *Return on Investiment*(ROI)[12]. La lista dei requisiti (iniziali e continui) è chiamata *Product Backlog*.

Il PO è il responsabile diretto di questa lista; il suo compito nella gestione di questo artefatto è di assicurarsi che le funzionalità con più alto valore siano prodotte o sviluppate nel più breve tempo possibile. Per tale ragione, uno dei suoi compiti fondamentali è riordinare continuamente, insieme ai committenti, la suddetta lista in modo tale da avere sempre all'apice tutte le funzionalità con il più alto valore possibile (per il committente) ad ogni iterazione che si compie.

La collaborazione con il team e le altre figure presenti sono un punto fondamentale per questo ruolo e, al fine di poter agire con successo, è altrettanto fondamentale che tutti rispettino le sue decisioni.

[12]Il *Return on Investment* (o ROI, tradotto come indice di redditività del capitale investito o ritorno sugli investimenti) è un indice di bilancio che indica la redditività e l'efficienza economica della gestione caratteristica a prescindere dalle fonti utilizzate: esprime, cioè, quanto rende il capitale investito di una determinata azienda

1.3.2 Il Development Team

Il *Development Team* è il diretto responsabile per lo sviluppo delle funzionalità richieste dai committenti e stakeholders.

Esso si compone di un numero di membri che tipicamente va da tre a nove persone; questo poiché con meno di tre persone il carico di lavoro risulterebbe eccessivo e le skills possedute potrebbero risultare insufficienti nello Sprint, mentre con più di nove persone la gestione del team risulterebbe altrettanto complessa e la produttività calerebbe drasticamente.

Un particolare che contraddistingue lo Scrum team da altri team di diversi modelli è la sua completa autonomia nella gestione e organizzazione. Ciò non vuole dire che non ci sia un controllo e verifica del lavoro svolto, ma semplicemente non esiste una figura che monitori assiduamente il lavoro svolto, contrariamente a quanto proposto dalle teorie manageriali sulla produzione di squadra proposte da Alchian e Demsetz[13].

Una seconda, ma non meno importante caratteristica da evidenziare, è che lo Scrum team è cross-funzionale, ovvero ogni membro del team è in grado di svolgere compiti differenti, che variano per esempio, dallo sviluppo allo unit test per esempio; non vi sono quindi figure strutturate e rigide che si occupano solo di un preciso compito.

Si tratta di una rivoluzione totale nel project management, ben pensata dai creatori del framework, al fine di ottenere un continuo assorbimento di conoscenza da parte di ogni membro. Così facendo, se un elemento per qualche ragione non dovesse essere presente, l'altro potrà svolgere il suo incarico.

Il trasferimento del know-how è fortemente radicato in Scrum.

Una terza e ultima peculiarità dello Scrum team è la responsabilità di capire come trasformare tutto ciò che è presente nel Product Backlog in un incremento di funzionalità all'interno di un'iterazione e di gestire il proprio tempo ed il proprio lavoro per farlo. Tutti i membri del team sono di fatto i responsabili per il successo della iterazione e del progetto stesso che si sta sviluppando; si tratta di una collettività, non di un singolo: si vince e si perde insieme.

[13]Secondo Alchian e Demsetz, l'impresa è un'entità che riunisce una squadra al fine di renderla più produttiva nel lavoro coeso. Per rendere lo sforzo della coesione non inutile, sottolineano l'importanza di avere una funzione di monitoraggio e controllo, ovvero una figura specializzata nel controllo degli individui impegnati nella produzione di squadra.

1.3.3 Lo Scrum Master

> *"Why did I choose a strange name like Scrum Master? Because I wanted to hightlight the extend to which the responsabilites are completly different from the traditional project manager"* [Ken Schwaber]

Lo Scrum Master non è un Project Manager.

Personalmente penso che ciò debba risultare ben chiaro: se nelle metodologie tradizionali (Waterfall) che abbiamo descritto sopra la figura che si occupa della gestione e ha la responsabilità di tutto il processo del progetto è il Project Manager, nel framework di Scrum questa figura non esiste.

Lo Scrum Master è un ruolo di grande importanza; di fatti è il responsabile del processo Scrum sopra descritto. Tipicamente possiamo dire che si tratta di un coach, che come in tutte le squadre, dove l'intesa nel gioco e l'allenatore sono decisivi per conseguire la vittoria, rappresenta l'elemento chiave che può fare la differenza . Uno dei suoi scopi è infatti diffondere la cultura e le pratiche di Scrum all'interno del team, con l'obiettivo di farne trarre vantaggio a tutti, dal committente allo sviluppatore, cercando di coinvolgere tutti i membri che partecipano all'evoluzione del progetto, costantemente.

Tra i suoi interessi vi è anche la continua collaborazione con il PO; l'obiettivo è cercare di aiutare quest'ultimo nella selezione delle features[14] di valore per il committente, tenendo conto anche l'interesse del team.

Questa figura ha una profonda conoscenza del framework Scrum: non può essere interpretata da persone qualsiasi, perché si tratta di un vero punto di riferimento all'interno del team, una linea guida, il volto della *Scrum Guide*. Il rispetto degli eventi e artefatti che può diffondere nel team sono di fondamentale importanza e non vanno lasciati al caso.

Lo Scrum Master è di fatti un *facilitatore*; non riceve nessuna medaglia o premio per quello che fa, in quanto si tratta di una figura che crede fermamente nel framework e nella sua diffusione.

[14] Con il termine *features* si intende l'insieme di tutte le caratteristiche che il software deve avere per essere considerato tale.

1.4 Gli artefatti

Una caratteristica di fondamentale importanza in Scrum sono sicuramente i suoi artefatti. In genere, in molte delle metodologie proposte per lo sviluppo software esistono svariati tipi di artefatti più o meno criticati.

Nel nostro contesto diventa fondamentale capire fino in fondo il loro impiego e le loro funzionalità a vantaggio non solo dello Scrum team, ma bensì di tutto il processo e di chi ne prende parte. Questi artefatti sono specificatamente progettati per massimizzare la trasparenza delle informazioni chiave.

Scrum introduce tre artefatti che vengono utilizzati durante tutto il processo precedentemente descritto:

- Product Backlog
- Sprint Backlog
- Incremento

1.4.1 Il Product Backlog

I requisiti del sistema e del prodotto che si va a sviluppare in un determinato progetto con l'ausilio di Scrum sono elencati nel *Product Backlog* (vedi Figura 1.3). Si tratta di una lista ordinata per valore, ovvero le funzionalità con priorità più alta saranno all'apice e quindi saranno quelle con la precedenza più alta di implementazione: tipicamente gli elementi in questa posizione sono più chiari e meglio dettagliati rispetto a quelli più in basso.

Quando parliamo di elementi ci riferiamo non sono alle user story[15], ma anche a feature, task, bug e a tutto quello che serve per migliorare il prodotto. Si tratta di un artefatto che non risulta mai completo, ovvero non vi è una fine all'inserimento di features fino a quando il progetto non termina, e viene aggiornato costantemente dal suo responsabile, il Product Owner, insieme ai committenti, stakeholders e all'aiuto dello Scrum Master. Esso è l'unica fonte di requisiti per le modifiche da apportare al prodotto.

Il Product Backlog, evolve insieme al prodotto che viene sviluppato e all'ambiente nel quale si trova; questi sono in continuo mutamento a causa di un mercato che si manifesta sempre più dinamico e altamente variabile in tempi ridotti. Pertanto i cambiamenti

[15]M. Rehkopf, *User stories*, in *Atlassian Agile Coach*, 2019, `https://www.atlassian.com/agile/project-management/user-stories`

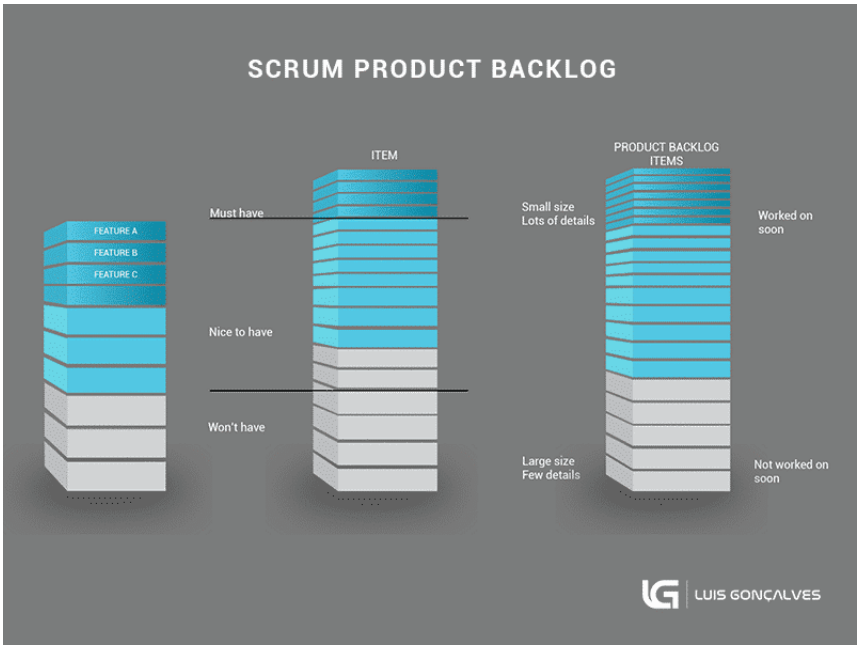


Figura 1.3: Il Product Backlog.

nei requisiti di business, nelle condizioni di mercato o nella tecnologia possono causare cambiamenti al Product Backlog.

Uno degli obiettivi del Product Owner è proprio la gestione di questo artefatto, il quale come detto evolve e cambia sia per via delle variazioni di mercato e quindi dello scope, sia per le richieste del committente che spesso variano all'avanzare del progetto. Lo scopo di questo artefatto è essere di aiuto e impatto, identificando di cosa il prodotto ha bisogno per risultare; appropriato al contesto, competitivo sul mercato e utile al suo scopo.

Fino a quanto il prodotto esisterà, il Product Backlog esisterà: per questo viene definito un *artefatto vivente*.

1.4.2 Sprint Backlog

Un altro artefatto vincolato al processo di Scrum è lo *Sprint Backlog*.

Il suo scopo è definire le user stories, features, task e bug che il Team ha selezionato e scelto dal Product Backlog come incremento di prodotto e raggiungimento dello Sprint Goal per quel determinato Sprint. In sostanza si tratta di una grande immagine, ad alta definizione e creata sul momento che identifica il lavoro che il team pianifica di svolgere e compiere durante lo Sprint successivo.

Per assicurare un miglioramento continuo, esso include almeno il miglioramento di un processo ad alta priorità.

Chi organizza e gestisce lo Sprint Backlog è il team, il quale può aggiungere e rimuovere oggetti durante il corso dello Sprint, non però al fine di modificare l'attuale Sprint - che ricordiamo essere una pratica vietata -, ma al fine di far emergere funzionalità utili che necessitano di essere implementate allo Sprint successivo. Questo sistema garantisce agli stakeholders che le funzionalità implementate saranno sempre le più utili e necessarie al fine di rilasciare un prodotto di qualità nel tempo, in modo iterativo e incrementale. In qualsiasi momento durante uno Sprint, il lavoro rimanente e totale all'interno dello Sprint Backlog può essere sommato.

Questo consente di tenere traccia della quantità rimanente di lavoro e di proiettare al team ad ogni Daily Scrum la probabilità di raggiungere lo Sprint Goal. Così facendo il Team sarà sempre in grado di gestire il proprio avanzamento nel progetto.

Burndown chart e Velocity report

Uno strumento a supporto, in particolar modo al Product Backlog, consiste nel *Burndown chart* (vedi Figura 1.4). Si tratta di un eccellente mezzo visivo che identifica il lavoro restante nel tempo, correlando tale lavoro al punto di progresso nel quale il team si trova in quel preciso momento.

Sull'asse verticale abbiamo il lavoro rimanente, tipicamente rappresentato da story point, che non sono altro che punteggi attribuiti alle varie user stories, task, features e bug.

Sull'asse orizzontale invece troviamo il tempo rimanente, che in Scrum rappresenta lo Sprint di qualsiasi durata esso sia. La linea grigia che vediamo nella Figura 1.4 rappresenta la *linea guida*: questa è estremamente utile in quanto aiuta il team a capire se il lavoro e lo sforzo che si sta impiegando saranno sufficienti per concludere tutte le features entro la chiusura dello Sprint.

Il Burndown chart è di fatti la collisione della realtà (lavoro svolto e quanto velocemente) con ciò che è pianificato o che si spera di raggiungere ed è questa è la sua reale forza. In un framework come Scrum, che è fortemente dinamico e reattivo al cambiamento, questo strumento riesce a dare una chiara idea di come lo Sprint stia proseguendo e fornisce risposte sulla produttività attuale.

Un secondo strumento di supporto meno utilizzato, ma sicuramente importante, è la *Velocity report* (vedi Figura1.5). Questo tool[16] si basa sugli story point esattamente come il Burndown chart; se il team completa tutte le user stories nello Sprint, la somma degli

[16]In informatica il termine *tool* descrive un piccolo programma di ausilio per attività specifiche, in genere fornito a corredo di pacchetti software.

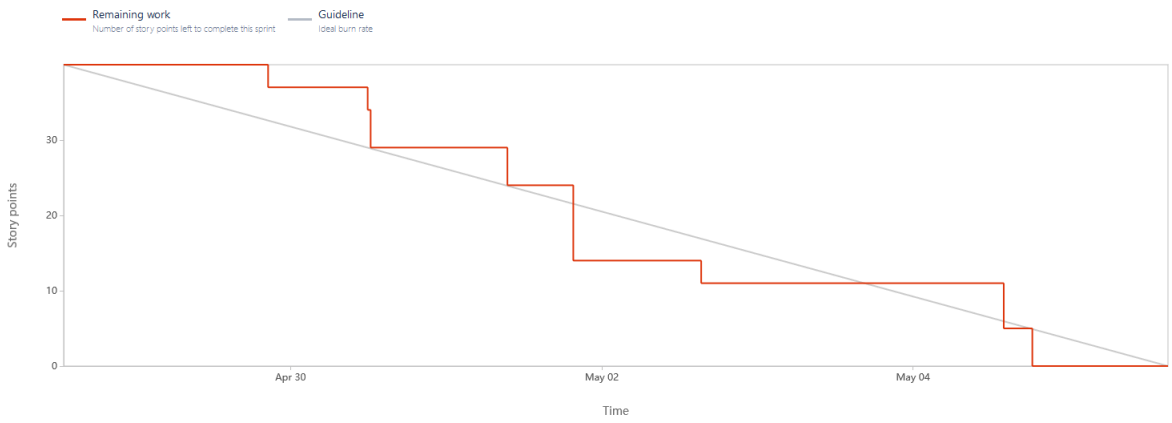


Figura 1.4: Il Burndown chart.

story point inseriti corrisponde alla velocity. In sostanza è uno strumento con il quale possiamo misurare la velocità nella produttività del team durante gli Sprint; permette al team di migliorare durante il flusso progettuale e fornisce precise indicazioni sulla tempistica di conclusione del progetto. Lo stesso *Jeff Sutherland* è molto puntuale nella documentazione rilasciata, in merito al progetto Sentinel descrivendo la Velocity report come segue:

> *"Il Velocity report non fornisce risultati immediati e non è una scienza esatta, datemi due mesi e vi dirò quando porteremo a temine Sentinel*[17]*"*

Con questa frase l'autore è stato decisamente chiaro. La velocity, prima di poter essere misurata, necessita di maturare all'interno del team; di fatti solo dopo qualche periodo si è in grado di provare che si è raggiunta una certa velocity, poiché il team ha maturato una costante in termini di story point che può affrontare all'interno di uno Sprint e sa fino a che punto può spingersi.

1.4.3 Incremento

Scrum richiede ai team di costruire e rilasciare un incremento funzionale del prodotto ad ogni Sprint. L'incremento è quindi la somma di tutti gli elementi del Prodcut Backlog completati durante uno Sprint e del valore degli incrementi di tutti gli Sprint precedenti. Questo deve essere potenzialmente erogabile al committente poiché è il Product Owner a scegliere il tipo di incremento di funzionalità che per l'appunto sono utili nell'immediato al committente.

[17] J. Sutherland, *Fare il doppio in metà tempo*, USA, Rizzoli, 2015, p. 13

Sprint	Commitment	Completed
SCRUM Sprint 2	29	29
SCRUM Sprint 3	40	37
SCRUM Sprint 4	33	33
SCRUM Sprint 5	39	28
SCRUM Sprint 6	40	40
SCRUM Sprint 7	52	52
SCRUM Sprint 8	30	30

Figura 1.5: Velocity report.

Quando parliamo di incremento, parliamo anche di *definition of done* o definizione di fatto: con questo termine indichiamo che i requisiti del prodotto rilasciato richiedono una fase di test ben strutturata, codice ben scritto e comprensibile che possa essere eseguito dal software e rispecchi le richieste del committente oltre che un valore aggiunto sull'utilizzo dell'utente finale.

L'incremento deve essere utilizzabile indipendentemente dal fatto che il Product Owner decida di rilasciarlo davvero.

1.5 Gli eventi

Fra i pilastri portanti del framework Scrum, che sorreggono tutto il processo dall'inizio alla fine, vi sono sicuramente i c.d *eventi Scrum*.

Questi eventi sono utilizzati per creare regolarità e ridurre al minimo la necessità di riunioni inutili, le quali generano un grande spreco di tempo che potrebbe essere impiegato meglio altrove.

Tutti gli eventi in Scrum hanno un periodo detto *time-boxed*, ovvero con una durata prefissata massima; questo garantisce quanto detto sopra, evitare perdite e spreco di tempo.

Oltre allo stesso Sprint, che un contenitore di altri eventi, ogni evento Scrum è un occasione di confronto e adattamento. Questi eventi nascono per creare trasparenza critica e innovazione; la mancata inclusione di uno qualsiasi di questi eventi riduce la trasparenza, l'occasione di confronto e innovazione.

Gli eventi Scrum sono così disposti:

- Sprint
- Sprint Plannig Meeting
- Daily Scrum
- Sprint Review
- Sprint Retrospective

1.5.1 Sprint

Il cuore di Scrum è lo Sprint. Si tratta di un periodo *time-boxed* che va dalle due alle quattro settimane (trenta giorni consecutivi); questo varia tipicamente a seconda della dimensione del progetto che ci troviamo ad affrontare. L'obiettivo è la creazione di un incremento di prodotto potenzialmente rilasciabile, utilizzabile e completato.

A partire da ciò, possiamo affermare che questo evento rappresenta il tempo totale richiesto al team per costruire qualcosa di significativamente interessante per il Product Owner e gli stakeholders.

Una nota sicuramente importante e spesso non tenuta realmente in considerazione è la durata costante nel tempo dello Sprint.

Uno Sprint ha una durata che rispecchia quanto sopra descritto; tuttavia ciò che va precisato è che questo periodo o intervallo scelto deve rimanere costante nel tempo. Ci

aspettiamo quindi che se un team sceglie un intervallo di due settimane per lo Sprint, lo mantenga per tutta la durata del progetto. La variazione di questo intervallo durante il corso del progetto è considerato un errore, che può anche rompere tutta l'armonia che Scrum cerca di creare.

Come l'intervallo dello Sprint deve essere rispettato, così anche il susseguirsi del successivo Sprint deve essere costante. Una volta concluso lo Sprint *n* si passerà allo Sprint *n + 1* immediatamente.

Come già scritto in precedenza, lo Sprint è un evento che contiene altri eventi. È quindi necessario descrivere questi ultimi al fine di chiarire la sua struttura. Essi sono: *Sprint Planning, Daily Scrum, Sprint Review e Sprint Retrospective* (che verranno approfonditi nei successivi paragrafi).

Scrum descrive nel dettaglio ogni suo evento e quindi anche cosa può accadere o meno durante uno Sprint, ovvero:

- Il Team può ricevere dall'esterno consigli, aiuti, informazioni e supporto durante tutto lo sprint.
- Non possono essere fatte modifiche che mettono a rischio lo Sprint Goal.
- Nessuno può dare ordini, istruzioni forzate, commenti, o direttive al Team durante lo Sprint: il Team si auto-gestisce.
- Gli obiettivi relativi alla qualità non devono degradarsi.
- Il Team preleva story, task, features dal Product Backlog durante lo Sprint Planning. Nessuno può modificare o cambiare il Product Backlog durante lo Sprint che rimane congelato fino alla fine dello Sprint attuale.
- Se lo Sprint per qualche ragione si rivela non praticabile, lo Scrum Master può in via del tutto eccezionale terminare lo Sprint e dar seguito ad un nuovo Sprint Planning per iniziare un nuovo Sprint. Lo Scrum Master si muove in accordo con il Team e il Product Owner. Questa situazione può accadere se la tecnologia è inutilizzabile o se le condizioni di business sono cambiate e quindi tale Sprint non rilascerebbe alcun valore al committente.

Possiamo considerare ogni Sprint come un progetto a sé stante delle durata prefissata inizialmente. Ogni sprint ha quindi un obiettivo di ciò che si andrà a costruire; questi obiettivi guideranno il lavoro e l'incremento di valore desiderato.

Una domanda spesso poco posta, ma lecita, è la seguente: perché gli Sprint hanno intervalli limitati?

La risposta è presto data. Supponiamo di avere uno Sprint senza limiti né intervalli; a questo punto qualcuno potrebbe pensare di avere un tempo infinito davanti a se e quindi sicuramente riuscire a concludere il progetto. Ma non è questa la filosofia sulla quale Scrum riposa.

I periodi hanno intervalli regolari e limitati per ragioni sia di business sia di valore per il committente. Se un orizzonte temporale è troppo lungo, la definizione di ciò che viene costruito può cambiare, la complessità può crescere e il rischio aumentare. Lo Sprint assicura agli stakeholders e ai committenti la prevedibilità e il rischio limitato di costi (ai giorni della durata dello Sprint) attraverso una ispezione del progresso del lavoro e una analisi dell'osservanza del valore finale che ci si attende. In questo modo, se qualcosa non sta funzionando, si cambia direzione allo Sprint successivo senza dover aspettare la conclusione del progetto con il rischio di aver fatto crescere un prodotto inutilizzabile poiché ormai fuori mercato.

