



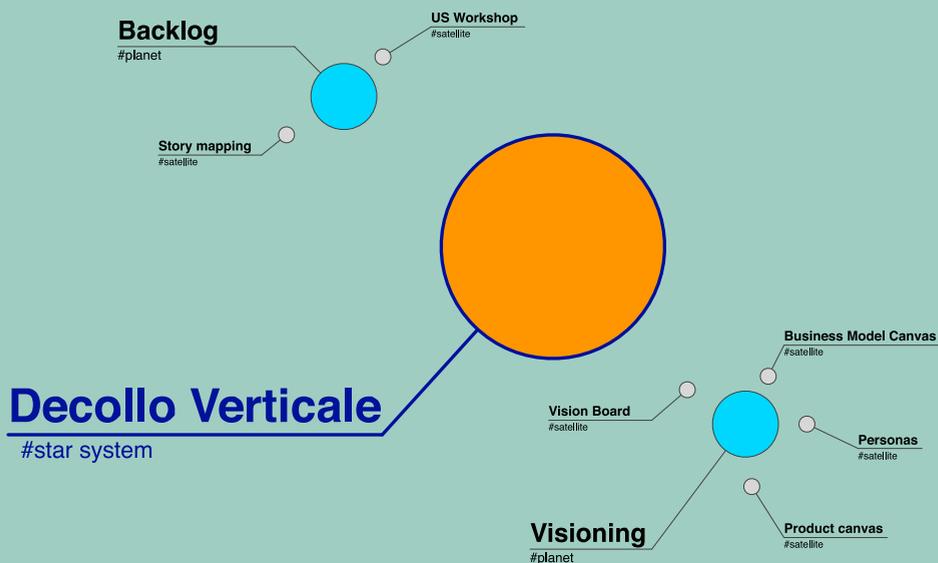


# PARTE II

# DALLA VISIONE

# AL PRODOTTO

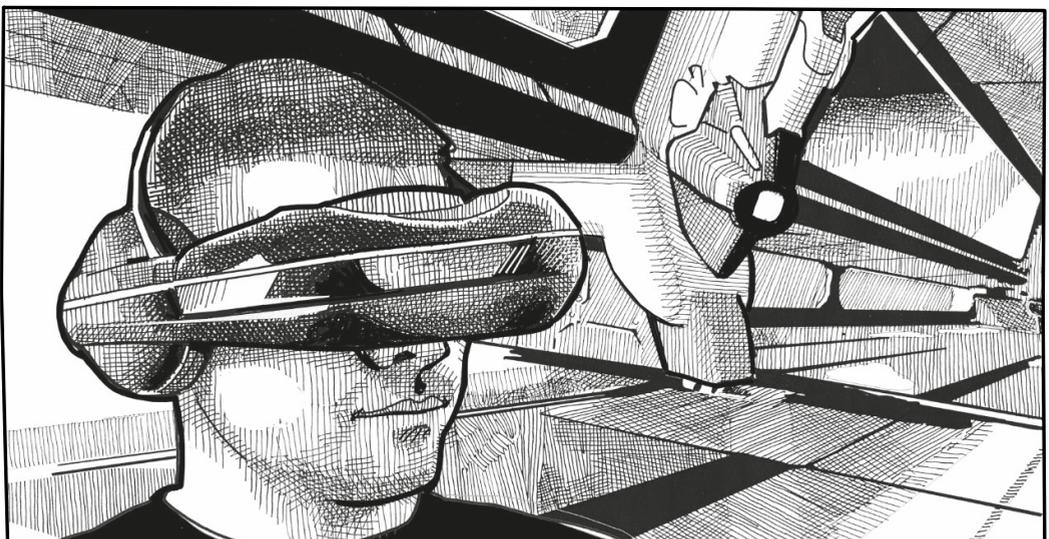
STEFANO LEI - GIULIO ROGGERO - GIOVANNI PULITI



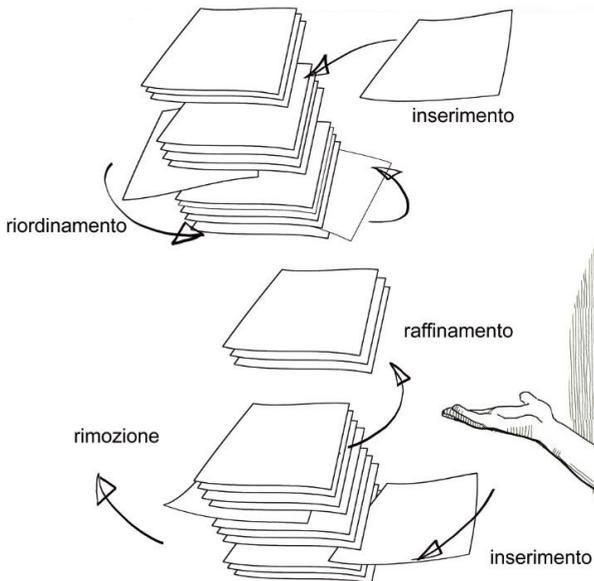
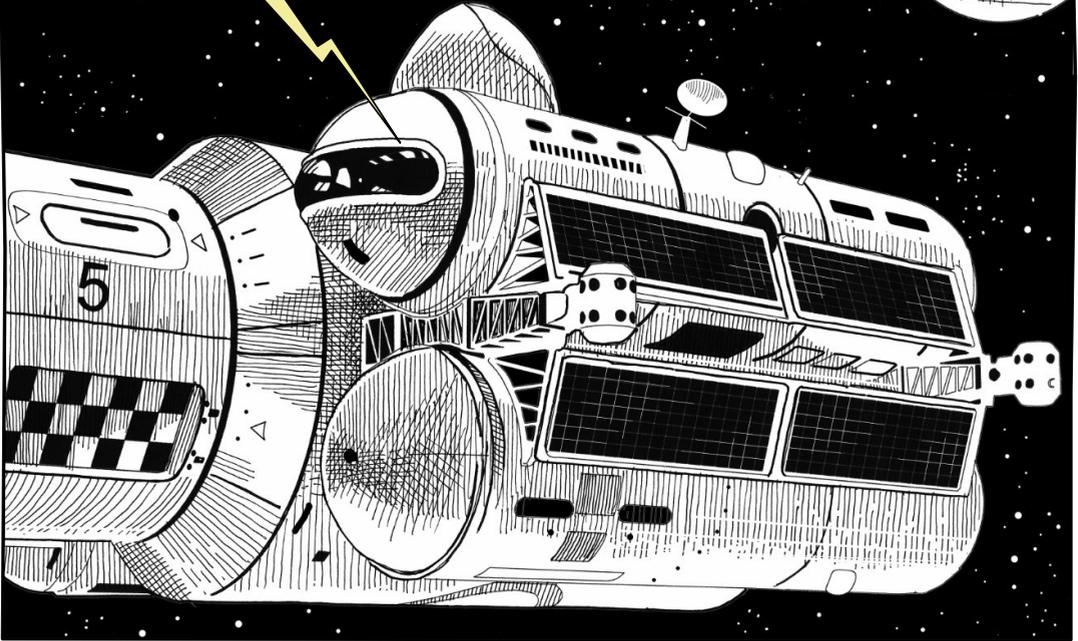