Questa è la vera forza dei framework iterativi-incrementali.

Cancellare uno Sprint

Anche se è veramente difficile assistere a una tale situazione, uno Sprint può essere cancellato prima che finisca il suo tempo (time-boxed). Chi ha l'autorità per farlo è solo ed esclusivamente il Product Owner.

Questo caso è molto raro: significa che lo Sprint Goal è diventato obsoleto, le condizioni di mercato sono cambiate oppure la tecnologia utilizzata risulta superata. Se ci troviamo in questa situazione vi è stato un problema a valle nella fase di analisi, durante la quale, per qualche motivo, non sono state prese in considerazione certe dinamiche che hanno portato al fallimento dello Sprint.

1.5.2 Sprint Planning Meeting

Lo Sprint Planning è un evento che precede lo Sprint. Si tratta anche questo di un evento con intervallo di tempo time-boxed di otto ore al massimo (per Sprint brevi), tipicamente meno e consiste in due segmenti da quattro ore cadauna.

Nel primo segmento vengono selezionati gli elementi dal Product Backlog, mentre il secondo segmento serve per la preparazione dello Sprint Backlog.

Diventa evidente dire che il lavoro che si svolgerà nello Sprint verrà pianificato nello Sprint Planning; tuttavia è importante sottolineare che tale pianificazione è fatta grazie al lavoro collaborativo di tutto lo Scrum team.

Durante questo evento è possibile invitare personalità esterne al team, le quali possono dare qualche informazione, consulenza tecnica o di dominio, ma nulla oltre a questo.

"Non ci sono polli, ma solo osservatori"[18]

Sarà lo Scrum Master che si occuperà dello svolgimento dell'evento e si assicurerà che tutto il team ne abbia compreso le finalità.

Il Product Owner ha il compito di preparare il Product Bakclog a priori; in questo modo durante l'evento sarà pronto al suo utilizzo. In assenza del Product Owner o del Product Backlog pronto, sarà lo Scrum Master ad incaricarsi di costruire adeguatamente l'artefatto prima che il meeting abbia luogo.

Lo Sprint Planning risponde alle seguenti domande[19]:

- Quale è lo Sprint Goal?
- Cosa può essere consegnato nell'incremento risultante dallo Sprint imminente?
- Come sarà messo in opera il lavoro necessario a consegnare l'incremento?

L'obiettivo del primo segmento (quattro ore) dello Sprint Plannig è quello di selezionare gli elementi dal Product Backlog e cercare di convertirli in un incremento potenziale di funzionalità rilasciabili. Questo avviene assieme al Product Owner, che specifica sempre l'obiettivo dello Sprint ed evidenza quali elementi del Product Backlog potrebbero permettere di raggiungerlo. Salvo imprevisti, sarà il team di sviluppo che selezionerà gli elementi per il successivo Sprint, poiché soltanto il team di sviluppo è in grado di valutare quanto sarà in grado di compiere durante lo Sprint successivo.

Durante questo evento, tutto lo Scrum team modella il concetto di *Sprint Goal*. Si tratta di un obiettivo chiaro che si prevede di raggiungere durante lo Sprint imminente, fornendo così al team l'ideadi cosa si stia costruendo e l'incremento atteso.

Il secondo segmento dello Sprint Planning avviene immediatamente dopo il primo; durante questo lasso di tempo il team di sviluppo deciderà come trasformare le funzionalità degli elementi estratti dal Product Backlog in un incremento di valore del prodotto.

Tipicamente si inizia con la progettazione del sistema e del lavoro che sarà necessario per ultimarlo, il quale può essere di varie dimensioni e richiedere un effort[20] diverso.

[18] K. Schwaber, *Agile Project Management with Scrum*, USA, Microsoft Press, 2015, p. 9

[19] K. Schwaber, J. Sutherland, *Scrum...*, *cit.*, p. 13.

[20] L'espressione *effort* è un termine inglese che si traduce in *sforzo*, tipicamente utilizzato in ambito di project management per descrivere l'impegno necessario in termini di energie a compiere un determinato lavoro.

Questo permette al Team di pianificare tutta la quantità di lavoro e l'effort necessario in modo tale da non ricevere spiacevoli sorprese in fase di sviluppo o Sprint avviato e di prevedere ciò che si ritiene di poter fare o meno nel prossimo Sprint.

Il Product Owner tipicamente non partecipa a questo segmento, ma rimane a disposizione nel caso il team ritenesse necessarie delle delucidazioni in merito. Rimane chiaro e inteso che anche in questa fase il team è completamente auto-gestito.

L'output che ci si aspetta alla conclusione del secondo segmento è lo *Sprint Backlog* che servirà al team per iniziare lo sviluppo delle funzionalità previste per lo Sprint.

1.5.3 Daily Scrum Meeting

Chiamato dagli abituali del settore anche *stand up meeting*, si tratta di un evento time-boxed di quindici minuti, non uno di più, dedicato al team di sviluppo.

Questo evento ha luogo tutti i giorni, alla stessa ora, nello stesso luogo, per una volta al giorno durante tutto lo Sprint. Se può sembrare una ricorrenza inutile e per molti una perdita di tempo, tuttavia rappresenta un aspetto fondamentale per la buona riuscita del progetto.

Attraverso il Daily Scrum il team pianifica tutto il lavoro che avverrà nelle prossime ventiquattro ore. Il cuore di questo evento è ottimizzare la collaborazione tra gli sviluppatori e l'eventuale emergere di problematiche di sviluppo che si ha difficoltà o non si è riusciti proprio a risolvere. Il team di sviluppo utilizza l'evento per ispezionare anche l'avanzamento verso lo Sprint Goal e questo ottimizza la probabilità che raggiunga l'obiettivo.

Una peculiarità di questo evento è sicuramente la sua struttura: essendo dinamica, non vi è una regola precisa, l'importante è che si finalizzi il risultato ultimo ovvero l'avanzamento verso lo Sprint Goal. Se è vero che non vi è una linea guida rigida, è anche vero che vi sono più o meno frequenti e condivisi modi di affrontare il Daily Scrum; uno dei più conosciuti è porre tre domande specifiche ad ognuno dei membri[21] , ovvero:

- Cosa ho fatto ieri che ha contribuito a procedere verso lo Sprint Goal?
- Cosa farò oggi per aiutare il Team a procedere verso lo Sprint Goal?
- Vedo ostacoli che impediscono a me o al Team di procedere verso lo Sprint Goal?

I membri del Team dovrebbero rispondere a queste domande, uno alla volta, poiché non è ammesso parlare tutti quanti insieme. Questo può portare solo a miglioramenti

[21] K. Schwaber, J. Sutherland, *Scrum...* , *cit.*, p. 15.

e progressi al suo interno, aiutando così coloro che si trovino in difficoltà per qualsiasi ragione e condividendo inoltre il know-how.

Vi è da puntualizzare che è sconsigliato disperdersi in discussioni su problemi di carattere generale; è infatti facile che da un idea o una opinione scaturisca un dibattito che potrebbe far perdere il controllo del meeting, trasformandolo in una riunione tra amici.

Lo Scrum Master è il responsabile per questo evento; deve sincerarsi che quanto detto non avvenga. Se ci sono problemi diversi, Scrum assicura e chiede che siano risolti, tuttavia lo stand up non è il momento adatto e, qualora si verifichi un episodio simile, sarà lo Scrum Master a riportare l'ordine e organizzare un meeting per discuterne in separata sede. Se sono presenti attori esterni (in quanto previsto dal framework), questi non dovranno intervenire, saranno al più solo osservatori.

Il Daily Scrum migliora la comunicazione, elimina altri incontri inutili o che determinerebbero una perdita di tempo che si potrebbe impiegare nello sviluppo, identifica ostacoli e fornisce il supporto per superarli, oltre a promuovere il livello di conoscenza del team di sviluppo. Rappresenta un incontro chiave d'ispezione e adattamento.

1.5.4 Sprint Review Meeting

Alla fine di ogni Sprint si tiene la Sprint Review per ispezionare insieme al team, Product Owner, committenti e stakeholders l'incremento del valore realizzato e adattare eventualmente il Product Backlog. Anche questo evento, come tutti quelli in Scrum, ha un intervallo time-boxed da rispettare di quattro ore e si svolge ad ogni conclusione di uno Sprint.

L'obiettivo di questo evento è presentare ai principali portatori di interesse invitati le funzionalità che sono state implementate; questo si traduce nel valutare se quello che è stato realizzato rispecchia la definizione di *fatto* (definition of done) e può quindi essere potenzialmente rilasciato o implementato all'interno del Software. Tutto ciò che non rispecchia la definizione di fatto non può essere presentato. Tutti i partecipanti all'evento collaborano alle prossime implementazioni che potrebbero essere realizzate per ottimizzare il valore.

La presentazione delle features e funzionalità avviene tipicamente attraverso una workstation di un membro del team ed è eseguita su un server chiuso alla produzione, tipicamente uno dello sviluppo e qualità (QA).

In questa fase vengono anche discusse le difficoltà incontrate e come queste siano state risolte dal team. Inoltre lo stesso team si presta a rispondere alle domande, criticità,

osservazioni fatte da committenti e stakeholders relative a funzionalità che non sono state rilasciate o non sono state implementate come richieste e pensate. Tutto questo al fine di poter valutare le impressioni, desideri di cambiamento e priorità che magari sono variati nel tempo e non si rispecchiano più nello scope.

È importante precisare che la Review non è un momento per giudicare l'operato di nessuno; in Scrum ciò non deve avvenire, questo è un importante punto di forza del framework. Ciò spiega il motivo per il quale i membri del team tipicamente lavorano bene e meno stressati: nessuno li punta il dito accusatorio contro. Resta inteso che l'obiettivo è la soddisfazione totale del committente e va raggiunto, senza tuttavia far leva sul morale degli elementi.

Il risultato della Review è un Product Backlog che definisce gli elementi per il prossimo Sprint, revisionato in base ai feedback ricevuti nel corso dell'evento.

In tutto questo sarà lo Scrum Master ad organizzare il meeting, preoccupandosi di determinare il numero dei partecipanti attesi in modo tale da svolgerlo in posto appropriato e accomodante.

Alla fine dell'evento lo Scrum Master annuncia la data e il luogo della successiva Sprint Review al Product Owner e agli stakeholders.

1.5.5 Sprint Retrospective Meeting

L'ultimo evento che si tiene durante uno Sprint, ma non per questo meno importante, è la Sprint Retrospective.

Si tratta di un evento ad intervallo limitato (time-boxed) della durata massima di tre ore. Questo evento, a differenza di altri, è spesso a libera interpretazione, ovvero è previsto che venga svolto nel framework, ma il *quomodo* è a totale discrezione del team. Noi in questa sede ne descriveremo un tipo del quale abbiamo avuto una piccola esperienza durante un progetto nel corso di Ingegneria del Software.

In Scrum la Retrospective è un'occasione per il team di analizzare in retrospettiva lo Sprint appena concluso e quindi il lavoro che si è svolto, al fine di creare un piano di miglioramento comune da attuare al successivo Sprint.

Come dice il temine stesso il meeting ha luogo al termine di un'iterazione, tipicamente subito dopo la Sprint Review, e viene ripetuto per tutte quante le iterazioni fino alla conclusione del progetto.

L'obiettivo di questo evento è migliorare continuamente sia il processo sia l'esecuzione, tenendo in considerazione il feedback dato da ciascun membro del team, identificando nell'ultimo Sprint gli aspetti positivi e negativi in merito a relazioni, processi e strumenti.

È necessario sottolineare che non si tratta di un momento nel quale si giudica l'operato di qualcuno, come già scritto in precedenza, ma di un momento di miglioramento e confronto, dove non esistono giudici e giuria. Sarà compito dello Scrum Master assicurarsi che ciò non avvenga e che ogni membro partecipante all'evento abbia ben chiare le finalità in tal senso.

Ogni membro del team quindi, parlando sempre uno alla volta, esprime la propria opinione sullo Sprint e gli elementi che lo hanno caratterizzato, rispondendo a due semplici domande[22]:

- Cosa è andato bene durante lo Sprint?
- Cosa possiamo migliorare?

In questo modo, si riescono a ricavare molti caratteri comuni e utili a tutto il team, il quale può prendere spunto per lo Sprint successivo.

Spesso questo evento è una vera occasione per il miglioramento della produttività personale che si riflette su tutti gli elementi appartenenti al team negli Sprint seguenti. Resta inteso che, rispetto agli altri, questo incontro è decisamente meno formale; di fatti non vi partecipa nessuno se non il team di sviluppo e non sono previste presenze esterne come stakeholder o Product Owner. In altre parole, si tratta di un momento riservato al team nel quale esprimere anche opinioni per un miglioramento di carattere generale che non per forza sono circoscritte al progetto.

Entro la fine di questo evento tutto il team dovrebbe aver individuato i miglioramenti che implementerà al prossimo Sprint. Anche la Sprint Retrospective fornisce una opportunità formale per focalizzare l'ispezione e l'adattamento.

1.6 Implementazione e utilizzo di Scrum

Anche se può sembrare molto semplice, l'applicazione di questo framework non è affatto scontata; tuttavia se ben capito e successivamente applicato, può presentarsi come una chiave di volta. In questo paragrafo analizzeremo per l'appunto una sua possibile implementazione, focalizzandoci sul capire come e quando risultasse corretto l'utilizzo di questo framework piuttosto che altri.

L'analisi comparativa svolta è stata con Kanban (altro modello discusso successivamente nella tesi). Resta inteso che non si tratta di una scienza esatta; anzi, quanto

[22] K. Schwaber, J. Sutherland, *Scrum...*, *cit.*, p. 19

emerso fin qui ci fa proprio comprendere che non vi è una regola dettagliata e precisa e che occorre valutare caso per caso, eventualmente scalato all'occorrenza.

Partendo dalla natura della committenza, Scrum si presta meglio quando ci troviamo difronte a un progetto vincolato a requisiti che cambiano rapidamente e che richieda rilasci iterativi e incrementali. D'altronde spesso, se non il più delle volte, le condizioni volute e richieste dal committente non sono conosciute all'inizio del progetto; dunque ci troviamo difronte a una committenza che sa quello che vuole ma poi è vincolata a cambiamenti di business dettati da un mercato tipicamente molto dinamico.

Inoltre , si presta sicuramente bene ad un progetto che richieda un rilascio immediato e rinforzato mano a mano che si procede, dove tipicamente ci si trova a che fare con un consumatore che necessita di funzionalità nell'immediato e può attendere per altre; il "tutto e subito" non rispecchia decisamente la filosofia di Scrum ed è altamente sconsigliato per chi necessita di solidi prerequisiti per passare alla fase successiva.

Uno dei vantaggi nell'applicare questo framework sono sicuramente gli strumenti a supporto.

Oggi giorno infatti, in vista anche di quelli che sono i recenti avvenimenti in materia di sanità e contagi (Covid-19), gli strumenti software a disposizione degli Scrum team sono tanti, ben strutturati e molto specifici, tra i quali i più conosciuti risultano essere:

- Jira (Software o Confluence)
- BitBucket
- Zepel
- Zoho Sprints
- VivifyScrum
- nTask

Questi descritti sono tra i più conosciuti ed utilizzati, anche se ve ne sono altri e possiamo dire senza tema di smentita che Scrum è decisamente coperto sul mercato nel frangente attuale.

Alla luce di quanto detto, viene da porsi una domanda: quale è la giusta dimensione del progetto che si vuole affrontare?

Tipicamente Scrum è fortemente consigliato per progetti di medie dimensioni, tenendo in considerazione il periodo time-box dello Sprint di due o quattro settimane. Sicuramente va puntualizzato che, se si sta valutando il suo utilizzo per un progetto importante, è sconsigliato a team immaturi oppure a team che prevedono la conclusione in

meno di tre/ quattro settimane, in quanto risulta difficile, se non impossibile, analizzare la Velocity (vedi Figura 1.5) dopo solo uno o due Sprint.

Si tratta di un framework con eventi e ruoli fondamentali e la sua applicazione va ponderata secondo il tipo di progetto, l'esperienza del team e la dimensione nella quale ci si trova.

Scrum è applicato anche su larga scala (Scrum@Scale, SAFe[23] e LeSS) e scalato pure su imprese molto grandi che necessitano della collaborazione di più Scrum team per il medesimo progetto. Tuttavia l'applicazione non è sempre ben vista perché, seppur rispettando le dimensioni, si corrode la filosofia che sta dietro al framework e per la quale Jeff Sutherland e Ken Schwaber lo hanno ideato.

[23]Scaled Agile, *SAFe 5.0*, 2020, in `https://www.scaledagileframework.com/`

Capitolo 2

Kanban

Sebbene si sia certi del fatto che Kanban faccia parte del mondo *Agile*, spesso vi è molta confusione su come descriverlo e contestualizzarlo all'interno di questa dimensione.

È comune sentir paragonare questo modello con Scrum, ma la realtà è che non sono uguali.

Se Scrum è un *framework* ben strutturato, Kanban sicuramente non lo è. Per comprendere al meglio di che cosa stiamo parlando, è necessario fare due distinzioni:*Kanban* con la lettera maiuscola è un *modello* basato sulle metodologie Agili nel mondo dello sviluppo Software che, per certi versi, si avvicina più a una metodologia piuttosto che a un framework, mentre *kanban* con la lettera minuscola è la *metodologia* che identifica il processo in sé, creato non in ambito di sviluppo software.

Visto che vi sono tutt'ora dibattiti su come definirla noi, per logica di argomentazione nell'ambito dell'Ingegneria del Software, parleremo di Kanban il modello che si associa alle metodologie Agili per lo sviluppo del software. Come per il precedente (Scrum), prima di approfondire il modello Kanban necessitiamo di conoscere le sue origini per comprendere il pensiero e la filosofia che vi sono dietro.

Un po' di storia

Kanban nasce dall'approccio Toyota che negli anni '50 in Giappone fu portata dalla leadership di Taiichi Ohno, noto come il padre del *Toyota Production System* (c.d *Lean Production*) detta produzione snella, al massimo livello nel settore Automotive.

Questo ingegnere non solo condusse la Toyota a battere sul mercato i *competitors* Occidentali come Ford, sia per quantità prodotte sia in termini di difetti di produzione, ma rivoluzionò completamente il sistema di produzione, passando da una tipologia *push system* - nella quale vi era una spinta da monte verso la valle nella catena produttiva - ad

una tipologia *pull system*, nella quale l'attività a valle trascina e preleva quella a monte nella catena produttiva. L'occhio attento di Ohno portò la Toyota a massimizzare la produttività, riducendo il costo di produzione, progettazione e sviluppo senza annullare la qualità del prodotto e l'efficienza della catena di produzione.

Ohno non si limitò a correggere la mera catena produttiva; una seconda rivoluzione che compì all'interno dell'impresa fu sicuramente quella da lui chiamata *eliminazione dello spreco*. Si accorse infatti che, se il difetto nella parte meccanica veniva scoperto immediatamente e risolto, preveniva ulteriori difetti futuri: questo anche a costo di fermare la produzione. Nonostante l'idea di rendere reale l'incubo di ogni top manager, ovvero fermare la catena produttiva, Ohno dimostrò che il risultato ottenuto era straordinario e rifletteva una perdita economica decisamente inferiore rispetto al mandare avanti la produzione e risolvere successivamente il problema creato.

> *"Tutte le ispezioni condotte allo scopo di individuare difetti sono uno spreco e una perdita di tempo. Le ispezioni vanno fatte per prevenire i difetti."* [Taiichi Ohno]

La dimostrazione dell'eccezionalità di questa filosofia è dimostrata dal fatto che tale sistema è tutt'oggi presente in imprese come Lamborghini, Ferrari, Maserati, Fiat, MGM e altre ancora.

La nascita e l'evoluzione di Kanban nello sviluppo software

Con l'affermarsi delle metodologie Agili, nel 2007 David J. Anderson, dopo una lunga esperienza professionale, decide di riscrivere e sintetizzare i processi logici per rendere i team più performanti e indirizzando il metodo di kanban anche allo sviluppo software.

Basandosi proprio sulla linea di Taiichi Ohno, scrive e pubblica uno dei libri che hanno fatto la storia per l'utilizzo di Kanban nei modelli Agili[24].

Anderson prese l'idea di base di Ohno del processo di produzione di Toyota e la rese ancora più semplice e versatile, portando così un sistema di produzione di tipo *pull system* nel mondo dello sviluppo software, sfruttando la board[25] del sistema Toyota, per segnare il processo produttivo del software in maniera visiva così che tutti fossero sempre aggiornati, in qualsiasi momento e quando volessero, sul dove si era nel corso del progetto e se qualche impedimento o rallentamento si presentava.

[24] D. J. Anderson, *Kanban: Successful Evolutionary Change for Your Technology Business*, Washington, Blue Hole Press, 2010.

[25] Con il termine *board* si intende un bacheca o lavagna tipicamente utilizzata nel modello Kanban per il tracciamento delle feature da implementare, si veda M. Rehkopf , *What is a kanban board?*, USA, 2019, in `https://www.atlassian.com/agile/kanban/boards`

Il risultato è quello che oggi chiamiamo *il modello Kanban.*

2.1 Il modello Kanban

Nel software development Kanban è un modello per la gestione progettuale, il quale ha l'obiettivo di migliorare il flusso del lavoro, aumentare la produttività e garantire un rilascio continuo al committente chiamato in gergo *Continuos Delivery.*

Rifacendosi ad un sistema di tipo *pull system* e *work in progress*, rende da subito evidente lo stadio del processo in cui si trovano le singole attività, stimolando la collaborazione tra tutti i membri del team e attivandone la responsabilizzazione dei singoli; così facendo, il modello riesce a rendere sempre migliore lo sviluppo attraverso collaborazione e lavoro di squadra.

Concettualmente, come ogni framework, modello o metodologia Agile, anche Kanban basa tutto quanto sulla collaborazione tra i membri del team, affidando a loro la responsabilità del processo produttivo senza continue interferenze da parte del top management.

Questo tipo di propensione consente al modello di diventare uno strumento essenziale per poter creare una maturità crescente nel team di sviluppo, favorendone la distribuzione delle responsabilità e condivisione del know-how. L'insieme di questi elementi garantisce rilasci continui, di valore e qualità, rispettando sia tempi di consegna sia il budget assegnato dai committenti. Resta inteso che, come per Scrum, non si tratta di una scienza esatta e, se non vi è l'appoggio del top management, Kanban fallisce miseramente[26]. Nei prossimi paragrafi analizzeremo nel dettaglio tutto il processo di produzione proposto da Kanban e i vantaggi che questo comporta.

2.2 Il processo e i principi

2.2.1 Visualize Work

Il processo di questo modello inizia con la visualizzazione del lavoro da svolgere necessario per il completamento del progetto.

Uno degli strumenti che rende possibile la visualizzazione immediata del lavoro e consente di allineare tutto il team costantemente è la Kanban Board: strumento della quale discuteremo nei dettagli nei prossimi paragrafi.

[26]E. Brechner, *Agile Project Management with Kanban*, USA, Microsoft Press, 2015, p. 1

La creazione di una Kanban board è il primo passo quindi verso la visualizzazione del processo di sviluppo del software. Creando un modello visivo del flusso lavoro, diventa immediato per tutti i partecipanti visualizzare in quale punto ci si trova nel ciclo progettuale, osservando così l'intero flusso di lavoro che si muove. Questo si traduce in tre vantaggi immediatamente percepibili:

- Visualizzazione immediata di problematiche o blocchi progettuali
- Condivisione e suddivisione del lavoro ponderata e ben gestita
- Trasparenza a vantaggio di tutto il team (principio fondamentale nei modelli Agili)

La visualizzazione del lavoro in Kanban non significa visualizzare solo il processo produttivo, ma ogni singolo task, bug o lavoro mentre si muove letteralmente attraverso tale processo.

Così facendo, diventa comune vedere nei team che la condivisione e suddivisione del lavoro porta immediatamente ad una maggiore collaborazione e comunicazione, che si traducono in un lavoro di qualità dettato da un minore stress fisico, psicologico ed emotivo che lo sviluppatore deve sopportare durante il progetto. Questa visualizzazione immediata da parte di tutto il team del workflow[27] aiuta a stabilire le responsabilità e la trasparenza all'interno del team stesso; ciò comporta un grosso vantaggio soprattutto nell'ambito software, dove spesso sono richiesti sforzi coordinati da parte degli sviluppatori per svolgere il proprio lavoro in modo efficiente.

[27] Con il termine *workflow*, che tradotto dall'inglese si traduce in *flusso di lavoro*, si intende un flusso di lavoro costituito da un modello orchestrato e ripetibile di attività di business abilitate dalla sistematica organizzazione delle risorse nei processi che trasformano i materiali, i servizi o le informazioni di processo. Esso può essere rappresentato come una sequenza di operazioni, il lavoro di una persona o di un gruppo, il lavoro di un'organizzazione di personale, oppure uno o più meccanismi semplici o complessi.

2.2.2 Stabilire il Minimum Viable Product (MVP)

Quando si riceve una commessa per lo sviluppo di un tool o software è tipico ricevere informazioni generiche su come il prodotto dovrà essere o quanto meno su quello che ci si aspetta di ricevere. Il dettaglio arriverà con la fase di sviluppo mano a mano che ci si accorge di cosa va e cosa non va.

Durante la fase di sviluppo il team ha tipicamente le idee chiare di come sarà il workflow e la produzione, di ciò che verrà sviluppato prima, come ad esempio una feature molto importante per il committente, come anche di ciò che ha meno urgenza ed è più un dettaglio implementativo non richiesto nell'immediato.

Se in Scrum avevamo un ciclo iterativo incrementale (Sprint), il quale prendeva determinati elementi in ordine di importanza di produzione dal Product Backlog e li portava allo Sprint Backlog per essere prodotti, in Kanban abbiamo il *Minimum Viable Product (MVP).*

L'MVP è l'insieme degli elementi (task, features, user stories, bugs e altri) presenti nel Product Backlog che devono essere completati prima del successivo rilascio[28]. Quando parliamo di MVP non stiamo assolutamente dicendo che tutto ciò che si trova all'interno del Product Backlog è da considerarsi tale, ma anzi tipicamente, gli elementi che vengono identificati come rientranti nell'MVP sono una piccola percentuale del totale presente nel Product Backlog.

Si tratta di features che servono assolutamente per poter rilasciare valore al committente nella release successiva; parliamo di funzionalità di base del software che si sta sviluppando e che il committente si aspetta di avere prima che qualcuno provi il loro prodotto.

Stabilire dunque l'MVP diventa una parte fondamentale per il corretto funzionamento di Kanban, poiché questo ci garantisce una via eccellente per rispettare le deadline[29] ed evitare di produrre qualcosa che non sia di immediato valore per il committente.

2.2.3 Impostare il Work in Progress limit (WIP)

I team di sviluppo software spesso possono trovarsi sopraffatti dal grande carico di lavoro da gestire, come nuove funzionalità, correzione di bug, implementazioni richieste

[28] E. Brechner, *Agile Project Management with Kanban*, USA, Microsoft Press, 2015, p. 27 ss

[29] Con il termine *deadline*, che tradotto dall'inglese significa *scadenza*, si intendono tutte quelle scadenze progettuali che sono imposte in termine di tempistiche di produzione e rilascio per una determinata e prefissata parte del software. Tipicamente, se si tratta di una azienda di consulenza che sviluppa il software, queste scadenze vengono indicate nel contratto di progetto.

nell'immediato. Possiamo identificare nelle metodologie di Project Management l'essenza di come limitare il *caos* che si può generare dal carico di lavoro; è un punto fondamentale in ogni modello o framework sia Agile sia non Agile ed è altrettanto importante visualizzare come avviene la gestione.

Facendo un breve confronto tra i framework e modelli fino ad ora visti:

- Nell'approccio tradizionale Waterfall, il caos è limitato specificando tutto il lavoro in anticipo e sincronizzando i team in corrispondenza di traguardi prestabiliti.
- In Scrum, il caos è limitato pianificando il lavoro in un periodo time-boxed detto Sprint, durante il quale non è possibile modificare la pianificazione di sviluppo fino a quello successivo, sincronizzando così il committente con lo sviluppo.
- In Kanban, il caos è limitato attraverso la limitazione del carico di lavoro attuale o work in progress limit, che si può gestire nello stesso momento. Si tratta quindi di limitare le cards che sono presenti in ogni colonna della Kanban board ad un numero prescelto.

Il work in progress limit ha due strategie fondamentali attraverso i quale controlla il caos. Primo, limita il carico totale di lavoro che può essere affetto da cambiamenti di priorità, requisiti, o design e questo assicura che il team resti produttivo e riesca a raggiungere gli obiettivi. Secondo, limita il workflow in modo tale che corrisponda al ritmo più lento, poiché avere qualcuno che va molto più veloce può creare blocchi non indifferenti.

2.2.4 Continuos Delivery

Nei modelli e framework Agili esistono diverse pianificazioni di rilascio del valore verso il committente. Se in Scrum abbiamo visto un rilascio iterativo incrementale, in Kanban tutto cambia.

In questo modello infatti, il rilascio è *continuativo* e per questo viene definito *Continuos Delivery*, ovvero consegna continuativa del valore al committente.

La Continuos Delivery è una pratica introdotta nello sviluppo software per eseguire contemporaneamente le fasi di sviluppo, rilascio, feedback e gestione della qualità in brevi intervalli in un ciclo continuo. Lo sviluppo in questo modo diventa più efficiente e il cliente riceve prima il prodotto, anche quando questo non è ancora pronto. La Continuous Delivery fornisce feedback allo sviluppatore sulla base di test automatizzati che controllano principalmente la build ogniqualvolta viene modificato il codice sorgente. La

mission non cambia; la filosofia di pensiero, rivoluzionaria rispetto a quella tradizionale Waterfall, è quella Agile. Come in Scrum si tenta di rilasciare nel più breve tempo possibile valore, ma in modo continuativo; così facendo è possibile ricevere feedback quanto più veritieri e immediati per procedere a modifiche o correzioni.

La grande differenza è dunque data dal fatto che in Kanban con il Continuos Delivery si ha un rilascio continuativo; ipoteticamente possiamo assumere che è possibile rilasciare giornalmente piccoli incrementi del prodotto che si sta sviluppando.

Questo comporta diversi vantaggi, quali[30]:

- Sforzi collegati ad una release di grandi dimensioni vengono meno. Lo sviluppatore si concentra sul mero sviluppo del prodotto.
- Le release veloci e frequenti accelerano i feedback e i miglioramenti.
- Essendo un test continuo, i costi sono ridotti perché non si entra mai in fase di alpha e beta test[31].
- Lo sviluppo del software avviene velocemente, dato che lo sviluppatore sarà sostituito in gran parte dal processo automatizzato di release, prevedendo meno pause.
- Lo sviluppatore subisce meno pressione in fase di cambiamento al codice sorgente, perché gli errori vengono trovati velocemente. Ciò influisce positivamente sul lavoro, rendendolo più motivante e portando ispirazione.

Naturalmente, come tutte le attività e scelte implementative, la Continuos Delivery porta anche svantaggi, quali[32]:

- Costi per la gestione e costruzioni di server di integrazione per effettuare test automatici in velocità e affidabilità.
- I test automatizzati devono funzionare perfettamente.

[30] Inos, *Continuous Delivery: sviluppo di software tramite pipeline*, 2019, in `https://www.ionos.it/digitalguide/siti-web/programmazione-del-sito-web/continuous-delivery/`

[31] Con i termini *alpha e beta* test, si intende l'insieme dei test che si devono effettuare prima di poter dichiarare la feature sviluppata, funzionante e pronta alla consegna. L'alpha test tipicamente viene effettuato da sviluppatori che hanno partecipato allo sviluppo della feature, mentre il beta test viene erogato a un numero selezionato e limitato di elementi esterni allo sviluppo per farne verificarne l'efficienza anche a terzi. Generalmente l'impiego di questo sistema richiede uno sforzo aggiuntivo e una conoscenza pratica dello svolgimento non banale.

[32] *Ibidem*

- Sintonia del team per collaborazione a modifiche del codice in modo efficiente che avvengono tipicamente di frequente.
- Committente che si aspetta rilasci continui e quindi necessità di avere copertura di sviluppo costante; questo richiede una gestione non indifferente delle risorse di sviluppo.

In buona sostanza, come ogni modello o framework, Kanban ha i suoi vantaggi e svantaggi e va calato in accordo alle necessità dell'impresa, al tipo di committente e stakeholders, con i quale il team si trova a collaborare. Una cosa è certa: il Continuos Delivery non è per tutti. Bisogna avere un team organizzato, fortemente collaborativo e, se si ottengono questi strumenti, diventa una pratica estremamente efficiente sia in termini di risparmio di risorse economiche sia di soddisfazione finale del committente sul prodotto rilasciato.

2.2.5 Definition of Done

È importante dare una chiara definizione di cosa sia il completamento di un determinato task; così facendo risulta evidente cosa sia completo e pronto per essere passato allo step successivo e cosa invece no.

Tipicamente è consigliato che un membro del team valuti se il task che è stato posto nella colonna Done (il termine verrà spiegato a breve, nel paragrafo dedicato alla Kanban board) lo sia veramente o no. Questo richiede una grande esperienza sia nello sviluppo sia nell'utilizzo del modello Kanban.

Come sottolinea E. Brechner[33], le regole per la *definition of done* vanno condivise e strutturate con tutto il team e applicate sulla board in modo tale che tutti ne abbiamo chiara la definizione.

2.3 I ruoli

A differenza di altri modelli Agili dove i ruoli sono fondamentali e ben strutturati, il modello Kanban non ha mai fornito di default ruoli specifici (cosa che Scrum fa chiaramente) ma piuttosto sono stati creati nel tempo con l'esperienza legata all'utilizzo di questo modello.

Nonostante non vi siano linee guida sui ruoli, è comune oggi giorno vedere aziende che implementano i seguenti ruoli strutturati all'interno dei loro team Kanban:

[33]E. Brechner, *Agile...*, *cit.*, p.13 ss

- Service Delivery Manager (SDM)
- Service Request Manager (SRM)
- Development Team

2.3.1 Il Service Delivery Manager (SDM)

Il Service Delivery Manager, noto anche come Flow Manager o Delivery Manager, è responsabile nel garantire un corretto funzionamento del workflow del progetto. Questo ruolo è emerso la prima volta nel 2005 quando Microsoft decise di iniziare ad utilizzare Kanban.

Nella sua concezione iniziale l'SDM non era altro che il Project Manager; successivamente, sentendo la necessità di elevare la sua responsabilità verso il team e i committenti, è stato traslato per svolgere le necessarie funzioni di fornitura del servizio che ci si aspettava.

Visto lo studio che abbiamo affrontato, possiamo dire che l'SDM è l'omonimo dello Scrum Master in Scrum. Per visualizzare chiaramente le sue responsabilità e gli aspetti dei quali si occupa possiamo sintetizzare così i compiti che svolge[34]:

- Assicurarsi che gli elementi di lavoro fluiscano attraverso il sistema Kanban
- Facilitare le iniziative di miglioramento continuo e le revisioni dei processi
- Facilitare i meeting e assicurarsi che tutti vi partecipino
- Garantire che i processi funzionino per la massima soddisfazione del cliente

Non esiste una figura professionale precisa per questo ruolo e non vi sono certificazioni erogate da enti riconosciuti, come ad esempio per lo Scrum Master (Scrum Alliance).

Vi sono diverse scelte su chi eleggere con il titolo di SDM: questi opzioni includono un membro del team di un determinato progetto o l'acquisizione di una figura nuova, reperita anche sul mercato che abbia esperienze in questo ruolo particolare. Di estrema importanza è assicurarsi che la persona prescelta già abbia conseguito una elevata capacità e maturità nella figura e sia in grado di soddisfare le aspettative del top management e dei committenti.

La figura ideale non ha quindi problemi a sottolineare ritardi e colli di bottiglia nella fase di sviluppo, ha ottime capacità di relazionarsi con il team, i committenti, gli

[34]L. Boiser, *Kanban Roles for Successful Project Management*, 2019, in `https://kanbanzone.com/2019/kanban-roles-successful-project-management/`

stakeholder e inoltre cerca continuamente di migliorare il processo per garantire livelli costantemente elevati di soddisfazione.

Nel mercato dello sviluppo software attuale questa figura è diventata di grande importanza.

2.3.2 Il Service Request Manager (SRM)

Se l'SDM era l'omonimo dello Scrum Master, il Service Request Manager possiamo paragonarlo al Product Owner in Scrum; tuttavia bisogna prestare molta attenzione poiché i due ruoli, seppur simili, giocano su responsabilità diverse.

Lo scopo principale di questa figura è cercare di filtrare tutte le feature e implementazioni richieste in quello che realmente è di valore per il committente, per soddisfare tutte le aspettative del software che si sta sviluppando.

Il *modus operandi* di questa figura è cercare di riunire tutte le persone coinvolte nello sviluppo e assicurarsi che tutti abbiano ben chiaro il prodotto da sviluppare e quanto che per il committente è di valore, che spesso come abbiamo visto, potrebbe non rispecchiare le idee di uno sviluppatore.

Detto questo, è importante precisare un punto: l'SRM non decide per nessun motivo gli elementi sul quale il team si dovrà focalizzare ed è una grossa differenza dal Product Owner, che invece ha grande influenza sul Product backlog e sulle features prioritarie.

Il compito di questa figura è quindi facilitare la comunicazione, avviando la collaborazione tra committenti, stakeholders e team, portando le discussioni sul processo decisionale dello sviluppo del prodotto. Il Service Request Manager è responsabile del mantenimento del meccanismo che guida la governance e la direzione del prodotto. Ciò richiede la creazione e il mantenimento delle politiche di sistema, inclusa la valutazione e la gestione dei rischi che entrano in queste decisioni sui prodotti.

Secondo il modello Kanban a nessuno dovrebbe essere data la responsabilità di decidere da solo cosa sia o meno di valore per il committente; per questo l'SRM è differente rispetto ad Product Owner.

Nelle aziende moderne a volte questa figura viene sottovalutata o confusa: non sono nulli infatti i casi nei quali l'SDM svolge anche il ruolo di SRM, oppure l'SRM non è presente ed sostituito da Account Manager o Project Manager. Tendenzialmente mischiare, ruoli o non integrarli può portare a grossi problemi e disfunzioni interne, le quali possono a loro volta causare una produzione del prodotto non di qualità e insoddisfacente.

2.3.3 Development Team

Come in Scrum anche in Kanban abbiamo il team di sviluppo.

Il suo scopo è quello di sviluppare nel miglior modo e nel minor tempo possibile software di alta qualità. Tipicamente il team sviluppo include dai 3 ai 10 elementi, tutti con capacità cross-funzionali e self-organized poiché il lavoro viene scelto (basandosi sulle priorità) e non dato.

2.4 Gli artefatti

2.4.1 La Kanban Board

A differenza del framework Scrum precedentemente visto, che utilizza diversi artefatti, Kanban si focalizza principalmente su uno: la Kanban Board.

Abbiamo già parlato di questo potente strumento per la gestione del processo di Kanban e per quale utilizzo sia stato pensato da Taiichi Ohno.

Sebbene sia stato concepito per le aziende manifatturiere, ed in particolare per il Lean Manufacturing, grazie a David J. Anderson durante la fase di implementazione del modello Kanban nel mondo Agile, la board è stata assorbita volutamente e tradotta all'occorrenza con l'aumentare dell'esperienza per lo sviluppo software.

Ad oggi questo strumento non risulta presente solo in Kanban, ma è possibile ritrovarlo anche in altre metodologie Agili, come Scrum con la Scrum Board.

Nel software development la Kanban board è uno strumento di gestione dei progetti Agili progettato per aiutare a visualizzare il lavoro, limitare il lavoro in corso (WIP limit) e massimizzare l'efficienza (o workflow) di produzione.

Scegliere una posizione

È comune leggere o sentir dire che la posizione fisica della Kanban Board non sia fondamentale; la verità invece è che fa la differenza, come anche il fatto che il team sia raggruppato nella stessa stanza. Per chiarire questo punto cruciale facciamo un breve esempio: supponiamo di far parte di un team di 6-7 persone, le quali magari sono collocate in differenti uffici. Il primo problema è: dove posizioniamo la Kanban board?

Scegliere un ufficio piuttosto che un altro sarebbe in ogni caso improprio, perché quella posizione riceve visibilità ad una parte del team e non dall'altra; diventa quindi fondamentale in prima battuta radunare i componenti in un luogo comune di lavoro.

Una volta radunato il team emerge il secondo problema: dove posizionare all'interno della stanza la Kanban board?

Non è in realtà una domanda ozioso in quanto da una sua infelice collocazione potrebbero derivarne conseguenze nefaste sulla produttività del team, come per esempio avere alcuni membri non aggiornati costantemente con conseguente rallentamento del workflow anche dell'intero team stesso.

Per praticità dunque la Kanban Board va posizionata in un punto visivo nel quale tutti i membri del team ne abbiano l'immediata visualizzazione dalla loro esatta posizione. Lo stesso E. Brechner[35] spiega dettagliatamente che essi lavorano meglio e con notevole rapidità se sono localizzati vicino alla Kanban Board, poiché questo migliora la velocità e facilità di lavoro nonché la comunicazione tra i membri, oltre a far scaturire un senso di appartenenza e identità nell'impresa e nel progetto stesso, aprendo le porte quasi a un senso di patriottismo verso il progetto e la sua riuscita.

In questo modo diventa anche più facile per tutti gli stakeholders interessati al progetto visualizzare il workflow e percepire l'affiatamento del team, oltre al vantaggio nel caso in cui si dovessero confrontare con qualche membro del team e avere l'occasione di visualizzare direttamente la board nella medesima stanza.

La Kanban Board funziona in maniera ottimale anche se digitale, ma secondo E. Brechner[36] non comunque come la visualizzazione dal vivo. Vedremo nei successivi capitoli come vari strumenti a supporto implementano questa board, con particolare riferimento a Jira Software.

Costruire le colonne

Una volta scelto il giusto posizionamento, è fondamentale costruire le colonne.

Un notevole vantaggio del modello Kanban è il fatto che lascia grande apertura anche sulla strutturazione della board; di fatti ognuno potrebbe implementarla come meglio crede al fine di soddisfare le esigenze del team e della pianificazione stessa: da una semplice "To do, In progress, Done", a qualcosa di più articolato.

Vi sono tuttavia delle linee guida che sono fornite spesso da persone con grande esperienza in merito; in questa tesi seguiremo quella descritta da E. Brechner, IT Manager e Team Leader presso Microsoft[37], in quanto particolarmente completa.

Il primo passo per ottenere una buona board è sicuramente quello di assicurarsi che la larghezza delle colonne sia sufficiente per consentire di impilare oggetti come stiky

[35] E. Brechner, *Agile...*, *cit.*, p. 9.
[36] *Ibidem*
[37] *Ivi*, p. 8 ss

notes, cards o altro occorrente, secondo un principio di visualizzazione della stessa che nei paragrafi precedenti abbiamo sottolineato.

Il secondo passo è dividere le colonne. Secondo l'esperienza di E. Brechner alla fine il risultato deve essere come quello mostrato in Figura 2.1.