# Parte 2

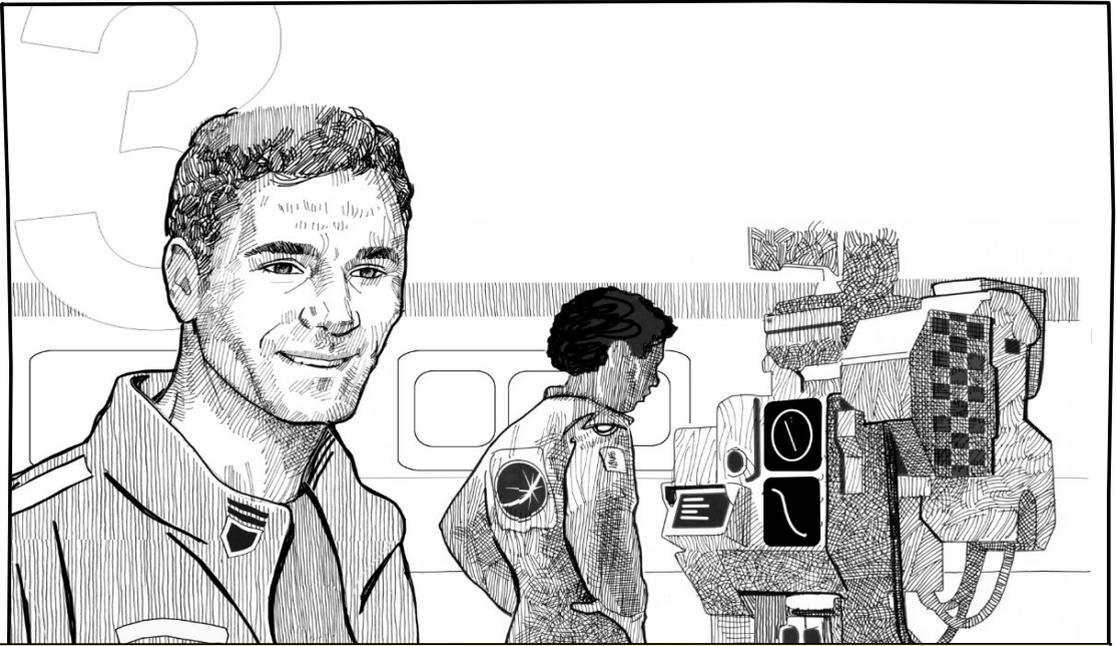
## Dalla visione al prodotto



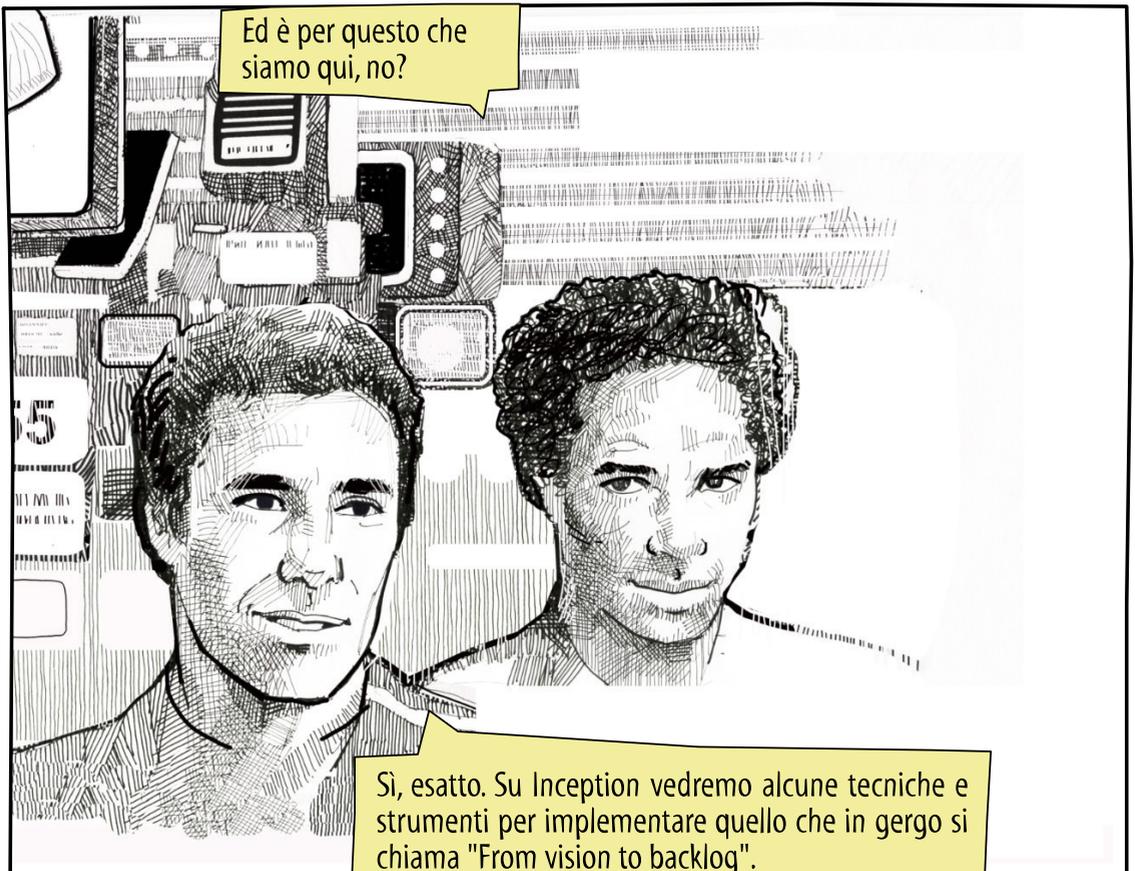
Il mio dubbio è: "Come arrivo ad avere il backlog pronto per la lavorazione?"



Dubbio legittimo: in Scrum, per esempio, si parla di raffinamento del backlog, ossia di quel processo evolutivo continuativo che permette di avere sempre cose lavorabili in testa alla pila...



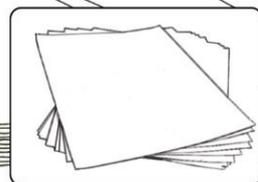
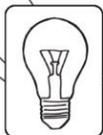
Scrum, ma anche altre metodologie, come Kanban, non si preoccupano della fase iniziale di popolamento e di alimentazione del backlog. Il modo in cui arrivare alla prima versione di questa raccolta delle cose da fare messe in "pila" esula dagli scopi di Scrum...



Ed è per questo che siamo qui, no?

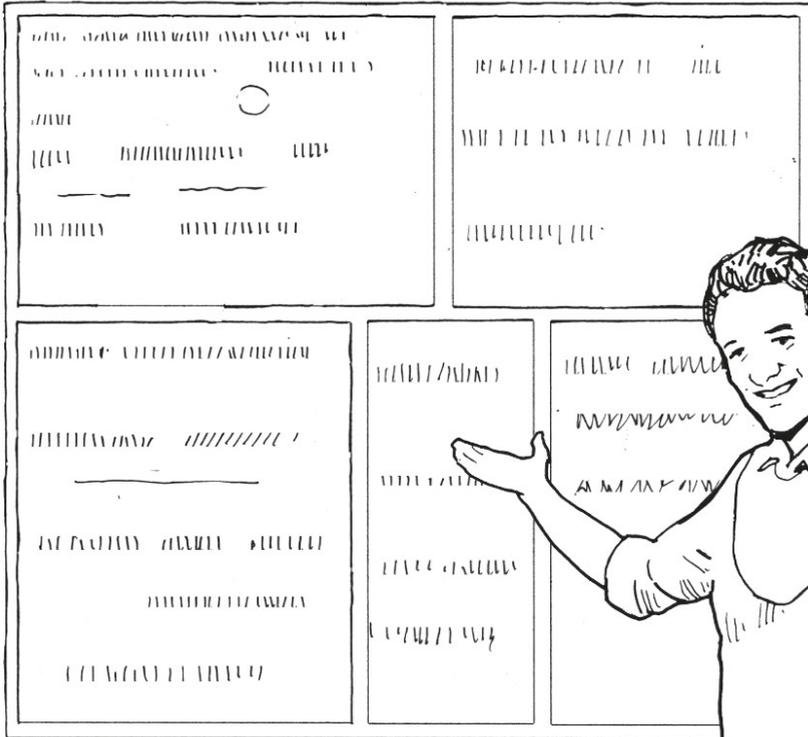
Sì, esatto. Su Inception vedremo alcune tecniche e strumenti per implementare quello che in gergo si chiama "From vision to backlog".

Questo flusso può essere formalizzato in modo efficace secondo questo schema.

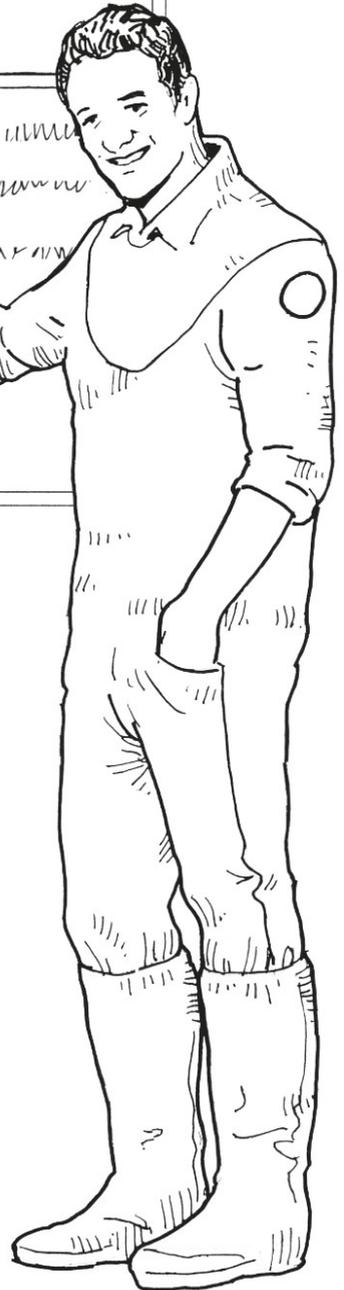


Realizzare un prodotto vuol dire affrontare questi tre aspetti: il mercato e i bisogni utente, i fattori differenzianti del prodotto e i bisogni dell'azienda che lo produce.

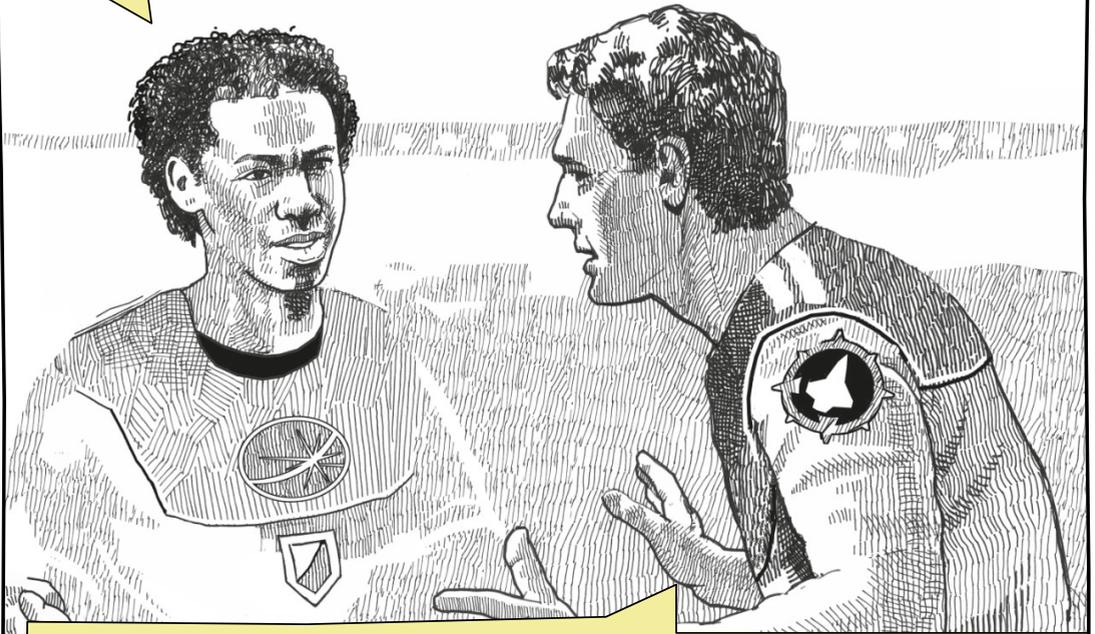




Grazie a una serie di strumenti come la Vision Board, il Business Model Canvas, lo Story Mapping o le Impact Map, possiamo lavorare in modo conciso e sintetico su questo compito.

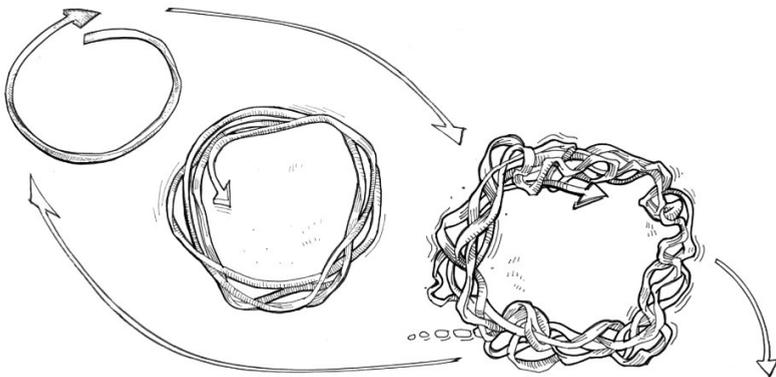


È vero che questo processo è molto più rapido rispetto al processo tradizionale?

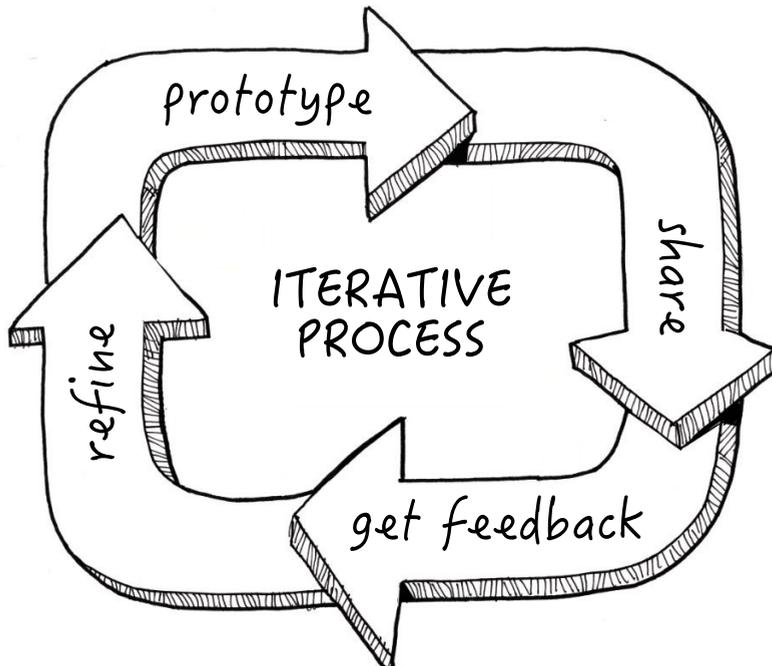


Sì; spesso è una fase rapida, ma non è questo l'aspetto più importante di questo approccio. La cosa più interessante, infatti, è che è un processo iterativo e incrementale finalizzato a produrre in modo rapido qualcosa che possa essere mandato in lavorazione al team di sviluppo. Questo approccio permette di iniziare a sviluppare il codice molto prima rispetto all'approccio classico waterfall.

Un po' come se avessimo una iterazione sulla fase di progettazione delle iterazioni. Più che, "from vision to backlog", sarebbe più corretto chiamarlo "from vision to backlog AND back", a sottolineare che non si tratta di un processo di lavoro monodirezionale a cascata. Anche in questo caso si itera ed si raffina il modo in cui si trasforma l'idea in un prodotto.



Le prime funzionalità che si realizzano sono molto utili per raffinare l'idea di quello che dovrà essere realizzato in seguito.



I feedback che si otterranno dai primi rilasci permetteranno di affinare le fasi successive di trasformazione di idee in funzionalità da implementare.