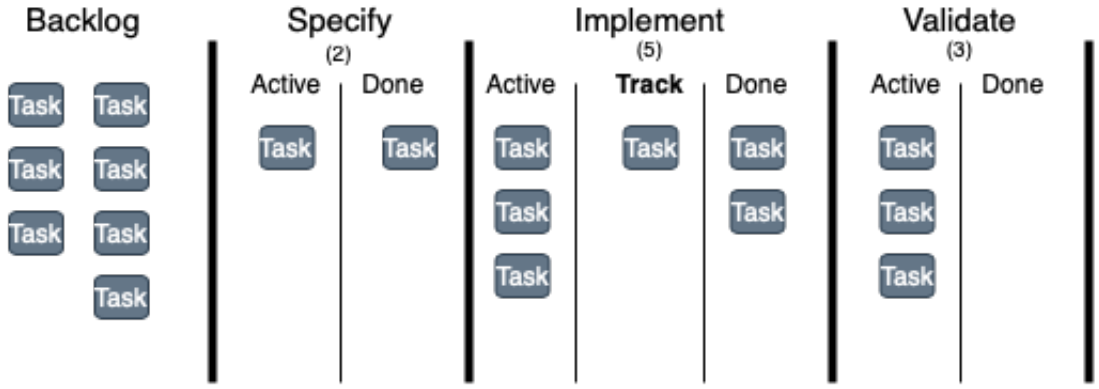


Figura 2.1: Kanban Board

Descrivendo il workflow e la suddivisione, la board presenta la colonna di *Backlog* dove vengono inseriti tutti i task, user stories, features e bugs che necessitano di essere implementati o corretti. Le cards in questa colonna sono posizionate nell'ordine nel quale vogliamo che siano svolti i task a seconda della priorità, dall'alto verso il basso. Quelle più alte sono quelle che il team deve pronto a lavorare appena possibile.

Nella colonna *Specify*, tipicamente un analista ha il compito di rompere le eventuali user stories e cercare di rendere tutto in task assumibili da un singolo componente. Viene a sua volta suddivisa in due sotto colonne, *Active* e *Done*, dove nella colonna di sinistra abbiamo le cards che sono pronte per essere lavorate e nella colonna di destra quelle che sono state lavorate e sono pronte per essere prese in consegna dallo step successivo attraverso il *pull system* precedentemente descritto.

La colonna *Implement* ha come compito l'effettiva implementazione dei task che il team andrà a sviluppare; tipicamente è in questo frangente che avvengono i blocchi più complessi da risolvere da parte del team e che ritardano il flusso progettuale (per questo motivo è stato creato il WIP precedentemente descritto). Anche questa colonna e suddivisa in tre sotto colonne: *Active* e *Done* rispecchiano le precedenti descritte, mentre *Track* ha una funzione di supporto, in quanto spesso ci si trova a sviluppare qualcosa che necessita di una approvazione o di una implementazione esterna e che non dipende dal team. Pertanto, per non bloccare il workflow segnato da un WIP, si collocano i suddetti task all'interno di questa colonna che non è presa in considerazione nel WIP stesso e, una volta che il task al suo interno viene sbloccato dall'evento esterno, si inserisce nuovamente

all'interno della colonna *Active* se non concluso o *Done* se concluso, riprendendo ad essere considerato nel WIP. Questi particolari task che attraversano la colonna *Track* hanno la precedenza sugli altri che stazionano sulla colonna *Active*.

L'ultima colonna è quella del *Validate*: in questa fase i tester prendono in consegna il prodotto lavorato dalla sotto colonna *Active* per testarlo e infine passarlo alla sotto colonna *Done*, che non viene conteggiata nel WIP della board.

Il WIP nella board presentata sono contrassegnati da numeri tra le parentesi disposti sotto il nome della colonna in oggetto; quei numeri rappresentano il limite superiore di task che possono stazionare nella colonna nello stesso momento.

L'implementazione presentata non è una regola, ma sicuramente la chiara espressione dell'esperienza di un professionista del settore con una inequivocabile idea di cosa sia questo artefatto e come vada utilizzato. Rimane importante calare queste esperienze e sfruttarle secondo le necessità di ogni team Kanban; come già sottolineato, non ci sono regole scritte come in Scrum, ma al contrario solo esperienze condivise e fatte proprie.

2.4.2 Comulative Flow Diagram

La descrizione della Kanban Board vista nei paragrafi precedenti ha sicuramente dimostrato di essere un ottimo strumento per la gestione del workflow di un team Kanban; tuttavia non è l'unico strumento che ci fornisce un supporto allo sviluppo.

La Comulative Flow Diagram (CFD) è un potente strumento per analizzare costantemente il workflow e le tendenze sulle prestazioni di un team.

Si tratta di un grafico come in Figura 2.2, il quale mostra l'andamento del team durante tutto il flusso progettuale.

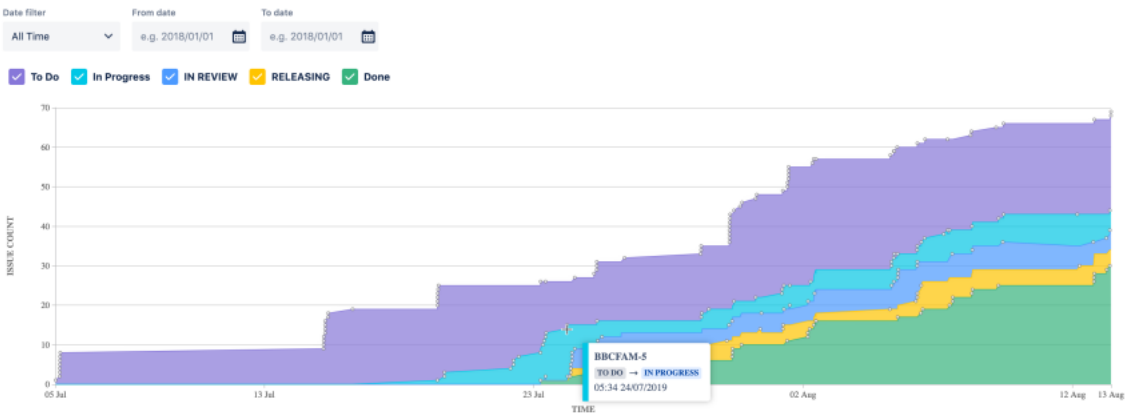


Figura 2.2: Comulative Flow Diagram

Se vogliamo utilizzare un metafora, possiamo dire che il CFD è come un narratore che racconta una storia disegnandola graficamente, in un preciso momento e intervallo di tempo.

Grazie a queste informazioni infatti, il team è in grado di diagnosticare eventuali problemi e migliorare il proprio processo per creare un workflow sempre più stabile e prevedibile.

Volendo una descrizione accurata e scientifica, il CFD è una rappresentazione grafica del lavoro mentre scorre durante la fase di sviluppo. Si tratta di un grafico che mostra sull'asse delle x un intervallo temporale, mentre su quello delle y tutte le cards, task e features presenti nel progetto, sia da sviluppare sia sviluppate.

Viene suddiviso in fasce colorate come da Figura 2.2, ognuna delle quali rappresenta una diversa colonna della Kanban board del team.

Come leggere il Comulative Flow Diagram

Può sembrare banale, ma la lettura di questo artefatto, oltre che complessa è altrettanto importante, e come già detto, è uno strumento che può fare la differenza anche nei rapporti con i committenti e stakeholders, le preoccupazioni dei quali vertono spesso su quando qualcosa verrà consegnato.

Partendo dalla parte superiore, la prima fascia tipicamente rappresenta tutti gli elementi alla quale il team deve lavorare e questo comprende task, features, user stories, bug e altro ancora.

La fascia inferiore invece rappresenta tutto ciò che il team ha concluso ed è stato posto a *Done* nella colonna *Validate*.

Le fasce centrali rappresentano tutto ciò al quale il team sta lavorando in quel preciso momento.

Il CDF dunque ci mostra quanti elementi sono passati da uno stato all'altro in un dato periodo.

Tipicamente la traiettoria del grafico dovrebbe essere costantemente crescente, con fasce che rimangono più o meno parallele anche in larghezza; questo è un ottimo segnale per un committente che visualizza il lavoro, in quanto significa che il team sta rilasciando valore, prodotto costantemente ad un ritmo stabile. In questa rappresentazione grafica l'unica eccezione consentita di ampliamento della fascia è quella bassa del *valore consegnato*, la quale tipicamente si ingrandisce anche in larghezza all'aumentare degli elementi che passano in stato *Done* nella colonna *Validate*.

Quello descritto è un flusso progettuale lineare quasi perfetto, ma non sempre è così nel mondo reale. È infatti spesso necessario aggiustare e porre dei rimedi di vario genere durante tutta la fase progettuale e il CFD fornisce sicuramente una visione immediata di quanto sia necessario predisporre.

Di seguito descriviamo modelli di risoluzione attuabili visualizzando e leggendo correttamente il CFD[38]:

- **Restringimento delle fasce**: indica che il lavoro è inferiore alla capacità del team. Potrebbe essere necessario allocare nuovamente i membri del team per bilanciare il carico di lavoro.
- **Ampliamento delle fasce**: indica un collo di bottiglia. Ciò significa che il tasso di entrata degli elementi in questo stato è più veloce del suo tasso di uscita. È necessario analizzare cosa sta causando il blocco e applicare un'azione correttiva.
- **La fascia *Active* è più ampia della banda *Done***: indica che il team non può completare il lavoro alla stessa velocità con cui viene aggiunto. Ciò può significare diverse aspetti che possono includere problemi di capacità e informazioni insufficienti sulle attività.
- **La pendenza è piatta**: indica che non è stato eseguito alcun lavoro. Non sempre è motivo di allarme, in quanto ciò può accadere durante le vacanze o in assenza di elementi del team nei giorni lavorativi. Ma se questo non è il caso, è necessario approfondire il motivo per cui non viene fatto nulla.

2.5 Gli eventi

Sebbene vi siano eventi, Kanban per come è stato pensato non ne è ricco né tanto meno forza a farli, al contrario di Scrum, per rispettare e assicurarsi un ottimo risultato dall'utilizzo di questo modello.

Tuttavia la stragrande maggioranza delle aziende che utilizzano questo modello sceglie di usare pochi, ma decisivi eventi, che possono fare la differenza sul risultato finale.

La scelta che viene fatta in Kanban è in linea con la sua filosofia di pensiero: semplicità e velocità. Nulla deve impiegare più dello stretto necessario o rubare tempo allo sviluppo concreto; sicuramente una *forma mentis* Agile a tutti gli effetti e che rispecchia anche altri

[38]L. Boiser, *Cumulative Flow Diagram: Your Most Powerful Tool to Create a More Stable & Predictable Flow*, 2019, in `https://kanbanzone.com/2019/power-of-cumulative-flow-diagram/`

modelli e framework, ma che in qualche mondo Kanaban incentiva e accresce. Dunque non esistono sprint review, retrospective, sprint planning o milestone.

Gli eventi in Kanban più comuni nei team e che possono essere implementati sono dunque:

- Il Planning Meeting
- Il Daily stand up meeting

2.5.1 Il Planning Meeting

Come in Scrum, anche nel modello Kanaban troviamo il *Planning Meeting*.

Si tratta di un evento che tipicamente avviene all'inizio di ogni progetto, durante il quale tutti i membri del team si riuniscono insieme alle figure interessate nella riuscita del progetto stesso, sia committenti sia stakeholder.

Nella maggior parte dei casi questo evento avviene davanti alla Kanban Board, ancora vuota per ovvie ragioni. Tipicamente l'SRM o l'SDM seguono il meeting cercando di coordinare per evitare confusione.

In una prima fase tutti gli elementi del team scrivono su un oggetto, che può essere uno stiky note, cards (o su qualsiasi oggetto sia poi in grado di essere attaccata alla board), tutto quello che secondo loro è da sviluppare per ottenere il risultato voluto. Una volta concluso questo intervallo di tempo, si inizia a popolare il Product Backlog (PB), dando maggiore priorità a ciò che ha reale valore per il committente. Resta inteso che non tutto riuscirà ad essere disposto nel PB; per questa ragione gli elementi validi non inseriti saranno conservati e successivamente inseriti nel PB a tempo debito.

Come sottolinea E. Brechner[39] il coinvolgimento di tutti quanti al popolamento di questo artefatto è significativamente differente rispetto a quanto visto in Scrum, dove è il Product Owner ad incaricarsi di definire il PB insieme ai committenti del progetto. Questa può sembrare una leggera differenza, ma in realtà ne rispecchia una molto grande: il coinvolgimento di tutti gli elementi del team in Kanban avviene dall'inizio alla fine ed è proprio un pilastro portante di questo modello portare ogni singolo elemento del team a sentirsi costantemente partecipe del progetto e della produzione. Tutto ciò, come si è detto, garantisce una sensibilizzazione da parte dell'elemento interessato e un forte senso di appartenenza che genere un miglioramento e incremento produttivo non indifferente.

[39] E. Brechner, *Agile...*, *cit.*, pp. 25-27

2.5.2 Il Daily standup meeting

L'ultimo evento da discutere è sicuramente quello che gioca un ruolo fondamentale in questo modello e che lo stesso Brechner definisce il "mantra di Kanban": il Daily standup meeting.

Il Daily ha inizio di fronte alla Kanban board con tutto il team riunito in piedi e ha una durata di quindici minuti, la stessa soluzione proposta da Scrum e, come anche quel framework, qualcuno lo guida non per senso di potere, ma per regolare il meeting stesso e assicurarsi che sia breve, dettagliato e non sfoci in una chiacchiera da bar.

A differenza di Scrum, questo modello non prevede invece un figura fissa per il coordinamento del meeting e, anche se ogni membro del team potrebbe occuparsene, tipicamente è l'SDM o il Project Manager a dirigerlo a seconda della formazione del team e da come i ruoli siano disposti, visto che, come abbiamo detto non vi è una regola chiara e rigida per le figure al suo interno.

Il moderatore scelto assieme al team esplora le attività da destra a sinistra, dall'alto verso il basso, discutendo le opzioni per il trasferimento più rapido alla fase successiva di ogni task. In questa fase tutti gli elementi del team sono incoraggiati a parlare ed evidenziare elementi importanti al fine del risultato finale atteso.

Appare opportuno puntualizzare il perché si esplorano le attività da destra a sinistra. Il motivo è presto detto: se visualizziamo la Kanban Board (vedi Figura 2.1), notiamo che all'estrema destra abbiamo la colonna *Validate* e che in questo punto ci sono i task che sono o stanno per essere completati; di fatti andando tutto il workflow con una linea di priorità, abbiamo i task più importanti stanziati in tale colonna. Per questa ragione si sceglie di partire da destra, al fine di assicurarsi di risolvere immediatamente eventuali problemi che infliggono sui task più importanti che devono essere rilasciati alla successiva release.

Il punto focale di questo meeting e che dovrebbe essere all'apice della lista degli argomenti da trattare in ognuno, dovrebbe rispecchiare le seguenti due domande:

- Ci sono blocchi o interferenze che stanno rallentando il workflow?
- Qualcuno necessita di supporto o assistenza per un determinato task che non riesce a completare?

Entrambi i quesiti sono strettamente simili a quelli che vengono posti durante un Daily Scrum Meeting, ma l'enfasi e la risoluzione sono differenti rispetto a Scrum.

In Kanban infatti se le domande sopra esposte risultano positive per qualche elemento, sarà l'SDM o il PM ad assegnare immediatamente un elemento del team a supporto del

task che sta generando il ritardo; in questo modo si auspica che il problema venga risolto, il workflow prosegua senza intoppi e vi sia una totale sincronizzazione tra gli elementi del team.

Il focus si concentra quindi sul minimizzare e ridurre il tempo speso per le attività da svolgere in tutte le fasi della Kanban Board e sul cercare e risolvere i colli di bottiglia che possono interrompere o rallentare il workflow e il risultato finale.

2.6 Implementazione e utilizzo di Kanban

La natura di questo modello è decisamente attraente per la sua semplicità implementativa e facilità di gestione, tutti fattori che posso trarre in inganno dal momento che non sempre può risultare la migliore soluzione. Nello studio che abbiamo compiuto infatti sono emerse diverse sfaccettature che andrebbero tenute in considerazione nel momento nel quale si sceglie questo come altri modelli e la banalità non si rispecchia certamente in Kanban.

Come ogni modello, anche Kanban va applicato e scalato eventualmente a seconda delle esigenze sia aziendali sia organiche, intese come team ed elementi che ne fanno parte, oltre che della natura del progetto e della richiesta di rilascio di valore. Di seguito riportiamo un'analisi fatta durante lo studio dei modelli e framework Agili al fine di chiarire quali siano le dimensioni di Kanban e dove questo si presti meglio.

A fronte di una prima analisi sulla natura della committenza, è emerso che per questo modello le priorità sono altamente variabili; questo significa che il team deve essere capace di variare anche giornalmente le priorità secondo le richieste e aspettative del committente e nel contempo è vitale mantenere una linea su rilasci continui (anche giornalieri) e incrementali di valore come vuole il *Continuos Delivery*. È quindi estremamente necessario che il team intero abbia una visione del workflow istantanea, ovvero tutti sappiano esattamente in che punto si trovano nella fase progettuale e riescano a sincronizzarsi di conseguenza. Anche se l'approccio è più semplice rispetto a Scrum, che richiede la conoscenza di molti eventi, ruoli e artefatti, l'organizzazione di Kanban, con particolare riferimento alla sintonia e sincronia del team, sono fattori fondamentali per il corretto funzionamento del modello ma non per questo di facile applicazione.

Tipicamente, Kanban è consigliato per progetti di dimensione piccola o media; tuttavia può tranquillamente essere scalato anche su tipologie di progetti di grandi dimensioni, creando più team che collaborano tra di loro, un po' come viene fatto con SAFe, un modello Agile che scala su grandi aziende o progetti molto estesi richiedenti l'ausilio di più team di sviluppo Scrum.

Un grosso vantaggio di questo modello è la curva di apprendimento. Essendo privo di molti eventi, ruoli e artefatti, si adatta estremamente bene per team tipicamente immaturi o con un background basso nel mondo Agile. Tuttavia, qualora si sia privi delle giuste nozioni teoriche e pratiche, l'ausilio eventuale di un coach è fortemente consigliato, in quanto la sua mancanza recherebbe grossa perdita di tempo e denaro poiché la soluzione finale non sarebbe quella voluta. Se guidati da un elemento esperto, un giovane team che si approccia al mondo Agile potrebbe trarre grandi benefici e ottimi risultati in poco tempo da questo modello considerata, la sua piccola dose di nozioni da assimilare e applicare, cosa che con Scrum non sarebbe possibile, poiché la sua curva di apprendimento è decisamente più alta e richiede uno sforzo ben più grande.

Un altro grande vantaggio a favore di questo modello sono sicuramente gli strumenti a supporto.

Non è infatti difficile trovare sul web una grossa quantità di tool e software orientati all'utilizzo della Kanban board; le offerte sul mercato sono per ogni gusto e prezzo, sicuramente quelle più gettonate sono Trello, Jira e Asana. Questi strumenti digitali, in un'epoca nella quale spesso può essere necessario lavorare a distanza per varie ragioni, offrono come grande vantaggio quello di poter continuare a mantenere un alto livello di sviluppo e rilascio anche se i membri del team non sono fisicamente collocati nello stesso posto. Una alternativa apre grandi spazi su tutti i fronti digitali e di risorse umane, anche se secondo gli esperti del settore come E. Brechner non è la soluzione ottimale.

Tutto ciò ha permesso al modello di evolversi ed espandersi, diventando così oggigiorno uno dei più utilizzati nel mondo Agile.

Capitolo 3

Jira

Jira ha iniziato la sua scalata che oggi ha portato i suoi prodotti ad essere tra quelli più utilizzati come programmi per la gestione e il tracking dei bugs, aiutando i team di sviluppo a tenere traccia e gestire i problemi nei loro progetti.

Sviluppata, pensata e tradotta in realtà da Atlassian, ha cercato nel corso degli anni di migliorare costantemente l'esperienza dello sviluppatore venendo incontro a tutte le esigenze del mercato e ritagliandosi così una grossa fetta di imprese che hanno orientato l'intero settore di sviluppo al suo interno.

Successivamente, con l'avvenire delle metodologie Agili, Jira ha migliorato le caratteristiche interne del software per supportare al massimo delle capacità il framework Scrum e il modello Kanban. Questo ad oggi consente al tool di gestire in maniera efficiente le metodologie Agili, con l'obiettivo realizzato di portare a zero l'effort per la gestione del team da distanza.

Nei successivi paragrafi andremo a descrivere un particolare software sviluppato da Atlassian nella famiglia Jira; si tratta di uno strumento a supporto dei team, prodotto specificatamente per lo sviluppo Agile utile al framework Scrum o al modello Kanban. Per comodità e correttezza descrittiva verranno utilizzati i termini in lingua madre (inglese).

3.1 Jira Software

3.1.1 Lo strumento

Jira Software è un particolare software che fa parte di una famiglia di prodotti di Jira, quali per esempio, Jira Confluence, Jira Allign, Jira Service Desk e Jira Core.

Si tratta di un tool per la gestione, la pianificazione e il tracking dei progetti software. Prima che questa versione venisse rilasciata, se si voleva usufruire di un servizio quanto più simile e che permettesse di sfruttare i modelli Agili, era necessario utilizzare la suite Jira con l'add-on Jira Agile.

Questo strumento oggigiorno è diventato quasi indispensabile per ogni team Agile, poiché il suo impatto in termini di aumento della produttività e sincronizzazione da parte dei membri del team è chiaro e visibile fin da subito.

La sua efficienza sta nel fatto che riesce a garantire un'ininterrotta collaborazione anche nei casi estremi o voluti, dove i membri del team non sono a diretto contatto. Ciò permette anche di migliorare la comunicazione che per via della distanza potrebbe rivelarsi non appropriata o confusa e aumentare la trasparenza tra gli elementi.

Attraverso Jira Software è possibile per tutti i manager incaricati (PM, SM, PO, SRM, SDM) tenere sempre sotto controllo la situazione ed eventualmente intervenire immediatamente qualora si presentasse un problema qualsiasi. Lo stesso vale per tutti i committenti e stakeholders coinvolti nel progetto, che possono controllare l'avanzamento dei vari rilasci a distanza.

Jira Software offre due tipologie di modelli per la gestione e l'implementazione del progetto, sia per Scrum sia per Kanban, i quali offrono diversi livelli di dettaglio e al contempo difficoltà nel loro utilizzo quotidiano. La scelta implementativa è a totale discrezione dell'utente, che viene dettagliatamente consigliato nelle istruttorie fornite da Atlassian[40] su quali dei due optare in base al tipo di esperienza e di progetto in corso di creazione. Andando nel dettaglio, i modelli disponibili oggetto della tesi possono essere di tipo:

- Classic software project
- Next-gen software

Classic software project

La tipologia Classic software project ha una predisposizione del software avanzata con una personalizzazione molto alta, che consente anche di compiere query attraverso il linguaggio di Jira orientato ai database denominato JQL (Jira Query Language), estremamente utile per ricercare issues; tipicamente questo modello è consigliato a team altamente esperti nell'utilizzo del software.

[40] Atlassian, *Learn the differences between classic and next-gen projects*, *Atlassian Support*, USA, 2020, in `https://support.atlassian.com/jira-service-desk-cloud/docs/learn-the-differences-between-classic-and-next-gen-projects/`

Next-gen software

La tipologia Next-gen software, al contrario della precedente, ha un predisposizione ridotta per non creare confusione durante l'utilizzo, che si presta meglio a team con poca esperienza o risorse ridotte. Poiché durante gli studi compiuti e le implementazioni progettuali ci si è avvalsi della tipologia Next-gen software, in questa tesi verranno discussi modelli con tale struttura.

3.1.2 La gerarchia

Come molti software, Jira Software organizza i suoi dati attraverso una struttura gerarchica[41]. Al livello più basso troviamo i *fields* che sono utilizzati per contenere le informazioni più grezze. Il software permette di scegliere se organizzarli e modificarli a piacimento; tipicamente dipende dal tipo di necessità del team sul progetto che si sta svolgendo. Al livello intermedio troviamo le *issues*: si tratta di unità di lavoro da eseguire che appartengono ad un progetto il quale ne definisce il contesto. Al livello più alto troviamo le *project categories* che sono gruppi di progetti logicamente simili.

3.1.3 Project Category

Il *Project Category*, che in italiano si traduce come *categorie di progetti*, è un raggruppamento logico di progetti, tipicamente simili in natura[42]. Si tratta di un elemento opzionale, non è obbligatorio utilizzare questo schema.

Senza l'utilizzo di questo raggruppamento un progetto non appartiene a nessuna categoria ed è considerato da Jira Software non categorizzato. È tuttavia importante sottolineare che posizionare progetti sotto delle categorie di progetti, in particolar modo quando si hanno molti progetti da coordinare o ai quali si partecipa, può essere uno strumento utile per organizzare meglio il workflow sia di uno sviluppatore sia di un ipotetico manager. È comune vedere team che hanno elementi di sviluppo condivisi, intesi come risorse umane, che necessitano di cambiare da un progetto all'altro. Questo strumento può allora diventare estremamente utile qualora questo caso si palesasse, permettendo così alla risorsa di muoversi agevolmente nel software e di avere un visione sempre chiara di quello che sta accadendo.

[41] P. Li, *Jira 5.2 Essential*, USA, Packt Publishing, 2013, p. 66 ss
[42] *Ivi*, p. 68

3.1.4 Projects

In linea generale i *Projects*, che in italiano si traducono come *progetti*, in Jira Software sono una collezione di issues[43]. Volendo ricorrere ad un'immagine, possiamo vedere il progetto come un grosso contenitore di issues (risolte e non risolte) all'interno del quale lo sviluppatore sa esattamente dove sono collocate (per tipologia) le stesse, che vengono prelevate (non risolte) e lavorate per poi essere inserite nuovamente al loro interno una risolte. Su Jira Software un progetto definisce anche le varie regole e i limiti di accesso alle issues stesse, come ad esempio chi avrà l'autorizzazione a visualizzare le issues e avvisare gli interessati quando vengono apportate le modifiche oppure vengono poste in stato risolto. Tipicamente questo comporta una serie di vantaggi per i manager o amministratori del software che gestiscono tutto il flusso progettuale. Così facendo, si scompone un problema in sottoproblemi in modo tale che gli elementi che sviluppano il prodotto non siano distratti dal lavoro altrui. Sicuramente può sembrare un controsenso nel mondo Agile dove la trasparenza gioca un ruolo fondamentale; tuttavia, nei progetti di grandi dimensioni, può essere necessario limitare la visibilità di qualcosa al fine di non confondere o distrarre i componenti del team.

3.1.5 Issues

Le *Issues*[44],che tradotte in lingua italiana sarebbero i *problemi*, rappresentano il lavoro da eseguire per gli elementi del team (vedi Figura 3.1). Una issue può essere creata da un utente e assegnata ad altre persone per essere lavorata e infine risolta. Tipicamente i project leaders (PM, SM, PO, SRM, SDM) generano dei report sull'andamento delle issues per monitorare lo stadio di avanzamento del progetto e misurare come il lavoro sta procedendo. In un certo senso, possiamo dire che Jira Software è un software centrato sulla gestione delle issues.

Una regola fondamentale per una gestione ottimale delle issues e del progetto stesso è conoscere le dimensioni nelle quali queste possono operare; tipicamente ci sono tre importanti fattori da tenere sempre in considerazione:

- Una determinata issue può far riferimento ad uno e soltanto un progetto, è unica nel suo genere.
- Possono esistere e coesistere tipi differenti di issues.

[43] *Ibidem*
[44] *Ibidem*

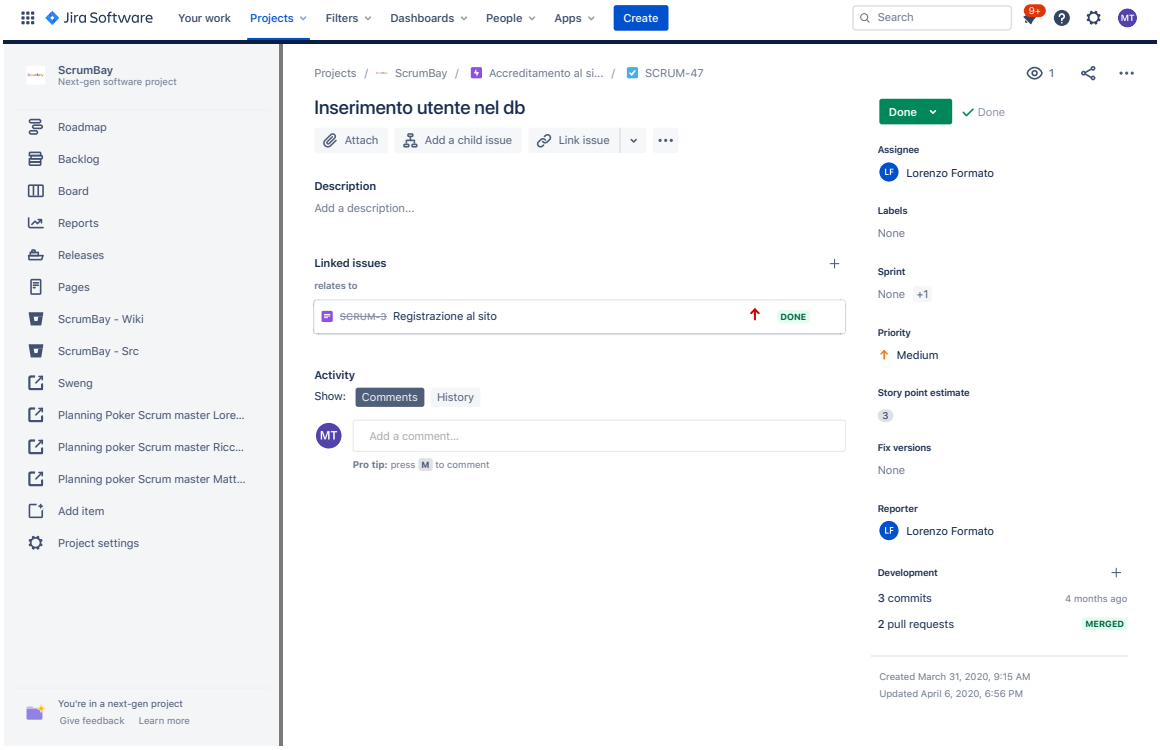


Figura 3.1: Jira Issue

- Una issue contiene molti campi, che possono comprendere diversi valori riferiti alla issue.

Cercando di spiegare in breve i punti elencati sopra, bisogna chiarire che una issue può rappresentare qualsiasi cosa, da una feature ad un bug in un progetto di sviluppo software oppure una richiesta di supporto all'help desk in un progetto strutturato per le Operations.

Proprio per questa ragione una issue non può esistere in progetti differenti, ma può far riferimento ad un unico e specifico progetto. Infine può ricomprendere molti campi, come in Figura 3.1, la quale contiene *activity* riferite a quella particolare issue che sono determinanti per la sua risoluzione.

Possiamo vedere tutto questo come un albero radicato, le cui foglie hanno a loro volta figli che svolgono compiti per soddisfare le richieste del genitore al fine di completare e chiudere il problema in oggetto.

3.1.6 Fields

I *Fields*, che tradotti in lingua italiana sono i *campi*, rappresentano l'unità di dati basica di Jira Software[45]. La loro natura e scopo principale è conservare i dati per le issues e dare a tali dati un significato ragionevole per esistere all'interno della issue.

I fields possono essere classificati in due categorie distinte, vale a dire i *system fields* o i *custom fields*. Entrambi possono essere rappresentati in molte forme diverse, come campi di testo, elenchi a cascata e selettori per l'utente. Allo stesso modo delle issues, anche i fields hanno tre fattori fondamentali che devono rispettare per coesistere con gli altri elementi di Jira Software, quali:

- I fields contengono valori per le issues
- I fields possono avere comportamenti (obbligatori o nascosti)
- I fields possono avere una vista o una struttura (campo di testo, elenco)

Secondo questi fattori quindi, i fields sono ciò che cattura e mostra i dati agli utenti interessati, che siano essi manager o sviluppatori. Comprendendo come utilizzare efficacemente i campi, è possibile trasformare Jira Software in un potente sistema informativo per la raccolta, l'elaborazione e il reporting dei dati.

3.2 Lavorare con Scrum in Jira Software

Jira Software fornisce la possibilità di utilizzare a pieno regime il framework Scrum al suo interno.

Bisogna sottolineare che sono due tipologie di board predisposte dal software stesso, una per Scrum e l'altra per Kanban; rimane possibile estendere delle personalizzazioni per altri framework e modelli che non saranno oggetto della nostra analisi.

All'interno di un progetto Scrum, in Jira Software è possibile distinguere tutti gli elementi caratteristici del framework: nella barra laterale (vedi Figura 3.2) è possibile visionarli in modo chiaro.

Scopo di questo paragrafo è cercare di chiarire la struttura del tool e il suo utilizzo con Scrum.

[45] *Ivi*, p. 69

3.2.1 Il backlog

Jira Software ci consente di gestire in modo snello e intuitivo le priorità per ogni attività o issues tramite gli Epics e gli Sprint che si trovano nel backlog di ogni singolo progetto. Il backlog di Jira Software rispecchia esattamente quello che è previsto per Scrum. La pagina del backlog (vedi Figura 3.2) è sviluppata su due livelli: backlog e sprint backlog.

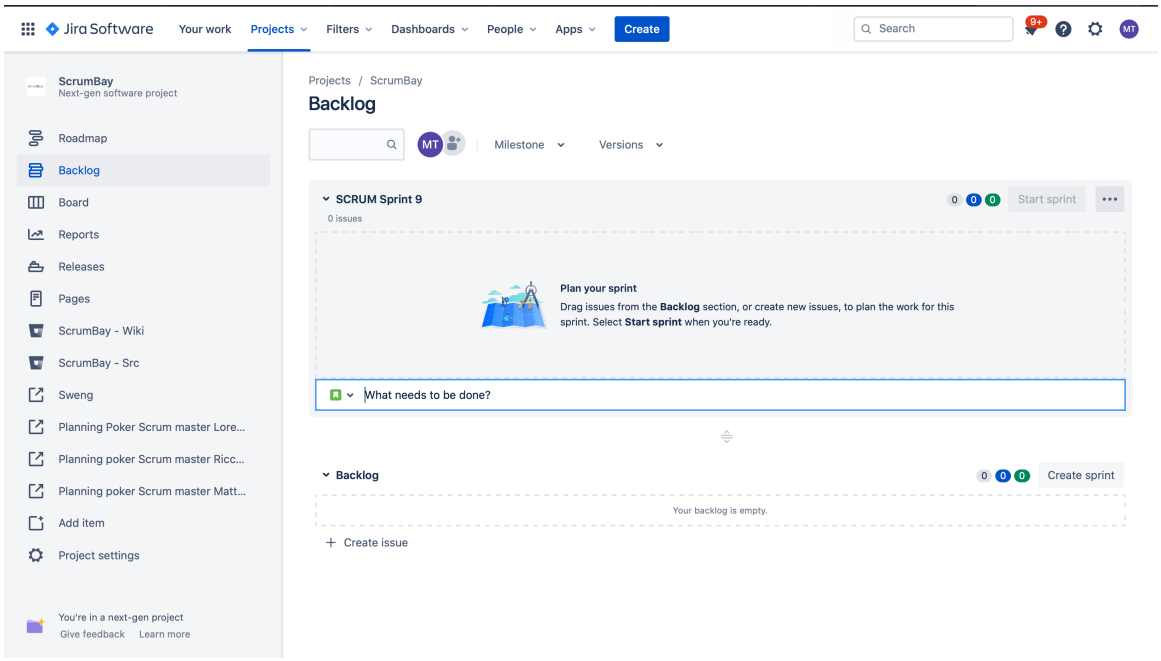


Figura 3.2: Jira Scrum backlog

Nella parte inferiore denominata backlog troviamo tutto quello che sarà sviluppato, ordinato per priorità dal Product Owner, che tipicamente risulta essere l'amministratore del progetto, ovvero colui che vede tutto in chiaro senza blocchi o restrizioni. Il backlog è composto da elementi: ogni elemento è completato da una descrizione, una posizione, una stima delle dimensioni e un valore. La parte superiore si riferisce allo Sprint n: non è altro che lo Sprint backlog, artefatto già discusso ampiamente nel primo capitolo. All'interno del backlog, e nello specifico dell'attività creata, è possibile fornire un valore in *story point* alla issue del caso, che sarà visualizzato nel *Velocity report*. Nel complesso, tutta la pagina è visibile a tutti gli elementi del team, che tipicamente partecipano attivamente alle modifiche.

3.2.2 La board

Lo strumento più utilizzato all'interno di Jira Software è decisamente la *board* (vedi Figura 3.3).

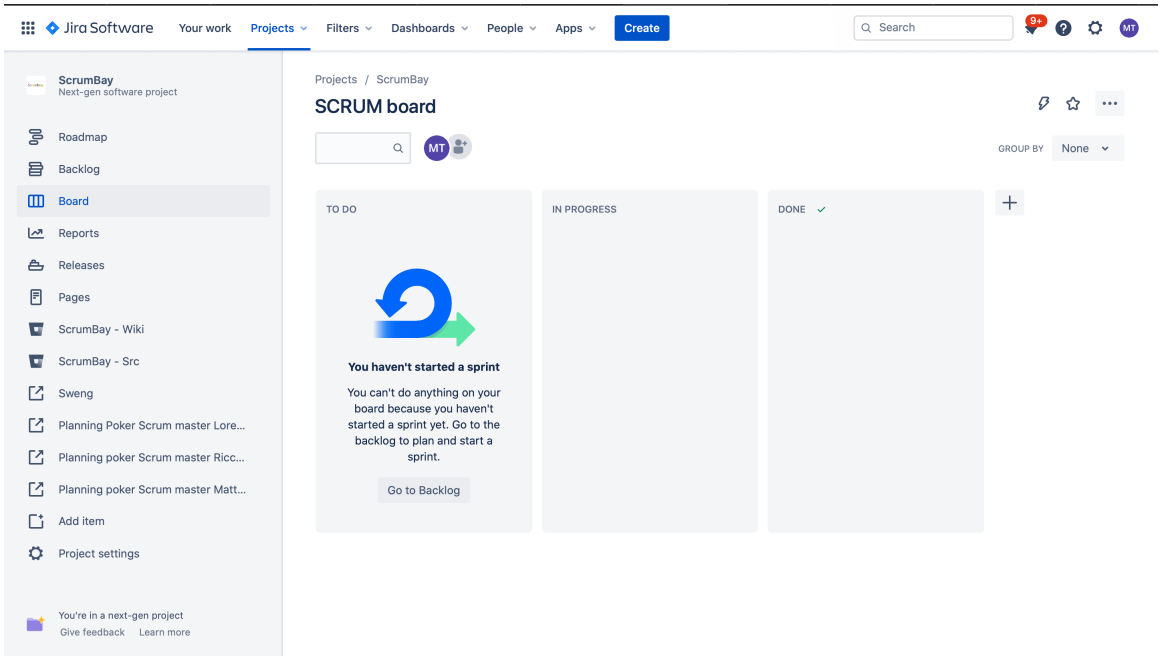


Figura 3.3: Jira Scrum board

Questo strumento viene quotidianamente visto e utilizzato da tutti gli elementi del team, compresi Product Owner e Scrum Master, oltre che in alcuni casi, dai committenti o stakeholders direttamente interessati al successo del progetto.

Tipicamente, ogni membro del team passa dalla board almeno due volte, una alla mattina e una alla sera. I due momenti temporali citati sono quelli fondamentali, in quanto la mattina si cerca di ottenere una chiara visione attuale del progetto per poter di conseguenza strutturare la giornata lavorativa, mentre la sera per controllare che tutti i task eventualmente conclusi siano stati spostati nella colonna di riferimento corretta.

Questi due momenti non sono gli unici; è buona pratica infatti che ogni elemento passi spesso a controllare durante la giornata lo stato della board, anche se questo è maggiormente richiesto con Kanban, poiché Scrum rimane comunque un processo iterativo incrementale basato su Sprint a intervallo temporale.

L'utilizzo di questa board elettronica permette di ottenere molti vantaggi che tipicamente abbiamo solo in presenza, quali:

- La board è progettata in modo tale da permettere ai team di lavorare rispettando il lasso temporale dello Sprint.

- Rende trasparente il lavoro del team.
- Tutti i membri del team possono accedervi in qualsiasi momento, nessuno ha dubbi su ciò che deve fare e può individuare con facilità eventuali bloccanti.

Tutto ciò consente di ottimizzare la comunicazione e la trasparenza, promuovere la pianificazione degli Sprint, lo sviluppo iterativo e infine migliorare di conseguenza il focus e l'organizzazione del team.

3.2.3 I report

Una parte per garantire il successo e il buon funzionamento dei progetti in JIRA è la creazione di report. Ciò implica l'acquisizione di conoscenze sulla salute, i progressi e lo stato generale dei progetti in Jira Software tramite la pagina di *report*.

All'interno della pagina dei report è presente una overview generale che ci mostra diversi tipologie di report, quali:

- Burnup report
- Sprint burndown chart
- Velocity report
- Cumulative flow diagram

Tutti quelli elencati sono grafici altamente descrittivi e che vengono costantemente utilizzati da figure come PO e Scrum Master per capire l'andamento del team e su cosa è possibile ottimizzare. Tipicamente, i più utilizzati sono sicuramente lo Sprint burndown chart e il Velocity report, artefatti già discussi ampiamente nel primo capitolo.

In generale, questi strumenti sono completamente automatizzati; questo consente di ridurre notevolmente l'effort e il grado di difficoltà rispetto alla strutturazione manuale, garantendo a tutto il team (non solo lato manageriale) dati istantanei e precisi, permettendo quindi una correzione o deviazione della strada intrapresa qualora necessario.

3.2.4 La roadmap

A volte screditato e mal utilizzato, lo strumento della *roadmap* può fare la differenza tra il successo o meno del progetto che si sta svolgendo e Jira Software ci garantisce di sfruttarlo al meglio (vedi Figura 3.4)[46].

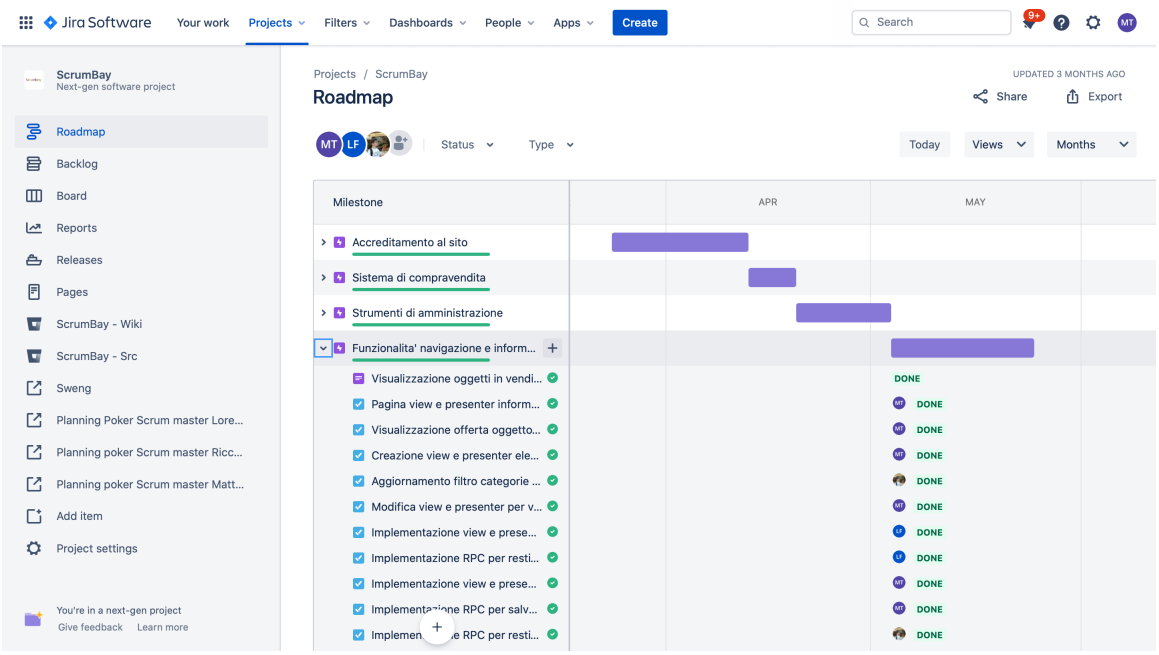


Figura 3.4: Jira roadmap

Si tratta di uno dei tool più potenti che un team possa utilizzare. Una buona roadmap garantisce che tutti coloro che lavorano sul progetto, dai dirigenti senior ai leader di prodotto e addetti alla progettazione e allo sviluppo, comprendano lo stato del lavoro e siano allineati alle priorità imminenti.