IDEA

VISION



TEST



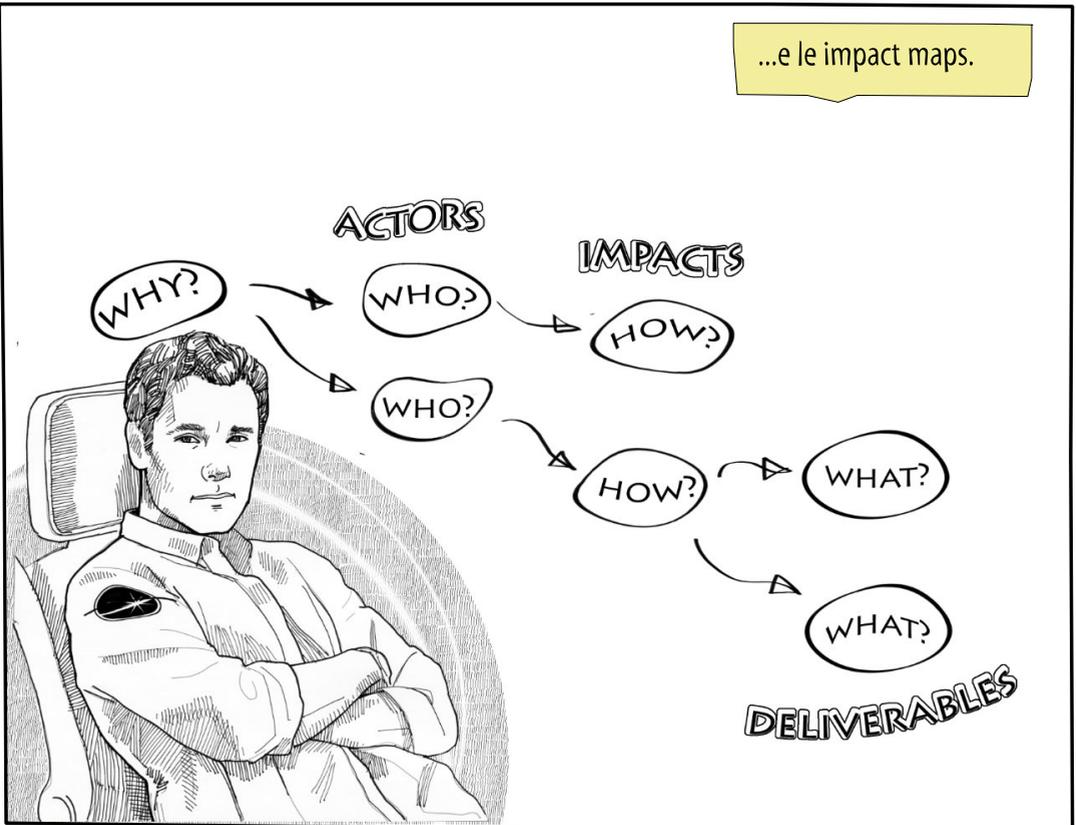
FEEDBACK

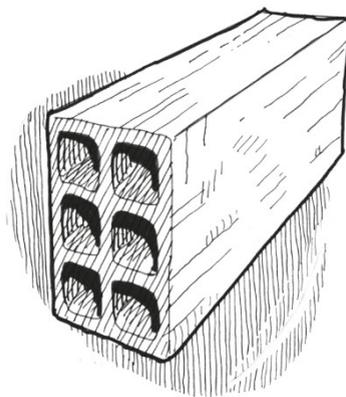
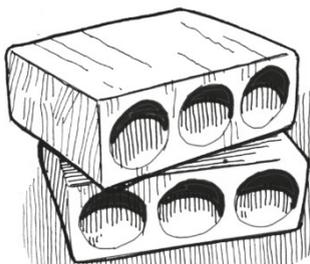
Infatti, se è pur vero che la vision di prodotto non dovrebbe cambiare più di tanto, i test con gli utenti finali permetteranno di cambiare opinione sui dettagli del prodotto.

Ma sul sistema Inception vedremo come utilizzare alcuni strumenti quali lo story mapping...



...e le impact maps.



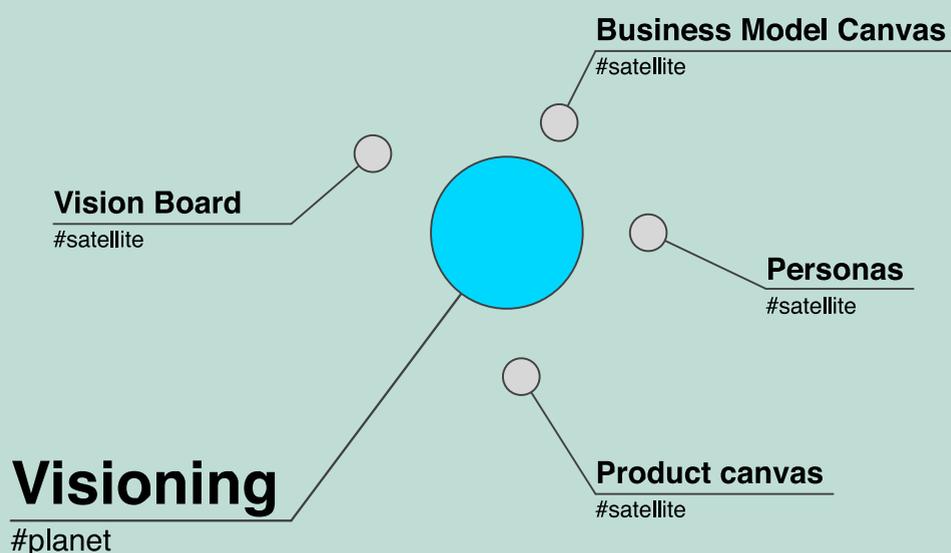


Sono tutti strumenti utili per completare l'ultimo passaggio del processo, ossia quello che mette in fila le necessità di business dell'utente finale e le trasforma negli elementi grezzi di un backlog



# Capitolo 1

## Impostare la visione di prodotto



## Introduzione

In questa parte del nostro libro sull'esplorazione della "galassia agile" parleremo di come **tradurre** un'idea in una serie di **funzionalità** da **implementare** per realizzare il prodotto. Parlando in termini comunemente utilizzati in ambito Agile, vedremo quali sono i passi per convertire una **vision** in un **product backlog** contenente elementi pronti per essere implementati dal team.

Questo argomento spesso viene trattato con titoli differenti, come **Lift Off, From Vision to Backlog** e altri ancora.

Questa fase è estremamente importante e delicata, dato che condiziona non solo **quello** che il team andrà a realizzare ma anche **come** lo realizzerà. Come responsabili del prodotto vogliamo evitare l'errore di costruire un prodotto che non soddisfi le necessità dei nostri utenti finali.

Prima di capire le dinamiche e le pratiche per concretizzare il **product backlog** è bene definire cosa si intende per **prodotto**, **servizio** e **progetto**.

## Prodotto, servizio e progetto

Un **prodotto** è un "manufatto" che soddisfa uno o più bisogni del suo utilizzatore o risolve uno o più problemi.

Un **servizio** è l'equivalente di un prodotto, ma in senso immateriale. Anch'esso soddisfa uno o più bisogni o risolve uno o più problemi del suo utilizzatore.

Un **progetto** è un'attività limitata nel tempo, che ha un inizio e una fine, il cui scopo è creare prodotti e servizi di valore per gli utilizzatori.

In questo capitolo e nei successivi utilizzeremo i termini **prodotto** e **servizio** come sinonimi, visto che si tratta di due realtà ("materiale" e "immateriale") abbastanza sovrapponibili.

Invitiamo però il lettore a **non** confonderli con il **progetto**. La distinzione è molto importante perché permette di definire bene anche i ruoli all'interno del team. Il **progetto non ha valore** per l'**utilizzatore**: noi **non** guidiamo i piani di progetto di un'automobile ma, l'automobile vera e propria (il **prodotto**) che è il risultato di quel progetto.