Tipicamente, vengono viste come un effort inutile e già gestito dallo stesso framework, poiché la creazione e gestione richiede chiaramente lavoro aggiuntivo che non tutti sono disposti a concedere. In Jira Software tutto questo cambia.

Le roadmap in Jira Software infatti consentono di creare rapidamente una sequenza temporale dei piani, aggiornare le priorità mano a mano che cambiano e comunicare lo stato del lavoro agli stakeholders o committenti, semplificando quindi anche la condivisione con le parti interessate al di fuori del team. La gestione e lo sforzo sono minimi e se cambiano le priorità per le quali necessitiamo di spostare una determinata epica, story o bug basterà utilizzare un veloce "drag and drop" per aggiornare la roadmap.

[46] Atlassian, *Why you need roadmaps in Jira Software*, in *Atlassian Blog*, 2018, `https://www.atlassian.com/blog/jira-software/roadmaps-in-jira`

Dunque, a differenza di quanto si può pensare, Atlassian è riuscita ad alleggerire uno strumento che prima richiedeva non solo una implementazione esterna tramite fogli di calcolo e tool ulteriori di terze parti, ma anche uno sforzo in risorse umane non indifferenti, che con le roadmap viene quasi annullato.

L'utilizzo di questo strumento è previsto dal tool sia per il framework Scrum sia per il modello Kanban.

3.3 Lavorare con Kanban in Jira Software

La seconda ma non meno importante board che troviamo già ben predisposta è sicuramente quella per l'utilizzo del modello Kanban, nota come Kanban board.

Un forte impatto determinato dall'utilizzo di questo modello è la possibilità che il team inizi senza quasi nessun requisito, tema già discusso nel secondo capitolo di questa tesi. Se il modello di per sé risulta rispetto a Scrum meno ostico allorquando si tratti di padroneggiarne i requisiti fondamentali al suo utilizzo, Jira Sofware semplifica ancor più questo paradigma, grazie al prefetto connubio tra modello e software che il team di Atlassian è riuscito a creare.

Lo scopo di questo paragrafo è cercare di chiarire l'utilizzo del modello all'interno del software e i vantaggi che questo comporta.

3.3.1 La board

Come già ampiamente discusso nel capitolo due, le board Kanban (vedi Figura 3.5) offrono al team visibilità completa sulle azioni da eseguire, così che, quando una card (o elemento di lavoro) è completata, il team può passare rapidamente oltre.

Jira Software offre board Kanban virtuali che contengono tutti i dettagli di ogni fase lavorativa, pur mostrando solo le informazioni pertinenti sulla board stessa. La personalizzazione è ad un livello quasi estremo; lo scopo infatti è cercare di dar la sensazione all'utente di avere la board fisicamente davanti, proprio per ovviare al problema sottolineato da E. Brechner, trattato nel capitolo due. Le board digitali come quella di Jira Software, infatti, consentono ai team che non condividono lo spazio fisico di un ufficio di utilizzare la board Kanban in remoto e in modo asincrono. A prescindere da dove o quando i membri del team effettuano l'accesso alla board, essi vedranno sempre lo stato più aggiornato del progetto.

Come per Scrum, anche nel modello Kanban sul tool è prevista una gestione amministrativa che permette ai vari manager di settore (SRM, SDM) di controllare il flus-

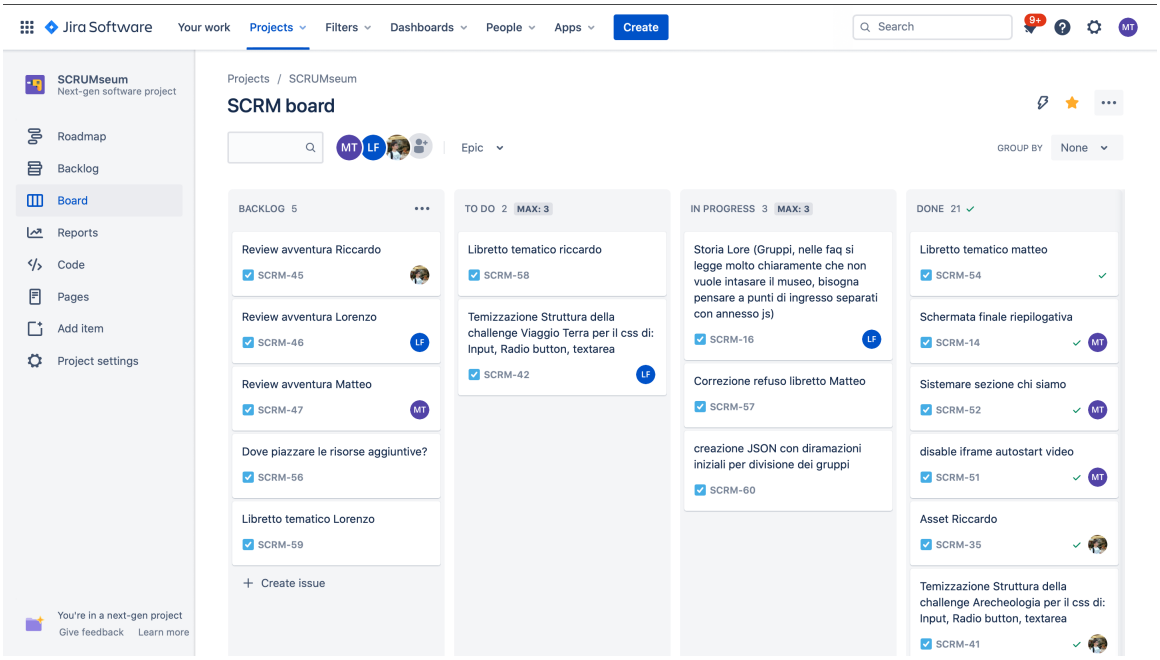


Figura 3.5: Jira Kanban board

so del progetto e del lavoro in corso, fornendo dati precisi che possono essere utili ad un'eventuale correzione della strada intrapresa dal team.

Vi sono diverse caratteristiche avanzate, spesso date per scontate, per la gestione delle colonne importanti da sottolineare:

- Personalizzazione delle colonne con alto livello di dettaglio
- Configurazione del WIP della colonna
- Workflow dinamico e flessibile attraverso la creazione ed eliminazione delle colonne

Tutte quelle descritte sono caratteristiche scontate in una board Kanban fisica, ma non lo sono altrettanto virtuale.

Le colonne infatti possono essere configurate con livelli di dettagli molto alti, dal rappresentare i principali stati del workflow ad aggiungere colonne al fine di raggruppare il lavoro in flussi per parametri importanti per il team, quindi altamente dipendenti a seconda del tipo di progetto che si sta svolgendo.

La configurazione WIP (work in limit progress), può essere impostata con il tasto *more "... "*, tasto posto nell'alto della colonna al fine di ridurre il numero di cards presenti nella colonna stessa.

Infine il workflow generale delle colonne può essere complesso o semplificato a seconda della scelta del team, al fine di modellare le colonne a piacimento e spostarle o modificarle anche a sviluppo già avviato.

Una chiara differenza di dettaglio è data dalla scelta iniziale sul modello utilizzato, Next-gen o Classic. È inteso che nel primo abbiamo funzionalità ridotte per garantire un basso livello di complessità di gestione, mentre nel secondo le colonne possono essere finemente modificate, garantendo ad ogni tipologia di team le strutture necessarie al fine di ottenere ottimi risultati.

La board di Kanban in Jira Software è decisamente uno strumento meticolosamente sviluppato per far fronte a tutte le esigenze di team con o senza molta esperienza e per rendere il loro percorso progettuale quanto più vicino possibile a quello in presenza, oltre che poco stressante poiché modellata con cura al fine di non "caricare lo sviluppatore più del necessario. Sicuramente una delle migliori board virtuali a supporto di Kanban al momento presenti sul mercato.

3.3.2 Le cards

Un altro elemento da evidenziare come componente decisivo del tool, sono le cards (vedi Figura 3.6), oggetti al centro del modello Kanban.

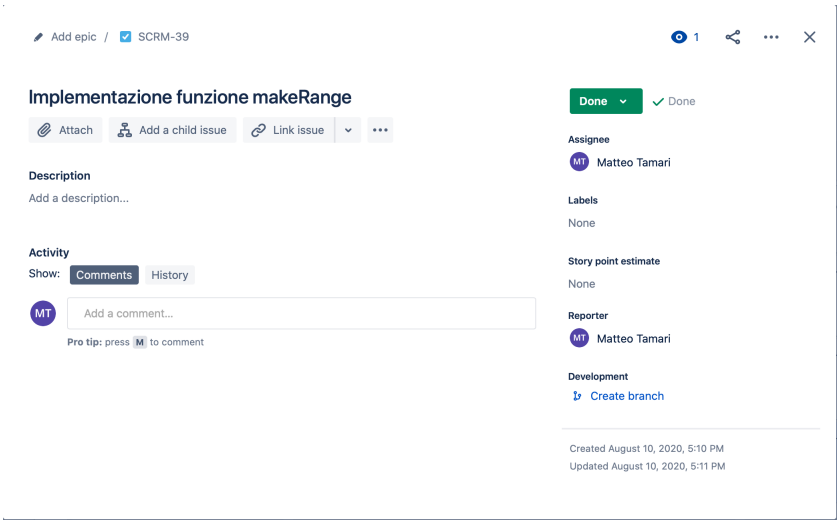


Figura 3.6: Jira cards

In Jira Software con cards intendiamo particolari tipi di oggetti che hanno funzioni altamente personalizzabili.

Vengono utilizzate per descrivere il lavoro da fare e compongono tutto il workflow del progetto. Le estensioni che questi oggetti possono avere sono molte, sicuramente troppe per essere descritte in questa tesi; comunque, le più importati e utilizzate tipicamente sono le seguenti:

- Inserire nodi parents creando una gerarchia strutturata di cards

- Assegnare le cards a un determinato elemento del team
- Assegnare le cards a un determitato branch (a patto di aver aggiunto bitbucket come repository)
- Esportare le cards in formati vari per inserirle nella documentazione da mostrare al top management

Come mostrato in Figura 3.6, le opzioni associabili a questi oggetti sono di immediata visibilità all'utente, proprio per evitare ogni effort aggiuntivo inutile e soddisfare immediatamente la sua funzione.

Per approfondimenti si rimanda al testo *Jira Essential*[47] che chiarisce in modo dettagliato ogni caratteristica e opzioni attivabili da una determinata card.

3.3.3 I report

È comune oggigiorno trovare strumenti a supporto del modello di Kanban che non presentano grande cura e dettaglio nella stesura e visione dei report progettuali; ciò accade sia per una ragione di complessità di sviluppo sia per una non grande richiesta da parte del mercato che non sempre esprime la soluzione migliore. In Jira Software questo non avviene e lo sviluppo dei report è chiaro, gratuito e ben costruito.

Jira Software include una serie di report predefiniti e gadget preinstallati che aiutano il team a tenere traccia della durata di ciclo per i rilasci dei prodotti. Controlla lo stato dei ticket nel tempo oppure verifica se i dati del processo corrente possono essere usati per prevedere le prestazioni future. I gadget consentono di mostrare sulla dashboard informazioni riepilogative inerenti i dati del progetto o ticket, fornendo una posizione centrale per l'accesso rapido ad esse.

Come abbiamo visto e spiegato a fondo nel capitolo due, il modello di Kanban presenta due fondamentali report che vanno distinti sia per funzionalità operative sia per tipologia di dati interpolati:

- Cumulative Flow Diagram
- Control Chart

[47] Atlassian, *Customize cards*, USA, Atlassian, 2019, in `https://support.atlassian.com/jira-software-cloud/docs/customize-cards/`

Cumulative Flow Diagram in Jira Software

Si tratta sicuramente del report più importante che il modello contempla, di fondamentale importanza per una corretta pianificazione e procedimento di tutto il workflow del progetto.

Il Cumulative Flow Diagram (CFD), ricordiamo, è un grafico ad area che mostra i vari stati degli elementi di lavoro per un'applicazione o una versione corrente del progetto. L'asse x orizzontale in un CFD indica il tempo e l'asse y verticale indica le cards (o problemi da risolvere).

In Jira Software ogni area colorata nel grafico corrisponde esattamente a uno stato preciso del workflow ovvero una colonna della board. È infatti possibile vedere che le aree rappresentate sono esattamente corrispondenti in numerazione alle colonne presenti nella board.

La Figura 3.7 mostra il funzionamento del grafico, che è stato estratto dal progetto di Tecnologie Web al quale il sottoscritto autore della tesi ha partecipato.

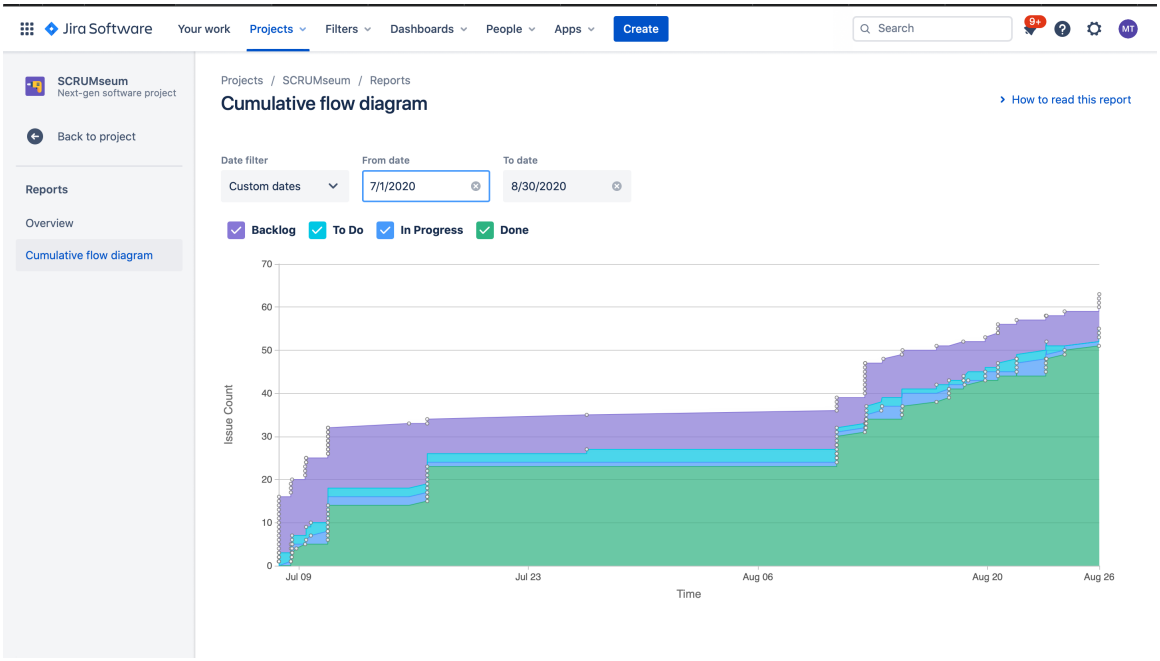


Figura 3.7: Jira Cumulative Flow Diagram

La configurazione del tool prevede anche che colui che analizza questo diagramma possa dinamicamente rimuovere o aggiungere determinate colonne dal grafico. Grazie a questo sistema, la figura incaricata sarà in grado di analizzare separatamente i vari settori e aree che stanno lavorando al progetto; questo permette di evidenziare carenze o

difficoltà che si possono presentare nel team e in particolar modo nei settori occupazionali specifici degli elementi che lo compongono.

Nella parte superiore della Figura 3.7 è possibile inoltre impostare l'intervallo temporale che si vuole analizzare utilizzando il filtro per data.

Control Chart in Jira Software

Il secondo report grafico, che tipicamente si trova all'interno di un progetto strutturato con il modello Kanban su Jira Software, è il *Control Chart*.

Il Control Chart è il grafico che ci mostra l'intero ciclo di vita del progetto.

Il suo compito è prendere il tempo trascorso da ogni singola issue in uno o più stati particolari e mapparli su un intervallo di tempo specificato.

Il risultato finale è la visione della media, la media mobile e la deviazione standard per i dati interessati nell'analisi. Questo report aiuta l'analista del caso a identificare se i dati attuali possono essere utilizzati per determinare le prestazioni future. Minore è la varianza nel ciclo di vita di una issue, maggiore è la confidenza nell'usare la media (o mediana) come indicazione della performance futura. L'impiego di questo particolare report può essere adatto alle seguenti analisi:

- Analizzare le prestazioni passate del team
- Misurare l'effetto di un cambiamento sulla produttività del team
- Fornire agli stakeholders esterni la visibilità delle prestazioni del team
- Analizzare e utilizzare le prestazioni passate per impostare obiettivi futuri per il team

Sebbene si tratti di un ottimo strumento di analisi, la sua presenza e il suo utilizzo nel modello non è assolutamente banale. Si tratta infatti di un report particolare che solo i progetti di tipo *Classic software project* implementano. All'interno dei *Next-gen software* non è prevista: si tratta di una scelta voluta da parte di Atlassian che non inserisce questo tipo di report nei Next-gen a causa della sua natura esemplificata orientata a team non pratici, con poca esperienza e il cui progetto è di piccole dimensioni. La sua analisi, infatti, richiederebbe un effort aggiuntivo nonché un'esperienza nella chiave di lettura non banale. Tutto questo porterebbe solo confusione all'interno del giovane team, che invece deve concentrarsi su un Continuos flow e Delivery, elementi chiave del modello e per la riuscita del progetto stesso.

È bene sottolineare che Jira Software offre molti report di analisi che possono essere utilizzati per visualizzare dati in modalità differenti e che vanno ad interpolare dati diversi sviscerando temi più profondi, quali: Average Age Report, Created vs. Resolved Issues Report, Pie Chart Report, Recently Created Issues Report e molti altri ancora. Questi report aggiuntivi non saranno oggetti della tesi e per una chiara delucidazione si rimanda alla guida ufficiale Atlassian[48], che spiega e chiarisce a fondo tutte le tipologie di report e analisi presenti nel software.

3.3.4 La roadmap

Come per Scrum, anche per Kanban viene messo a disposizione da parte di Jira Software il tool per visualizzare e personalizzare la *roadmap*[49] del progetto. Essendo il suo funzionamento identico in tutto e per tutto a quello illustrato nella parte relativa a Scrum, si rimanda a tale sezione per relativi chiarimenti.

[48] Atlassian, *Reporting in JIRA*, USA, Atlassian, 2018, in `https://confluence.atlassian.com/jirakb/reporting-in-jira-461504615.html`

[49] Atlassian, *Why you need...*, *cit.*, in `https://www.atlassian.com/blog/jira-software/roadmaps-in-jira`

Capitolo 4

Agile methods for Agile working: la fase di Analisi

Nei capitoli precedenti sono state analizzate dettagliatamente due tra le svariate tipologie di modelli e framework Agili. Lo scopo di tale studio è stato raggiungere una approfondita padronanza della materia, al fine di capire in linea generale dove e quando applicarli nonché, in determinati casi, scalarli in modo appropriato.

Grazie alla collaborazione con l'Università di Bologna insieme al Professor Davide Rossi (relatore della tesi) e al Professor Angelo Di Iorio (correlatore della tesi), è stato possibile sperimentare un'applicazione scalata dei modelli Agili; questo perché, per le necessità dei committenti, l'utilizzo di un modello o framework preciso non risultava funzionale, ma anzi avrebbe comportato maggiore confusione, complessità e tempo di apprendimento.

Al progetto è stato dato il nome di "A4A"[50].

Esso nasce dall'idea di migliorare lo smart working dei lavoratori in questa fase emergenziale, che li costringe per ragioni sanitarie a lavorare dalla propria abitazione. Si tratta di personale che non ha mai sviluppato l'esperienza di una vita lavorativa in remoto e le difficoltà in questa particolare fase si sono presentate immediatamente alla porta: il nostro ruolo è stato quello di cercare di risolvere tale problema.

Il dipartimento selezionato per il progetto è stato quello dell'*AUTC - Area Edilizia e*

[50]Il progetto presentato al Dipartimento AUTC ha ricevuto questo acronimo che si traduce in "Agile methods for Agile working". Inoltre è importante sottolineare che al suo interno si utilizza la parola "Lavoro Agile" come sinonimo di "smart working"; abbiamo dunque valutato, insieme ai diretti responsabili, di mantenere la stessa nomenclatura al fine di non confondere le idee generali agli elementi del Dipartimento, fornendo così una parola a loro nota e rendendo meno ostico l'assorbimento mentale dello scopo del progetto.

Sostenibilità; si trattata di molteplici uffici, distribuiti in varie divisioni, tutti altamente qualificati. Come descrive il loro stesso sito, la mission si rispecchia nei seguenti valori[51]:

- Coordinamento, progettazione e direzione di interventi edilizi
- Gestione degli affidamenti diretti e delle procedure negoziate relative ai lavori
- Gestione degli affidamenti diretti e delle procedure semplificate relativi ai servizi di ingegneria
- Programmazione ed esecuzione degli investimenti edilizi e degli interventi manutentivi al di sopra dei 5.000 euro nei distretti logistici dell'Ateneo in raccordo con l'Area Servizi Bologna e, per gli spazi di competenza, anche con le Aree di Campus
- Progettazione specialistica degli impianti audio video e individuazione dei relativi standard, indipendentemente dall'importo
- Assicurare le funzioni previste dalla normativa per il Mobility Manager
- Gestione integrata della sostenibilità di Ateneo declinata in tutte le sue forme
- Assicurare il raccordo delle azioni sulla sostenibilità promosse da altre strutture di Ateneo anche sulla base del programma Multicampus sostenibile
- Attività informatiche e amministrative di supporto

Il numero dei componenti facente parte di questo dipartimento è estremamente alto (circa 100 elementi); è infatti il dipartimento più numeroso e con maggiori responsabilità di tutta l'Università e una applicazione del modello su larga scala si sarebbe rivelata potenzialmente fallimentare a causa di una difficile gestione e coordinamento del progetto.