## Questioni agili

In un'ottica agile, questa distinzione è molto importante: l'approccio agile ha l'obiettivo di creare prodotti di valore, ossia soddisfare le necessità di business dell'utente finale, con il minor sforzo progettuale. Un prodotto non nasce dal nulla, ma si sviluppa grazie a un **processo graduale** che porta dall'identificazione di un bisogno o di un problema da risolvere, poi all'idea per la sua risoluzione, e infine alla realizzazione del prodotto o del servizio che soddisfa il bisogno o risolve il problema.

L'obiettivo di un qualsiasi sviluppatore di prodotto è riuscire a creare questi prodotti velocemente, a basso costo e generando il maggior valore possibile. È questa la sfida che ci poniamo quando affrontiamo un progetto. Vediamo di seguito i principali passi da svolgere per cercare di ottenere questo obiettivo.

## Bisogni e problemi

Quando ideiamo, progettiamo e realizziamo un prodotto, dobbiamo tenere sempre presente che il risultato del nostro lavoro andrà a soddisfare i **bisogni** degli utenti o risolverà i loro **problemi**. Si parla spesso di attenzione al cliente e di prodotto “pensato attorno al cliente”: purtroppo spesso questi buoni propositi si perdono per strada durante l'esecuzione del progetto, finendo per dare maggior rilievo ad aspetti ritenuti più importanti come le **scadenze** e il **budget**, **sottovalutando** invece di definire in modo approfondito cosa effettivamente crea **valore** per l'**utente** finale.

In Agile, volendo porre l'utente e i suoi bisogni al centro, si preferisce identificare fin da subito chi è l'**utente** a cui ci vogliamo rivolgere e quali sono i suoi **bisogni** che il prodotto andrà a soddisfare. Questi saranno i punti fermi durante tutto il progetto. Dovranno essere convalidati e verificati **costantemente** durante tutta la fase di implementazione del prodotto; qualora ci si accorgesse che il prodotto che si sta creando non risolve i bisogni identificati inizialmente, oppure che tali bisogni sono cambiati, è importante apportare le necessarie modifiche o addirittura interrompere il lavoro.

## Condividere l'idea di prodotto

Oltre alla necessità di costruire il prodotto in risposta ai bisogni dell'utente, un altro importante aspetto da tenere in considerazione durante la fase di progettazione e realizzazione è creare una **visione condivisa** fra i vari *stakeholder* di progetto: il business, il team di sviluppo, il cliente o committente, gli esperti di dominio e, non ultimo, l'utente finale, quando è possibile coinvolgerlo direttamente. In tal senso è molto importante imparare a **condividere** l'idea di prodotto che abbiamo, renderla comprensibile a tutti i partecipanti al progetto in modo che lavorino tutti nella stessa direzione.

In questo capitolo parleremo di una tecnica che sfrutta alcuni **Canvas** e che è stata proposta la prima volta da Roman Pichler [1]. Si tratta di un'ottima fonte di ispirazione che poi abbiamo usato effettivamente nei progetti per descrivere la **vision** e il **backlog** dei diversi prodotti.

Sono passati ormai alcuni anni da quando abbiamo cominciato a usarla, sia nei corsi che durante le prime fasi di analisi con i clienti, e il riscontro sul campo ha dimostrato che si tratta di uno strumento sempre molto utile ed efficace.

## Un grosso rischio: non condividere l'idea

Di solito, alla partenza del progetto, qualcuno — il **Project Manager**, il **Product Manager** o un'altra figura di rilievo — prepara una **presentazione** che descrive la **vision** del prodotto che si realizzerà e la presenta alle persone interessate per condividerne i punti e raccogliere i pareri.

Tipicamente, dopo questa presentazione, si procede a raccogliere i vari commenti e feedback per fare un **riassunto** delle **opinioni** e dei **consigli**. Tale report viene poi salvato da qualche parte e inviato ai partecipanti; di rado avviene un secondo meeting dove si prova a **rivedere** la **vision** sulla base dei commenti raccolti.

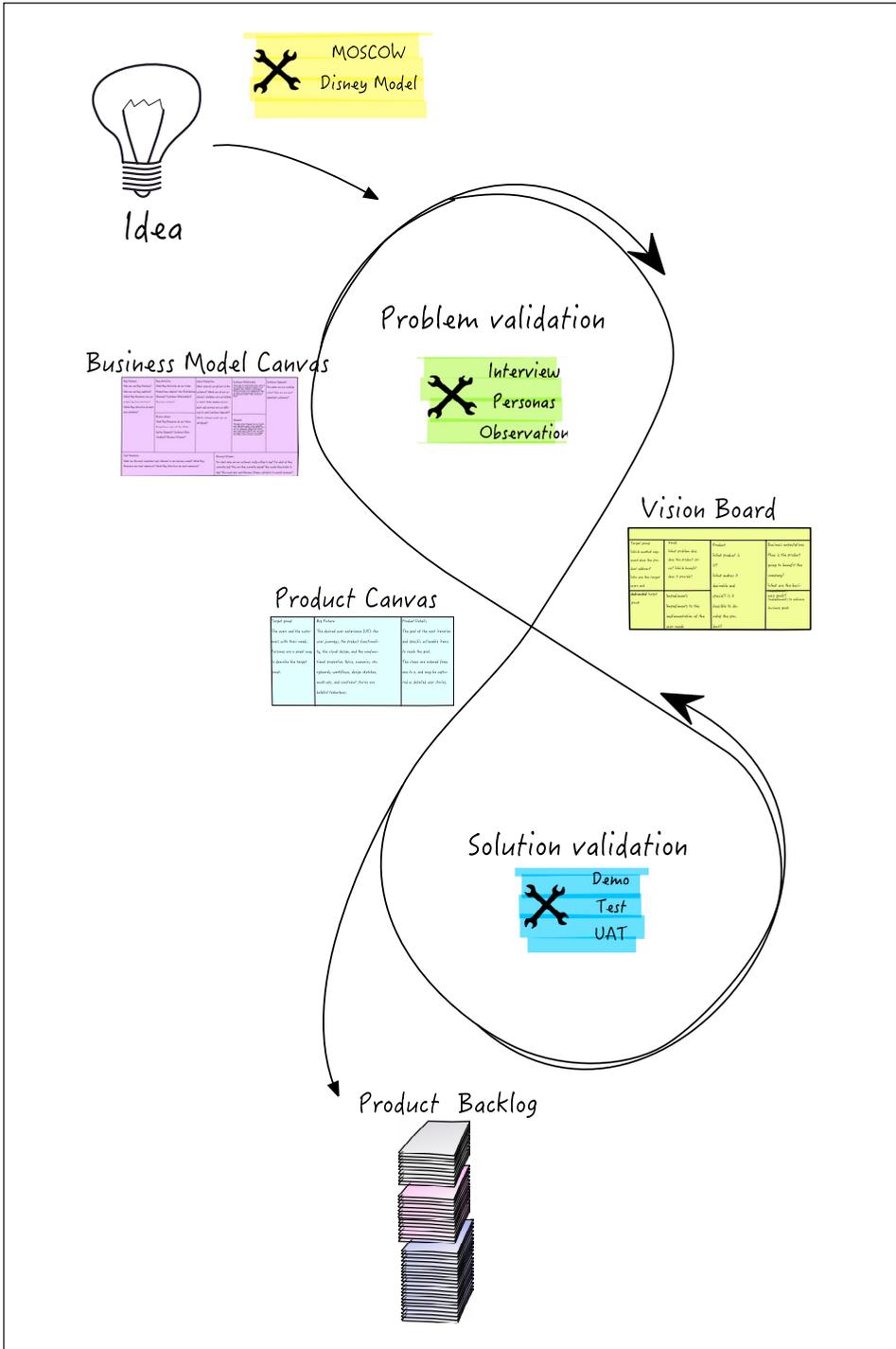


Figura II.1. Le fasi di envisioning di prodotto: dall'idea al product backlog.