Con il supporto dei responsabili di area abbiamo selezionato una serie di elementi, sedici per esattezza, al fine di partecipare allo sviluppo di questo progetto e all'implementazione di tale modello.

La sfida da affrontare era quella di dimostrare che l'implementazione dei modelli Agili era possibile anche al di fuori dell'ambiente di produzione esclusivamente software, al fine di comprovare la teoria fin qui sostenuta: modelli e framework Agili si prestano bene in ogni situazione, basta capire attraverso un'attenta analisi cosa e come farlo[52].

[51] AUTC, *AUTC-Area Edilizia e Sostenibilità*, Università di Bologna, in `https://www.unibo.it/it/ateneo/organizzazione/amministrazione-generale/10900`.

[52] Lo stesso Anderson, esplicita nel testo come abbia trovato incredibile l'utilizzo di Kanban anche per la gestione degli ingressi e uscite nei Giardini orientali al palazzo imperiale a Tokyo, durante una visita di piacere e per puro caso: vedi D. J. Anderson, Kanban..., *cit*, p.35-36.

La realizzazione di questo progetto richiede tre principali fasi (delle quali peraltro solo la prima è stata finora svolta):

- Analisi: si tratta di una fase di *inception*[53], dove ci siamo focalizzati nello studio dei bisogni dei committenti e nel produrre le specifiche dei requisiti in modo chiaro, più complete possibili e funzionali, utilizzando varie tecniche per far emergere tutti i bisogni e i rischi legati al progetto, così da consentire una visione comune o condivisa di una soluzione.

- Implementazione: è la fase di *construction*[54], che dovrebbe portarci ad introdurre determinate conoscenze agli elementi del team nonché strumenti a supporto del lavoro quotidiano, sia per la gestione operativa sia per il flusso di produzione.

- Validazione: è la fase di *validation*; si occupa di verificare i risultati ottenuti nella implementazione effettuata nel lungo periodo, dove si ha la possibilità di verificare i risultati del modello implementato, il quale necessita di molti dati alla mano per essere ben valutato e infine comunicato al committente.

In questo capitolo tratteremo la prima fase, quella di analisi.

4.0.1 La scelta della tematiche

Sebbene sia stato chiarito fin da subito lo scope del progetto e il risultato finale atteso, durante la fase di analisi è stato necessario capire il tipo di tematiche[55] da analizzare al fine di rimanere nella dimensione progettuale definita con il committente e soddisfare le aspettative.

In un primo momento, l'obiettivo è stato quello di analizzare il tipo di team con il quale ci stavamo approcciando; questa parte è stata di fondamentale importanza poiché solo conoscendo il tipo di lavoro che gli elementi del team svolgono, le commesse delle quali si occupano e gli strumenti a supporto a loro disposizione, nonché il numero di stakeholder interessati, siamo stati in grado di identificare il tipo di approccio Agile che più rispecchiava il loro interesse, ma che ancor più potesse realmente aiutarli a migliorare la qualità del lavoro e della vita lavorativa.

[53]Con il termine *inception* si intende la produzione di una visione condivisa degli obiettivi e dei contenuti del progetto.

[54]Con il termine *construction* si intende l'obiettivo di produrre il sistema, l'architettura o il modello, sfruttando i requisiti precedentemente raccolti.

[55]Con il termine *tematiche* vogliamo identificare tutte le sottodimensioni che ci interessa conoscere al fine di promuovere il modello Agile e innestarlo all'interno dell'area interessata, rendendolo scalabile laddove necessario.

Attraverso questa analisi preliminare abbiamo estratto delle tematiche di interesse che sono state oggetto del survey (elemento del quale parleremo nei successivi capitoli), realizzato al fine di conoscere le sensazioni del team sul lavoro, soprattutto in questa fase emergenziale (COVID-19), e sugli strumenti dei quali dispongono.

L'elaborato finale ha prodotto risultati riconducibili ai seguenti temi:

- Parità di genere/genitorialità
- Rispetto ambientale
- Produttività
- Ammodernamento
- Attrattività delle competenze
- Modalità condivisione conoscenza implicita
- Strumenti condivisione conoscenza
- Strumenti organizzazione del lavoro
- Adeguatezza strumenti
- Portabilità delle pratiche (da tradizionale a smart working)
- Intrusività strumenti di IM/chat/videoconferenza
- Regole chiare per l'uso degli strumenti IM/chat/videoconferenza
- Visibilità lavoro proprio
- Visibilità lavoro altrui
- Organizzazione dei meeting
- Appropriatezza regole e pratiche generali/specifiche
- Lavoro agile e responsabilizzazione
- Rischio burocratizzazione

Anche se a prima vista possono sembrare tante e alcune fuori luogo (come ad esempio la parità di genere, la genitorialità, la burocratizzazione, etc.), in realtà tutte le tematiche selezionate sono state meticolosamente analizzate al fine progettuale; ricordiamo infatti che i modelli Agili si basano su un approccio prettamente orientato al benessere dell'individuo[56].

4.0.2 La suddivisione in sezioni

La fase successiva è stata cercare di suddividere le domande elaborate, secondo le tematiche analizzate, in modo tale da costruire un survey lineare e che mantenesse una determinata scaletta; questo al fine di renderlo meno noioso possibile e al tempo stesso dare una sensazione di appagamento, garantita da un'alta considerazione del soggetto attraverso domande denotanti interesse al miglioramento della sua vita lavorativa in questa particolare situazione e anche a emergenza conclusa.

La suddivisione delle domande è stata così predisposta:

- Introduzione
- Background personale
- Lavoro Agile in generale
- Lavoro Agile in UniBO
- Lavoro Agile: strumenti e pratiche

In questo paragrafo analizzeremo ognuna di queste suddivisioni.

Introduzione

Si tratta della sezione più importante.

Per alcuni può sembrare quasi assurda l'affermazione di cui sopra; tuttavia, secondo studi noti sul tema del comportamento del consumatore, è stato provato che una descrizione del survey dettagliata e curata, rispondente ad ogni perplessità sul tema affrontato, abbassa in maniera drastica il tasso di abbandono dello stesso.

Diventa fondamentale dunque fornire una descrizione chiara di quello che l'individuo sta per compiere, a prescindere dalla rilevanza che esso riveste per chi crea il survey.

[56]Come abbiamo visto, il Manifesto Agile recita: "Gli individui e le interazioni più che i processi e gli strumenti".

Se si vuole ottenere un risultato attendibile sul pensiero reale di ogni partecipante, nonché un numero sufficiente per essere valutato statisticamente, è fondamentale cercare di massimizzare il numero di rispondenti; in caso contrario, questo può portare a una pessima valutazione delle risposte e di conseguenza a una pessima risoluzione del problema, con conseguente fallimento del progetto.

Nella Figura 4.1 è possibile vedere la realizzazione compiuta[57].

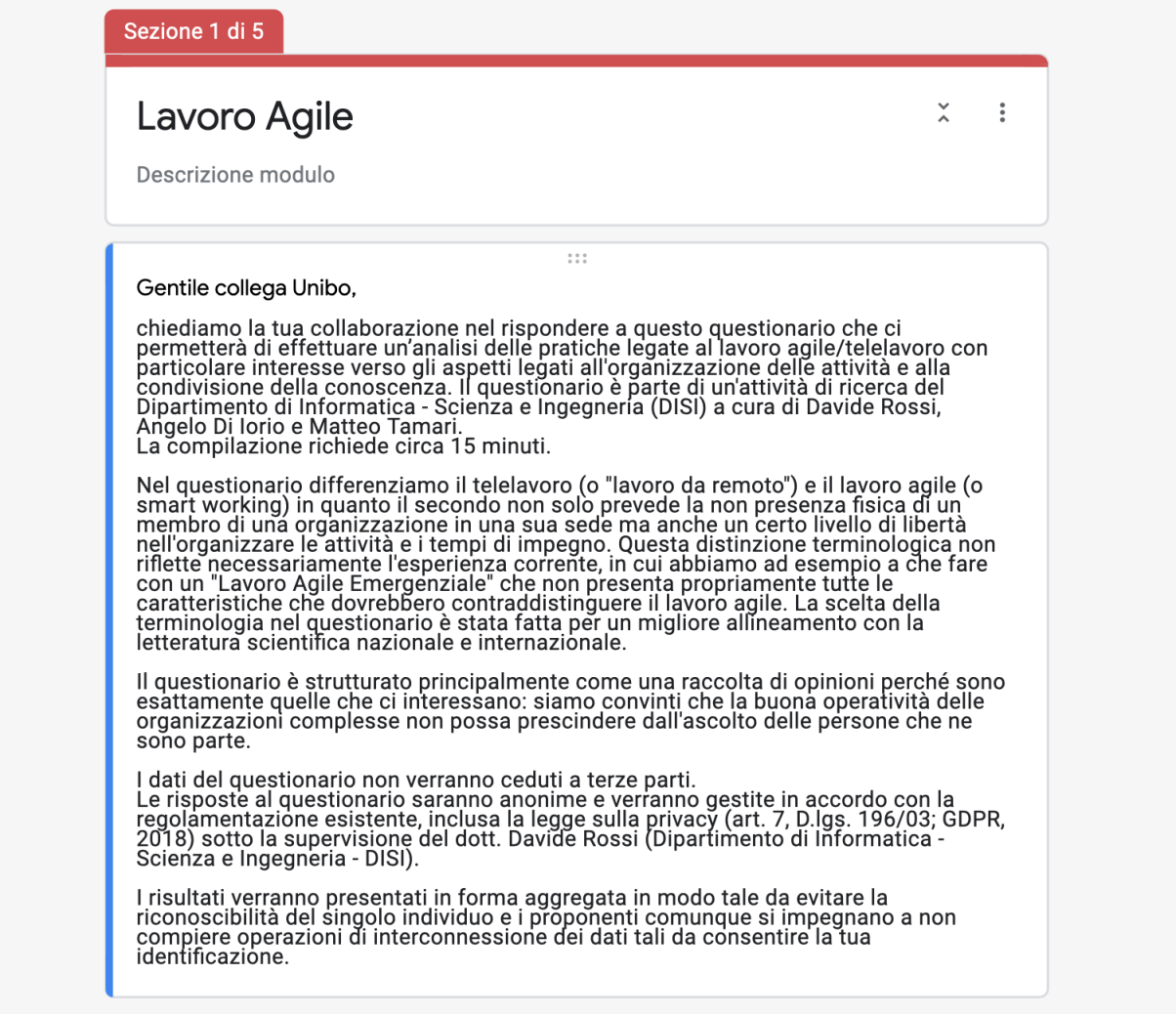


Figura 4.1: Introduzione survey Lavoro Agile

Background personale

Al fine di strutturare un'implementazione Agile corretta e che meglio si identifica nel Dipartimento, è stata strutturata una sezione dedicata alle capacità personali, sia tecniche sia di know-how, orientata al singolo individuo e alla sua esperienza nel settore.

[57]Prof. D. Rossi, *Lavoro Agile*, 2020, Bologna, p. 1 del survey.

Con capacità tecniche intendiamo tutto lo spazio di competenze nell'utilizzo di sistemi software e hardware. Con know-how intendiamo tutte le competenze di conoscenza in materia di gestione e pianificazione del lavoro.

Questo è stato necessario per catturare il livello medio attuale del Dipartimento; in questo modo siamo stati in grado, seppur per ora parzialmente, di capire il punto di partenza e fino a dove potremmo spingerci, valutando così non solo il tipo di approccio Agile, ma anche gli strumenti a supporto adatti, più o meno complessi.

Lavoro Agile in generale

La terza sezione del survey ha come scopo chiarire il pensiero personale sullo smart working e non mirato a quello in UniBO.

La ragione per la quale ci siamo interessati all'opinione dei singoli individui sul tema generale riposa su due fattori: il primo è che si fornisce un senso di considerazione all'individuo stesso e questo lo conforta, oltre a far capire che il lavoro svolto è nel suo interesse; il secondo è ricevere una informazione molto importate al fine di capire se le persone con le quali dobbiamo lavorare per migliorare un processo si sentono a loro agio nel contesto. Se le informazioni medie risultano positive, il lavoro sarà più semplice per tutti.

Lavoro Agile in UniBO

Sebbene potrebbe apparire una sezione per certi versi ripetitiva della precedente, in realtà la quarta sezione ha lo scopo di valutare l'opinione personale dell'individuo sulla propria situazione attuale in smart working nello specifico ambiente di lavoro del Dipartimento e sui benefici o meno che questa emergenza ha portato.

È quindi diverso l'approccio mentale che, da una linea generale o generalizzata, passa a quella specifica e personale che il singolo individuo vive tutti i giorni durante l'orario di lavoro.

Lavoro Agile: strumenti e pratiche

La quinta ed ultima sezione ha lo scopo di ottenere una valutazione personale sugli strumenti e le pratiche a supporto dello smart working e dell'uso che ne viene fatto.

Le tematiche di maggior interesse e sulle quali abbiamo cercato di orientare questa parte del survey sono quelle relative all'efficacia ed efficienza degli strumenti e delle pratiche che ne definiscono l'uso, con un particolare occhio di riguardo alla capacità

di condividere informazioni rilevanti nei team di lavoro senza diventare eccessivamente invadenti.

4.0.3 La creazione del survey

La creazione del survey ha richiesto uno studio approfondito su come impostare le domande e quale tool utilizzare.

Abbiamo scelto di lasciare la possibilità a ciascun partecipante di scrivere in ogni sezione un approfondimento sul tema trattato, al fine di catturare qualche punto che nella strutturazione ci era sfuggito.

Questo elemento è stato fondamentale poiché ci ha permesso di evidenziare criticità che non erano state inserite (per essere valutate) in fase di sviluppo del survey.

Come strumentazione ci siamo avvalsi dei tool forniti da Google, ovvero:

- Google Form: per la creazione e distribuzione del survey
- Google Docs: per la strutturazione delle domande
- Goolge Sheet: per l'analisi al fine evidenziare le dimensioni e domande ripetute

Sebbene ci siano tool a supporto decisamente più ottimizzati e specifici, per velocità e facilità di gestione ci siamo orientati su tali strumenti che oltretutto offrono la possibilità di conservare tutto su cloud, con accesso da parte di tutti, sempre e ovunque.

Siamo Agili: il risultato viene prima di ogni strumento!

Google Form

Questo tool è stato utilizzato per il fine ultimo, la distribuzione del survey; la scelta è ricaduta su di esso dopo un'attenta analisi su quale fosse il migliore per lo scopo: infatti è free, dinamico e altamente personalizzabile (vedi Figura 4.1).

Google Docs

Attraverso questo tool, abbiamo prodotto il documento più utilizzato al fine di creare domande collegate allo stessa tema, non torbide e le risposte alle quali ci fornissero una chiara visione dell'insieme[58].

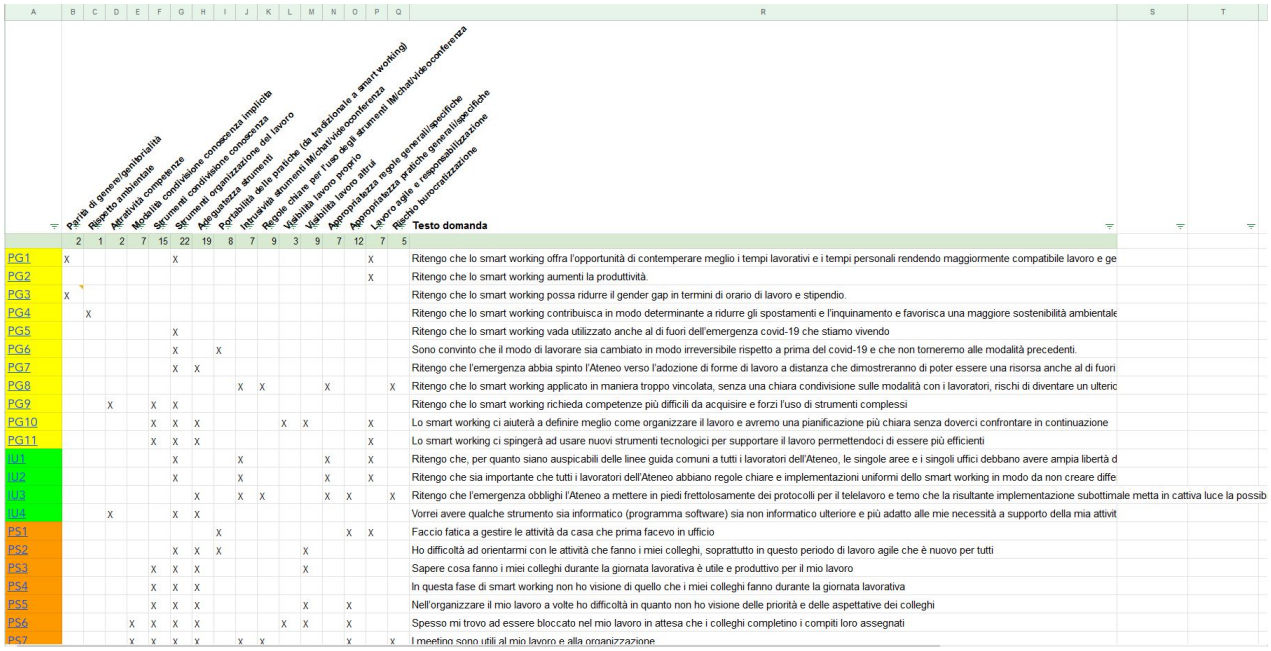
Tali domande sono state ulteriormente numerate e suddivise, con lo scopo essere rilevate all'interno dello "sheet", descritto nel successivo paragrafo.

[58] Al seguente link `https://docs.google.com/document/d/1NSS29bszOndcVt_xsj0jZqiCsbsg11LZBUF0grydd_o/edit?usp=sharing` è possibile visualizzare il documento e la sua struttura.

Google Sheet

Lo “sheet” o ”foglio di calcolo” è un documento di lavoro rilasciato anch’esso dalla suite Google che permette di emulare le funzioni di Excel.

L’utilizzo di questo strumento ci è servito per creare una matrice di confronto (vedi Figura 4.2) tra il testo della domanda (che veniva proposta nel survey) e le varie tematiche che abbiamo selezionato; questo al fine di valutare se tutte quelle domande avevano molti luoghi comuni ed eventualmente cercare di alleggerirle o accorparle ad altre già presenti.



	Parità di genere/genitorialità	Rispetto ambientale	Attività competenze	Modalità condivisione conoscenza implicita	Strumenti condivisione conoscenza	Strumenti organizzazione del lavoro	Adeguatezza strumenti	Portabilità delle pratiche (da tradizionale a smart working)	Intuitività strumenti IM/chat/videoconferenza	Regole chiare per l'uso degli strumenti IM/chat/videoconferenza	Visibilità lavoro proprio	Appropriatezza pratiche generali/specifiche	Appropriatezza regole generali/specifiche	Lavoro agile e responsabilizzazione	Rischio burocratizzazione	Testo domanda
	2	1	2	7	15	22	19	8	7	9	3	9	7	12	7	5	
PG1	X					X								X			Ritengo che lo smart working offra l'opportunità di contemperare meglio i tempi lavorativi e i tempi personali rendendo maggiormente compatibile lavoro e ge
PG2														X			Ritengo che lo smart working aumenti la produttività.
PG3	X																Ritengo che lo smart working possa ridurre il gender gap in termini di orario di lavoro e stipendio.
PG4		X															Ritengo che lo smart working contribuisca in modo determinante a ridurre gli spostamenti e l'inquinamento e favorisca una maggiore sostenibilità ambientale
PG5						X											Ritengo che lo smart working vada utilizzato anche al di fuori dell'emergenza covid-19 che stiamo vivendo
PG6						X		X									Sono convinto che il modo di lavorare sia cambiato in modo irreversibile rispetto a prima del covid-19 e che non torneremo alle modalità precedenti.
PG7						X	X										Ritengo che l'emergenza abbia spinto l'Ateneo verso l'adozione di forme di lavoro a distanza che dimostreranno di poter essere una risorsa anche al di fuori
PG8								X	X			X			X		Ritengo che lo smart working applicato in maniera troppo vincolata, senza una chiara condivisione sulle modalità con i lavoratori, rischi di diventare un ulteric
PG9			X		X	X											Ritengo che lo smart working richieda competenze più difficili da acquisire e forzi l'uso di strumenti complessi
PG10					X	X	X				X	X			X		Lo smart working ci aiuterà a definire meglio come organizzare il lavoro e avremo una pianificazione più chiara senza doverci confrontare in continuazione
PG11					X	X	X								X		Lo smart working ci spingerà ad usare nuovi strumenti tecnologici per supportare il lavoro permettendoci di essere più efficienti
IU1						X			X				X		X		Ritengo che, per quanto siano auspicabili delle linee guida comuni a tutti i lavoratori dell'Ateneo, le singole aree e i singoli uffici debbano avere ampia libertà d
IU2						X			X				X		X		Ritengo che sia importante che tutti i lavoratori dell'Ateneo abbiano regole chiare e implementazioni uniformi dello smart working in modo da non creare diffe
IU3							X		X	X			X	X		X	Ritengo che l'emergenza obblighi l'Ateneo a mettere in piedi frettolosamente dei protocolli per il telelavoro e temo che la risultante implementazione subottimale metta in cattiva luce la possib
IU4			X			X	X										Vorrei avere qualche strumento sia informatico (programma software) sia non informatico ulteriore e più adatto alle mie necessità a supporto della mia attivit
PS1								X						X	X		Faccio fatica a gestire le attività da casa che prima facevo in ufficio
PS2						X	X	X				X					Ho difficoltà ad orientarmi con le attività che fanno i miei colleghi, soprattutto in questo periodo di lavoro agile che è nuovo per tutti
PS3					X	X	X					X					Sapere cosa fanno i miei colleghi durante la giornata lavorativa è utile e produttivo per il mio lavoro
PS4					X	X	X										In questa fase di smart working non ho visione di quello che i miei colleghi fanno durante la giornata lavorativa
PS5					X	X	X					X		X			Nell'organizzare il mio lavoro a volte ho difficoltà in quanto non ho visione delle priorità e delle aspettative dei colleghi
PS6				X	X	X	X				X	X		X			Spesso mi trovo ad essere bloccato nel mio lavoro in attesa che i colleghi completino i compiti loro assegnati
PS7				X	X	X	X		X	X				X		X	I meeting sono utili al mio lavoro e alla organizzazione

Figura 4.2: Survey coverage Lavoro Agile

Tutti e tre i collaboratori del progetto, Prof. Davide Rossi, Prof. Angelo Di Iorio e il sottoscritto Matteo Tamari, hanno creato una propria matrice e inserito, secondo una propria linea guida, ogni tematica all’interno di una o più domande.