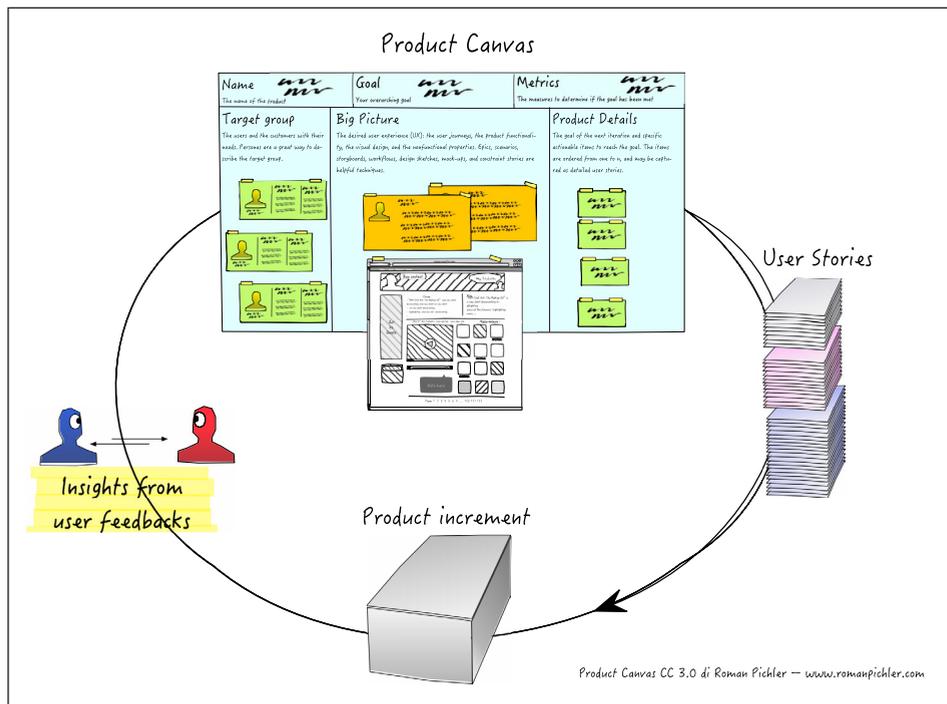


Figura 11.2. Ciclo di miglioramento continuo del Product Backlog con il Product Canvas.

Questo è un metodo piuttosto classico di lavorare, diffuso in aziende e gruppi di vario tipo. Il limite principale di tale approccio risiede nello **scarso** livello di **allineamento** e condivisione. Effettuare un solo raffinamento spesso dà luogo a molti fraintendimenti o a passaggi non chiari: durante la fase di realizzazione del progetto questo è fonte di cambio dei requisiti, che risultano in conflitto con l'idea iniziale e non raramente anche fra di loro.

Arrivano i ritardi, il progetto che non risponde alle esigenze di business, il team non capisce la direzione impostata dal business e il business non capisce come mai il team ci metta così tanto tempo a fare le cose: in fondo si chiedono loro solo delle piccole modifiche.

Una delle cause principali risiede proprio nel non avere una **visione condivisa** e **compresa** veramente **da tutti**, perché non è stata fatta propria da coloro che lavoreranno al progetto. I **documenti non** sono stati **letti** con attenzione e le **riunioni** sono state seguite **distrattamente**.

### Più coinvolgimento, più condivisione

Un buon modo per risolvere questi problemi, oltre a migliorare l'aspetto di comunicazione e condivisione, è quello di **coinvolgere** gli stakeholder non solo alla presentazione ma ben prima, durante la fase di definizione della **visione** di prodotto.

Utile in questo caso è quindi trovare un format di lavoro **collaborativo** semplice e rapido che mette tutti gli attori nella stessa stanza per un periodo breve ma molto efficace. Nelle prossime pagine vedremo il nostro approccio per risolvere questo problema tramite un workshop in cui i vari stakeholder lavorano fianco a fianco per un periodo breve ma molto intenso.

## Costruire una vision condivisa: i vari passi del workshop

Nel workshop che abbiamo ideato, partendo da un'idea, una frase, un "titolo", si procede nel descrivere la vision **completa** e **coerente**, condividendola da subito con tutte le persone coinvolte nel progetto.

Per questo scopo si possono usare vari strumenti, come l'Elevator Pitch oppure cose più fantasiose come la **product box** o la classica ma sempre efficace **Vision Board**, strumento con il quale si sintetizzano gli **aspetti principali** del prodotto.

Il passo successivo prevede di elencare i **fattori portanti** del prodotto dal punto di vista del **business**. In questa fase si può utilizzare il **Lean Canvas** o in alternativa il **Business Model Canvas**.

Infine tramite l'utilizzo dello strumento delle **Personas**, si procede a individuare gli attori/utenti del sistema elencandone i relativi bisogni di business. Questo elenco porterà, per esempio tramite una sessione di **User Story Mapping**, alla creazione delle funzionalità da implementare o meglio dei bisogni da risolvere: si creerà quindi il **Product Backlog**.

Le storie del **Product Backlog** verranno poi **raffinate** grazie a iterazioni successive e alla verifica del prodotto incrementale confrontando i "viaggi" degli **utenti reali** con quelli delle **personas** pensate durante il workshop.

La cosa importante della **vision** basata su questi tre canvas è che sia sempre coerente e allineata con il mondo reale che sta usando il nostro prodotto.

## Riferimenti

Roman Pichler, *The Product Vision Board*, maggio 2011

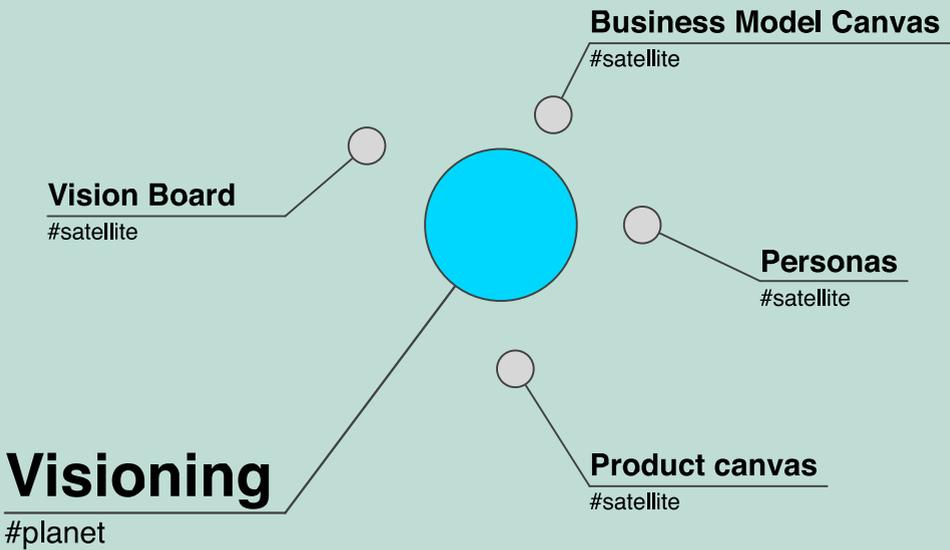
<http://www.romanpichler.com/blog/agile-product-innovation/the-product-vision-board>





# Capitolo 2

# La Vision Board



## Introduzione

Cominciamo quindi a fare i primi passi utilizzando uno degli strumenti a nostra disposizione: vedremo in questo capitolo in cosa consiste una **Vision Board** e quanto il suo corretto utilizzo possa aiutare a mettere a fuoco la visione di prodotto. Si tratta di uno strumento semplice ma decisamente efficace.

## Una lavagna per favorire la visione

L'obiettivo della **Vision Board** è di catturare i **quattro aspetti** fondamentali di un prodotto: gruppo target di **utenti**, **bisogni** di business, **proprietà** del prodotto e proposta del prodotto in termini di **valore** per l'azienda.

## Target Group

Indica a chi è rivolto il prodotto tramite l'elencazione delle categorie di utilizzatori finali del prodotto che state pensando. È utile non essere troppo dettagliati ma identificare le macro-categorie. Man mano che la visione sarà completata, si potranno far emergere nuove categorie oppure aggregarne di esistenti.

## User needs

È l'elenco dei bisogni di business, ad alto livello, dei macro utenti o quali problemi va a risolvere il prodotto. Durante la compilazione si deve cercare di identificare almeno

Vision Statement Description of the product in a sentence			
<b>Target group</b>  A quali segmenti di mercato si rivolge il prodotto? Qual è l'utente destinatario?	<b>Needs</b>  Quali bisogni gli utenti desiderano che siano soddisfatti? Quali benefici procura?	<b>Product</b>  Cosa è questo prodotto? Cosa lo rende desiderabile e speciale? È vantaggioso il suo sviluppo?	<b>Business expectations</b>  Quali benefici procura il prodotto all'azienda? Quali sono gli obiettivi di business?
<b>Not in the target group</b>  Chi non appartiene al target group?	<b>Impediments</b>  Impedimenti a soddisfare i bisogni elencati sopra.		<b>Impediments</b>  Impedimenti al raggiungimento degli obiettivi di business

Based on Roman Pichler's Vision Board - CC 3.0 - [www.romanpichler.com](http://www.romanpichler.com)

Figura 11.3. Vision Board, con le colonne per “gruppi bersaglio”, “bisogni”, “caratteristiche di prodotto” e “aspettative di business”.

un bisogno di business per ogni gruppo bersaglio. Nel caso il bisogno sia sentito da più target group e un gruppo bersaglio non abbia un bisogno unico, si può prendere in considerazione di aggregare i target group. È importante essere sintetici nella Vision Board.

### Product properties

Questa sezione indica quali sono le caratteristiche principali e di maggior rilievo del prodotto; è una applicazione web? Mobile? Prevede la sincronizzazione in tempo reale con il cloud?

In questa parte è bene limitarsi a descriverne pochi, 3, 4, massimo 5 è una buona metrica. In questa lista ristretta, non si devono elencare quelle caratteristiche scontate ma si deve tentare di descrivere quelle di maggior rilievo.

### Value Proposition

In questa parte sono riportate le aspettative di business che l'azienda pone nel prodotto che sta realizzando. Le aspettative possono essere sia di incremento di ricavi sia di altra natura come risparmio di costi di esercizio, mantenimento della quota di mercato, miglioramento della soddisfazione dei propri clienti.

### Vision Statement

Le informazioni contenute in queste quattro colonne sono poi riassunte nel **Vision Statement**, una **frase** che tutte le persone coinvolte nel progetto dovrebbero conoscere e fare propria. In figura 3, abbiamo riportato un esempio di Vision Board.

### Uso della Vision Board

Metaforicamente, la vision è il **faro guida** del progetto: se questo faro si spegnesse o venisse perso di vista, il progetto stesso potrebbe subire dei danni o dei cambi di rotta che potrebbero portare al suo fallimento.

La **sintesi** della vision aiuta a mantenere i concetti semplici da condividere, da memorizzare e da verificare costantemente. L'ideale sarebbe avere la **Vision Board** sempre a portata di sguardo, sia per il team che per il business, insieme agli altri **information radiators** (“diffusori di informazione”).

### Un esempio di Vision Board

A scopo di esempio, immaginiamo come si costruirebbe la Vision Board di un ipotetico servizio di home banking.

Per la compilazione della board, in genere si procede inserendo le informazione relative nelle quattro colonne della **Vision Board** da sinistra verso destra; al termine si prova a sintetizzare il tutto in una frase rappresentativa dei quattro fattori che compongono la vision.

Nel caso dell'esempio, il **Vision Statement** potrebbe essere: “Per chi vuole gestire il proprio conto corrente velocemente, pagando solo per i servizi attivati, **MyHomeBank**

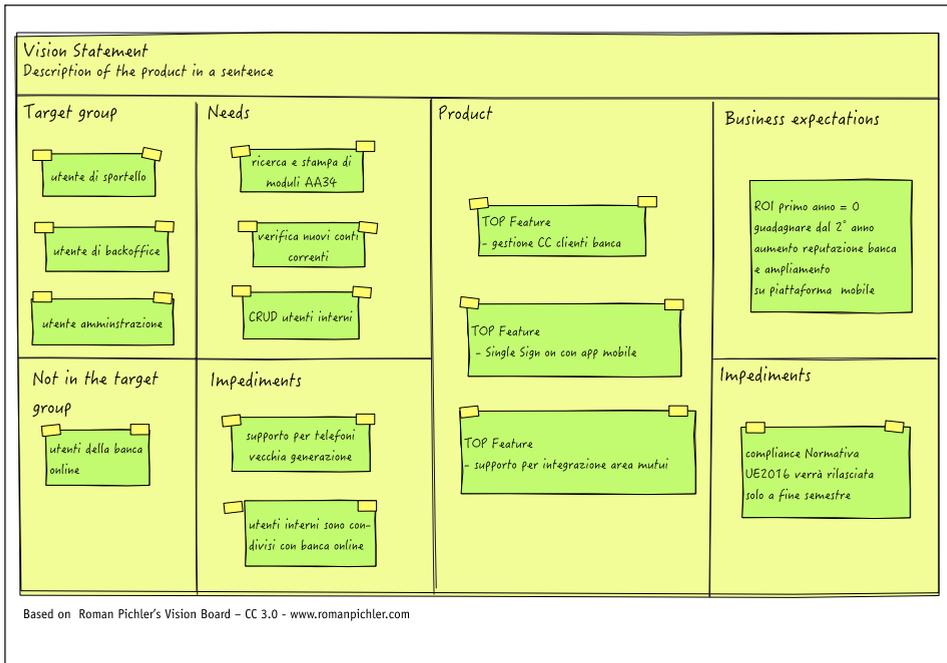


Figura II.4. Esempio di Vision Board per un ipotetico servizio di home banking.

è l'app mobile che ci permette di aumentare il numero di utenti". Si noti la parte "... che **ci** permette di aumentare...": in questo caso il plurale (il "**ci**") sta a significare "**noi, azienda** che commercializza questo servizio".

È questo un aspetto molto importante da sottolineare: troppo spesso, infatti, viene trascurato il **motivo aziendale** per il quale il prodotto è realizzato. È bene invece esplicitarlo ed evidenziarlo bene. Il motivo aziendale sarà infatti molto importante quando si dovrà procedere all'ordinamento degli elementi del **Product Backlog**.

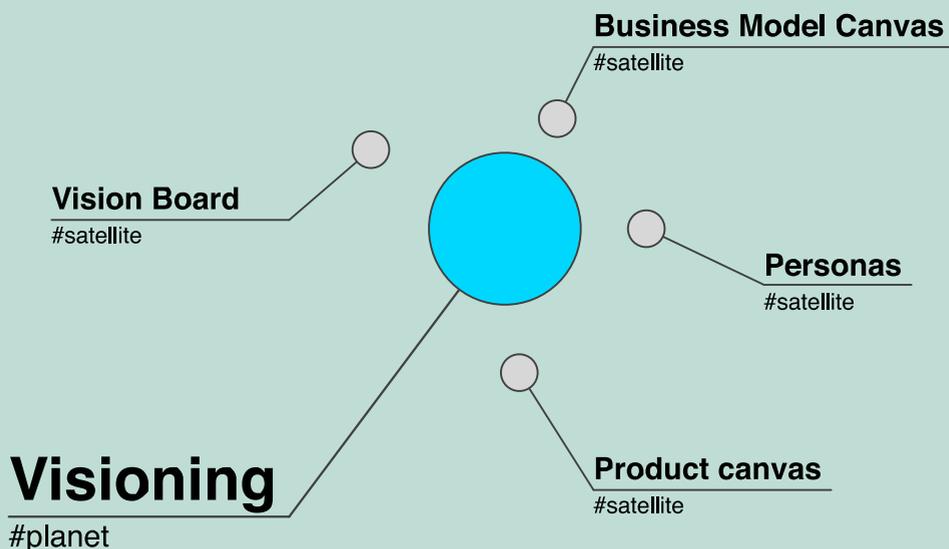
Una volta che la Vision Board sia stata completata e sintetizzata in una frase esplicativa, è utile lasciarla "sedimentare" un po' di tempo, per poi rivederla tutti insieme in modo da **verificarla** e consolidarla. Una volta rivista e migliorata, la **Vision Board** sarà pronta e a disposizione di tutti, e si potrà passare alle fasi successive quando si useranno altri due canvas molto importanti: il **Business Model Canvas** e poi il **Product Canvas**.