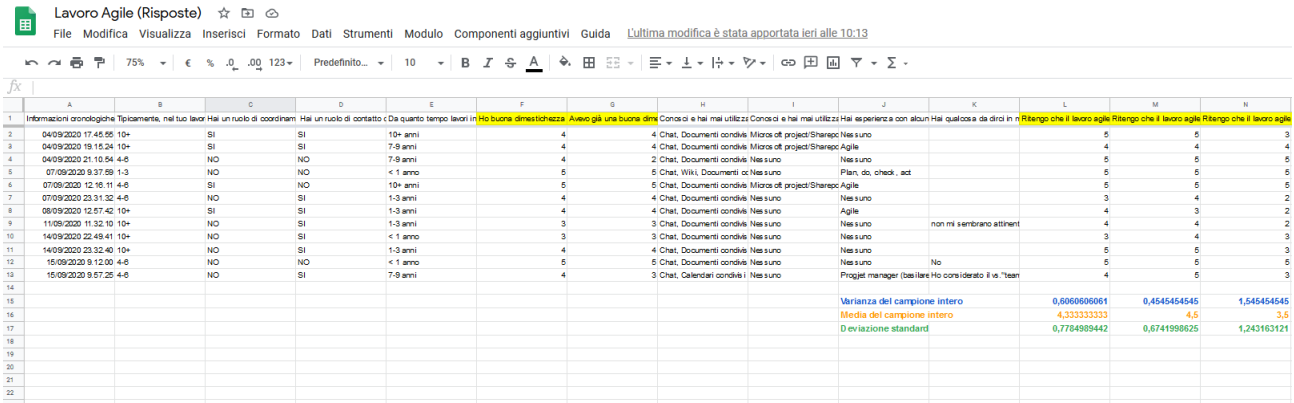
Il fine ultimo è stato quello di valutare in un tavolo di confronto se eravamo o meno concordi sulle dimensioni ed eventualmente sulla rimozione di una o più domande che si ripetevano con la stessa tematica; il risultato ci ha permesso di snellire il survey, portandolo ad un minor tempo di completamento per l’utente.

4.0.4 L'analisi dei dati

Come abbiamo già evidenziato in precedenza, il campione analizzato è composto da sedici elementi appartenenti al Dipartimento interessato.

Sebbene Google Form offra già al suo interno la possibilità di compiere un'analisi dettagliata del survey, che tipicamente approfondisce la percentuale di risposte alle domande fatte e il punteggio che per esempio si attribuisce ad una di queste, abbiamo optato per cercare di analizzare il dato rilevato approfonditamente, sfruttando la statistica e le sue metriche di misura.

Si è quindi optato per estrarre i dati ricevuti in singole risposte (del tutto anonime) e riportarle successivamente su un foglio di calcolo di Google Sheet (vedi Figura 4.3); ciò ci ha permesso di condividerli istantaneamente e di poterci lavorare insieme o contemporaneamente alla loro elaborazione.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	Informazioni cronologiche	Tipicamente, nel tuo lavor	Hai un ruolo di coordinam	Hai un ruolo di contatto c	Da quanto tempo lavori in	Ho buona dimestichezza	Avevo già una buona dime	Conosci e hai mai utilizza	Conosci e hai mai utilizza	Hai esperienza con alcun	Hai qualcosa da dirci in r	Ritengo che il lavoro agile	Ritengo che il lavoro agile	Ritengo che il lavoro agile
2	04/09/2020 17.45.55	10+	SI	SI	10+ anni	4	4	Chat, Documenti condivis	Micros oft project/Sharepc	Nessuno		5	5	3
3	04/09/2020 19.15.24	10+	SI	SI	7-9 anni	4	4	Chat, Documenti condivis	Micros oft project/Sharepc	Agile		4	4	4
4	04/09/2020 21.10.54	4-6	NO	NO	7-9 anni	4	2	Chat, Documenti condivis	Nessuno	Nessuno		5	5	5
5	07/09/2020 9.37.59	1-3	NO	NO	< 1 anno	5	5	Chat, Wiki, Documenti co	Nessuno	Plan, do, check , act		5	5	5
6	07/09/2020 12.16.11	4-6	SI	NO	10+ anni	5	5	Chat, Documenti condivis	Micros oft project/Sharepc	Agile		5	5	5
7	07/09/2020 23.31.32	4-6	NO	SI	1-3 anni	4	4	Chat, Documenti condivis	Nessuno	Nessuno		3	4	2
8	08/09/2020 12.57.42	10+	SI	SI	1-3 anni	4	4	Chat, Documenti condivis	Nessuno	Agile		4	3	2
9	11/09/2020 11.32.10	10+	NO	SI	1-3 anni	3	3	Chat, Documenti condivis	Nessuno	Nessuno	non mi sembrano attinent	4	4	2
10	14/09/2020 22.49.41	10+	NO	SI	< 1 anno	3	3	Chat, Documenti condivis	Nessuno	Nessuno		3	4	3
11	14/09/2020 23.32.40	10+	NO	SI	1-3 anni	4	4	Chat, Documenti condivis	Nessuno	Nessuno		5	5	3
12	15/09/2020 9.12.00	4-6	NO	NO	< 1 anno	5	5	Chat, Documenti condivis	Nessuno	Nessuno	No	5	5	5
13	15/09/2020 9.57.25	4-6	NO	SI	7-9 anni	4	3	Chat, Calendari condivisi	Nessuno	Progjet manager (basilare	Ho considerato il vs."tean	4	5	3
14														
15										Varianza del campione intero		0,6060606061	0,4545454545	1,545454545
16										Media del campione intero		4,333333333	4,5	3,5
17										Deviazione standard		0,7784989442	0,6741998625	1,243163121
18														
19														
20														
21														
22														

Figura 4.3: Tabella Google Sheet per le risposte al survey

Per dare una valida e ragionevole spiegazione ai dati ricevuti, abbiamo sfruttato due metriche di valutazione: la *media*[59] e la *deviazione standard* [60]. Grazie a queste, siamo stati in grado di giustificare le nostre conclusioni e proporre rimedi consoni alla situazione.

A fronte di questa analisi sono emersi diversi aspetti interessati che saranno chiariti nei paragrafi successivi.

[59]Con *media* in statistica intendiamo il rapporto tra la somma dei dati numerici e il loro numero complessivo.

[60]Con *deviazione standard* in statistica si intende la dispersione del campione analizzato rispetto alla media evidenziata; più questa metrica è grande, più il campione è disperso, mentre, viceversa, meno è elevata e minore sarà la dispersione.

Il campione

A chiusura del tempo stabilito per la compilazione del survey abbiamo raccolto dodici rispondenti su sedici previsti; ci aspettavamo che non tutti avrebbero risposto (e.g. a causa di ferie o malattie) e sapevamo fin da subito che per ottenere un campione valido ai fini statistici e dell'osservazione, e quindi poter svolgere al meglio il nostro lavoro, necessitavamo di ottenere almeno l'80% delle risposte previste: così è stato.

Essendo i rispondenti inquadrati all'interno del Dipartimento con qualifiche e responsabilità differenti, creare un solo campione era insufficiente; l'obiettivo infatti non era solo fornire un'analisi strutturale generalizzata, ma considerare tutte le differenze e consegnare qualcosa di molto dettagliato, con lo scopo di individuare le problematiche e proporre soluzioni ottimali, al fine di un miglior risultato, come avvenuto per l'uso delle metriche.

La suddivisione in sottocampioni

Come detto, i rispondenti erano dislocati in diverse strutture e con compiti differenti, al fine di coprire quante più aree possibili del Dipartimento; questo ci ha permesso di ottenere un quadro iniziale sulle varie situazioni che, come volevasi dimostrare, erano molto dinamiche.

Una volta ottenuto il quadro iniziale, abbiamo optato per la suddivisione in sottocampioni nel seguente modo (vedi Figura 4.4):

- **Totale**: si tratta dell'intero campione senza alcuna distinzione.
- **Responsabili di area**: abbiamo separato i rispondenti che hanno un ruolo di responsabilità da chi al contrario non lo possiede.
- **Anzianità di servizio**: questa particolare divisione ha richiesto più modifiche nel tempo. Inizialmente eravamo dell'idea di dividere i rispondenti nei rispettivi intervalli di risposta (minore di 1 anno, 1-3 anni, 4-6 anni, 7-9 anni, maggiore di 10 anni); tuttavia, durante l'analisi, ci siamo accorti che non avevamo rispondenti rispecchiantesi nell'intervallo di anzianità di servizio 4-6 anni e che in linea generale quelli appartenenti ai 7-9 anni e maggiore di 10 anni avevano similitudini nelle risposte, così come come quelli minori di 1 anno e tra 1-3 anni: sicuramente un dato dettato dall'anzianità di servizio. Per questa ragione abbiamo scelto di raggruppare i primi nel gruppo maggiore di 5 anni di anzianità (4-6, 7-9, maggiore di 10) e i secondi nel gruppo minore di 5 anni di anzianità (minore di 1 anno, 1-3 anno); così

facendo abbiamo aggregato i dati e reso il campione in analisi molto più leggibile da un punto di vista statistico.

- **Ruoli di contatto con pubblico o persone esterne**: abbiamo differenziato chi si interfaccia con persone esterne al Dipartimento o ufficio da chi non lo fa. Questo ci serviva per comprendere che tipo di rapporto con gli esterni era presente ed eventualmente trovare una soluzione idonea che tenesse in considerazione tale fattore.

Figura 4.4: Grafico statistico sulla produttività rispetto il Lavoro Agile

Lo studio ha richiesto la creazione di diversi fogli di lavoro (Google Sheet), tutti con al loro interno *media* e *deviazione standard*; alla fine sono stati ottenuti grafici di confronto per documentare ai committenti quanto appreso, dando così loro la possibilità di valutare le nostre proposte in base ai risultati.

La documentazione del risultato

Una volta conclusa la suddivisione delle sezioni e la compilazione delle metriche all'interno dei fogli di lavoro, abbiamo scelto di utilizzare sempre la suite dei tool di lavoro di Google per la creazione della presentazione: Google Presentation.

A fronte di specifiche domande sul tema del Lavoro Agile che interessavano in particolar modo i nostri committenti, abbiamo optato per creare una presentazione basata su grafici; essi mostravano chiaramente il risultato ottenuto, grazie all'utilizzo di media e deviazione standard debitamente inserita nella immagini, consentendo ai committenti di poterlo visualizzare graficamente, valutando in maniera oggettiva le nostre proposte,

le quali infine, dopo una attenta analisi, sono state frutto di una valutazione condivisa tra le parti.

Il focus è stato infatti presentare un problema, visivo in ogni sua parte, fornendo così una reale soluzione giustificata dal dato stesso; ciò ci ha permesso di far "toccare con mano" ai committenti il problema e far comprendere loro la necessità di un intervento.

Il risultato è stato quello di cercare soluzioni vantaggiose e continuative nel tempo, poiché l'emergenza vissuta potrebbe ripetersi, anche in altri contesti, se non addirittura proseguire per un tempo non definito; dunque cercare di modificare parte della struttura organizzativa è utile sia nell'immediato sia per arrivare pronti ad un ulteriore ostacolo. È di vitale importanza consentire una continuità nel lavoro e nel servizio che questo Dipartimento eroga ai suoi clienti.

Conclusioni

In questa tesi sono stati affrontati nei primi capitoli due tra i principali modelli e framework Agili, Scrum e Kanban, con l'obiettivo di analizzarli approfonditamente al fine di inquadrare le condizioni migliori di applicazione dell'uno o dell'altro.

In primo luogo le dimensioni di architettura del progetto, che possono essere per certi versi molto differenti; si tratta infatti di un elemento di analisi chiave, che abbiamo visto essere determinato dalle diverse nature della committenza, dalle cadenze, come anche dalle metodologie di rilascio dell'incremento, dagli eventi e dai ruoli, i quali sono chiaramente diversi tra i due.

Sia Scrum sia Kanban sono stati creati per arricchire la produttività e agevolare l'esperienza di sviluppo al team; lo scopo è identico, la bandiera Agile è la stessa, ma la via per la realizzazione si è dimostrata diversa.

In riferimento a quest'ultimo aspetto, secondo gli studi è emerso che, quando parliamo di Scrum, ci riferiamo ad un framework iterativo incrementale dove i rilasci avvengono in un intervallo di tempo ben preciso; mentre, quando parliamo di Kanban, ci riferiamo ad un modello di Continuos Flow ovvero a rilascio continuo, in ogni momento (e.g. anche una volta al giorno).

Queste differenze sono di estrema importanza poiché permettono ad un team di decidere quale scegliere a seconda della natura della committenza e del prodotto che si sta andando a sviluppare; in questa fase, un errore di interpretazione nella chiave di lettura del progetto può rivelarsi fatale. Riuscire a interpretare nel dettaglio le dimensioni del prodotto che si vuole sviluppare e ciò che il committente si aspetta, è fondamentale.

Nella esposizione di Scrum e Kanban abbiamo inoltre approfondito gli strumenti che vengono messi a disposizione dei team di produzione.

È emerso che, seppur strutturalmente e concettualmente differenti, Scrum e Kanban hanno delle analogie più o meno marcate; in particolare ci riferiamo alla board, sia fisica sia virtuale, che è un elemento fondamentale per il corretto funzionamento di entrambi.

Un interessante aspetto derivante da questo studio è il modo nel quale questo strumento si approccia alla produzione; se per Kanban si tratta di un riferimento al Continuos

Delivery al fine di non fermarsi mai, per Scrum è un modo di organizzare i task al fine di completare con successo lo Sprint e passare così alla iterazione successiva. In entrambi i casi, la board svolge anche una funzione di visualizzazione del lavoro in un preciso intervallo di tempo e, nello stesso momento, di massimizzare la produzione e il rilascio di valore continuo al committente.

Comunque non è possibile delineare una perfetta guida alla scelta di utilizzo di Scrum o Kanban; ogni team deve valutare in base alla propria esperienza e maturità lavorativa, analizzando prima le dimensioni del prodotto da sviluppare, per poi orientarsi sulla decisione.

Sicuramente il modello Kanban ha una curva di apprendimento inferiore, al contrario del framework Scrum; il che, comportando una semplificazione nella sua applicazione, tipicamente fa propendere a suo favore soprattutto in presenza di team immaturi o di giovane esperienza.

Al contrario Scrum, che ha una curva di apprendimento più ripida, è consigliato a team più maturi e a strutture aziendali che possono disporre di varie figure che ricoprono diversi ruoli.

Tutto questo non significa che se un team adotta un approccio Kanban allora non potrà mai adottare un approccio Scrum in futuro, ma che ciò necessita di esperienza maturata sul campo, che si consolida solo con il tempo e il lavoro di squadra.

Il progetto Lavoro Agile in collaborazione con il Dipartimento *AUTC-Area Edilizia e Sostenibilità* dell'Università di Bologna ha costituito il banco di prova per sperimentare un'applicazione scalata dei modelli Agili.

Esso è stato l'oggetto dell'ultimo capitolo del presente elaborato ed è tutt'ora in corso d'opera, seguito e monitorato dal sottoscritto Matteo Tamari, dal Prof. Davide Rossi e dal Prof. Angelo Di Iorio; durante la scrittura di questa tesi si trova nella fase finale di *analisi* e, successivamente, sono previste le fasi di *implementazione* del modello Agile e di *validazione* dello stesso.

Essendo un progetto di grandi dimensioni, non siamo ad oggi in grado di stimare una data di conclusione definitiva; tuttavia, sia i committenti del progetto sia noi, siamo molto fiduciosi nella sua riuscita.

Anche se la fase di implementazione è prevista, la sua applicazione non necessariamente sarà effettuata; è infatti importante che vi sia e si accetti un forte cambiamento da parte di tutto il Dipartimento, il quale deve diventare Agile in ogni aspetto, dalla struttura organica alle procedure di lavoro ordinario: ciò non è assolutamente automatico.

L'essere Agili oggi giorno comporta grandi vantaggi e lo scopo di questa tesi è stato anche cercare di evidenziarli; questo richiede però una grande volontà da parte di tutti,

nonché uno sforzo generale non indifferente, che sarà sicuramente ripagato nel lungo periodo, soprattutto nel miglioramento del quotidiano sia per il Dipartimento sia per i suoi collaboratori, che tutti i giorni prestano un grande servizio all'Università di Bologna.

Bibliografia

[1] Scaled Agile, cur. *SAFe 5.0*. 2020. URL: https://www.scaledagileframework.com/.

[2] State Of Agile, cur. *The 14th Annual State of Agile*. 2020. URL: https://explore.digital.ai/state-of-agile/14th-annual-state-of-agile-report.

[3] David J. Anderson. *Kanban: Successful Evolutionary Change*. A cura di Blue Hole Press. 2010.

[4] Atlassian, cur. *Customize cards*. 2019. URL: https://support.atlassian.com/jira-software-cloud/docs/customize-cards/.

[5] Atlassian. «Learn the differences between classic and next-gen projects». In: *Atlassian Support* (2020). URL: https://support.atlassian.com/jira-service-desk-cloud/docs/learn-the-differences-between-classic-and-next-gen-projects/.

[6] Atlassian, cur. *Reporting in JIRA*. 2018. URL: https://confluence.atlassian.com/jirakb/reporting-in-jira-461504615.html.

[7] Atlassian. «Why you need roadmaps in Jira Software». In: *Atlassian Blog* (2018). URL: https://www.atlassian.com/blog/jira-software/roadmaps-in-jira.

[8] Lena Boiser. *Comulative Diagram Flow: Your Most Powerful Tool to Create a More Stable and Predictable Flow*. 2019. URL: https://kanbanzone.com/2019/power-of-cumulative-flow-diagram/.

[9] Lena Boiser. *Kanban Roles for Successful Project Management*. 2019. URL: https://kanbanzone.com/2019/kanban-roles-successful-project-management/.

[10] Eric Brechner. *Agile Software Development with Kanban*. A cura di Microsoft Press. 2015.

[11] Kent Beck Ken Schwaber Jeff Sutherland Robert C. Martin Martin Fowler. *Manifesto for Agile Software Development*. 2001. URL: https://agilemanifesto.org/.

[12] Ikujiro Nonaka Hirotaka Takeuchi. «The New New Development Game». In: *Harvard Business School* (1986).

[13] INOS, cur. *Continuos Delivery: sviluppo di software tramite pipeline*. 2019. URL: https://www.ionos.it/digitalguide/siti-web/programmazione-del-sito-web/continuous-delivery/.

[14] INOS, cur. *Il modello a cascata*. 2019. URL: https://www.ionos.it/digitalguide/siti-web/programmazione-del-sito-web/modello-a-cascata/.

[15] ISO, cur. *ISO/IEC/IEEE 12207:2017 Systems and software engineering - Software life cycle processes*. 2017. URL: https://www.iso.org/standard/63712.html.

[16] Jeff Sutherland Ken Schwaber. *Scrum Guide*. 2017. URL: https://www.scrumguides.org/scrum-guide.html.

[17] Patrick Li. *Jira 5.2 Essential*. A cura di Packt Publishing. 2013.

[18] Max Rehkopf. «User Stories». In: *Atlassian Agile Coach* (2019). URL: https://www.atlassian.com/agile/project-management/user-stories.

[19] Ken Schwaber. *Agile Project Management with Scrum*. A cura di Microsoft Press. 2015.

[20] Jeff Sutherland. *Fare il doppio in metà tempo*. 2015.

Ringraziamenti

Ringrazio il Prof. Davide Rossi e il Prof. Angelo Di Iorio, punti di riferimento chiave in questa tesi, i quali mi hanno insegnato e trasmesso con grande passione la materia dell'ingegneria del software e del mondo Agile, accompagnandomi in uno dei percorsi più importanti della mia vita.

Ringrazio la mia dolce metà, Angela, che mi ha sempre supportato e soprattutto sopportato in ogni momento più buio, credendo sempre in me e dandomi il coraggio di andare avanti.

Ringrazio la mia famiglia e tutti i miei amici che sono sempre stati al mio fianco in questa grande avventura, sostenendomi e incoraggiandomi ogni giorno.

Ringrazio il mio caro amico e compagno di studio Riccardo Mioli, senza il quale oggi non sarei qui e che mi ha accompagnato per tutti questi anni; a lui devo tutti i miei risultati: grazie amico mio.

Un ringraziamento speciale all'Avv. Graziano Mioli, mentore e amico, che mi ha indicato la Via da percorrere ogni giorno e mi trasmette con grande passione l'amore per la cultura; senza di lui la presente tesi non sarebbe la stessa.

Un ultimo ringraziamento più che dovuto alla persona che mi ha spinto ad intraprendere questo meraviglioso percorso universitario, un amico e punto di riferimento, un padre adottivo per certi aspetti, il Dott. Luca Motebugnoli, che ha sempre creduto nelle mie capacità incoraggiandomi nel migliorare ogni giorno.