# Capitolo 3

## Validare il modello di business della vision



## Lean Canvas

Per quanto buona e ben descritta, una vision da sola non basta per creare un prodotto: affinché sia realizzabile, occorre legarla a un **modello** di **business**.

Una possibile risposta è lo strumento **Lean Canvas** [1] ideato da Ash Maurya in *Running lean* [2] e sviluppato a partire dal **Business Model Canvas** [3] di Alexander Osterwalder.

In breve, il **Lean Canvas** è uno strumento utilizzato principalmente per creare velocemente un modello di business. A differenza di un business plan, tipicamente composto da oltre 30 pagine e che necessita di settimane o mesi per essere redatto, il **Lean Canvas** è composto da un'unica pagina realizzabile anche solo in 20 minuti.

È quindi uno strumento **veloce** e conciso che permette, da un lato, di produrre e validare in poco tempo molteplici modelli di business e, dall'altro, **obbliga a focalizzarsi** esclusivamente sui punti salienti e più importanti del prodotto.

## Come è fatto un Lean Canvas

Il **Lean Canvas** è composto da **9 sezioni** che possono essere compilate in ordine sparso, man mano che si hanno le informazioni, anche se tipicamente si segue l'ordine seguente:

- customer segmentation
- problem

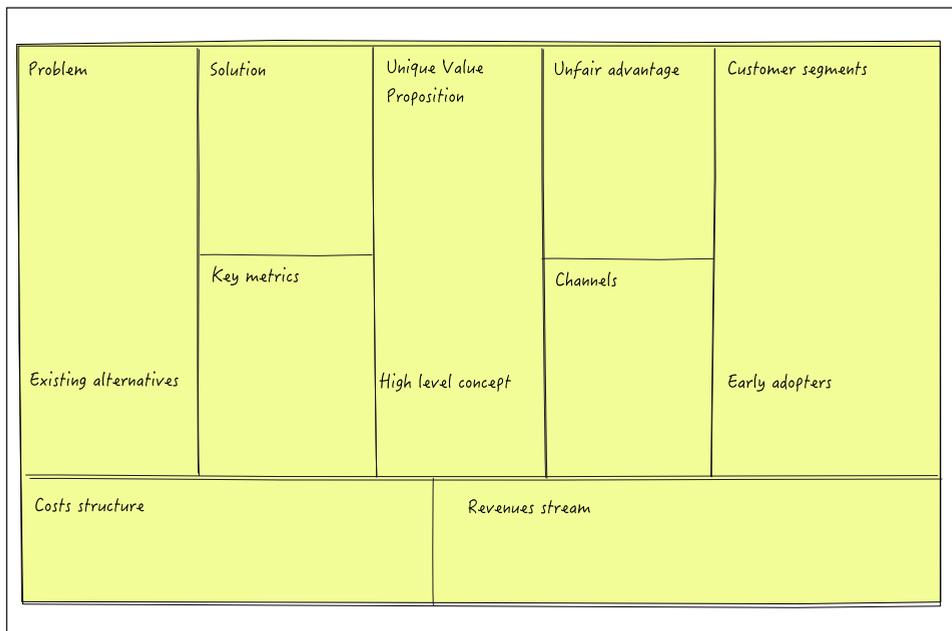


Figura 11.5. Lean canvas, con le diverse sezioni che lo compongono.

- unique value proposition
- solution
- channel
- cost structure
- revenue stream
- key metrics
- unfair advantage

### Customer segmentation

È difficile costruire un prodotto/servizio se non conosciamo **chi** sono i nostri clienti e i nostri utenti e come questi sono “segmentati”: stile di vita, età, amatori o professionisti del settore, livello di coinvolgimento tecnologico e così via. In questa sezione andiamo principalmente a sciogliere questi nodi.

È bene porre attenzione alla differenza tra **utenti** e **clienti**: mentre i primi sono quelli che **utilizzano** il nostro prodotto/servizio, i secondi sono quelli che lo **pagano**. In alcuni casi utenti e clienti coincidono, in altri no.

Si possono eventualmente valutare anche le caratteristiche dell’utente “ideale” che userà il prodotto o il servizio fin da subito, nello spazio **early adopters**.

### Problem

Citando un detto di Charles Kettering, “un problema ben identificato è un problema mezzo risolto”. In questa sezione si vanno a individuare i principali **3** problemi che deve risolvere chiunque dovrà realizzare il prodotto.

Se esistono, è consigliato anche elencare le possibili soluzioni del problema che ad oggi già esistono, anche se si tratta di soluzioni parziali. A tale scopo è presente lo spazio per le **existing alternatives**.

### Unique value proposition

La UVP è una **frase** che sostanzialmente spiega il motivo di esistere del prodotto/servizio che si sta realizzando. In che cosa si differenzia dagli altri? Perché i clienti dovrebbero comprarlo?

Questa forse è la sezione più importante del modello in quanto catalizza l’attenzione del lettore e illustra le **motivazioni** alla **base** della **vision**.

È possibile inoltre specificare un concetto di astrazione ad alto livello (spazio **high level concept**) che serva a spiegare ulteriormente la **unique value proposition**: l’esempio tipico è quello che ci spiega come “Netflix = Spotify dei film”.

### Solution

Specularmente alla sezione **problem**, qui si elencano le principali **3** **caratteristiche** del prodotto che risolvono i problemi sopra esposti e che dimostrano la **unique value proposition**.

## Channel

In questa sezione si vanno a elencare i **canali** per mezzo dei quali si può “consegnare il valore del prodotto” ai clienti. È molto importante capire **quanti** e **quali** sono questi canali, in quanto la loro mancata identificazione è una delle principali cause di fallimento.

## Costs structure

La sezione relativa alla **costs structure** serve a identificare quali saranno i **costi** da sostenere per realizzare il modello di business descritto. La **struttura dei costi** deve tenere in considerazione svariati aspetti:

- identificare la **frequenza** dei costi;
- stimare l'**ammontare** dei costi;
- individuare i costi **fissi**, ossia quelli che rimangono invariabili indipendentemente dalla quantità prodotta;
- individuare i costi **variabili**, ossia quelli che sono funzione della quantità prodotta e che variano al crescere di quest'ultima;
- individuare possibili **economic**: di **scala** in cui il costo si riduce all'aumento della quantità prodotta, di **esperienza**, in cui il costo diminuisce al crescere dell'esperienza accumulata nel realizzare il prodotto, di **scopo**, in cui il costo diminuisce grazie a produzioni congiunte e al riutilizzo di conoscenze, e così via.

## Revenues stream

In contrapposizione alla struttura dei costi, qui si va invece a identificare il **flusso dei ricavi** generato dalla realizzazione del modello di business. Più in particolare in questa sezione si andrà a descrivere la struttura dei prezzi.

Punto focale resta la definizione del **prezzo** e, per farlo, potrebbe essere utile considerare i seguenti elementi:

- la **strategia** di prezzi da utilizzare, ad esempio **freemium** (una parte del prodotto/servizio gratuita e un'altra a pagamento), **ads** (ricavi generati da pubblicità), **subscription** (ricavi generati da quote di iscrizioni/abbonamenti), **licensing** (ricavi derivanti da vendita di licenze), e così via;
- le **modalità** di pagamento (a **rate**, **una tantum**) e così via;
- il **posizionamento** del prodotto/servizio nel mercato.

## Key metrics

In questa sezione vanno elencate tutte le attività che si andranno a **misurare per capire** l'andamento del business. Esempi di queste **metriche** sono: **acquisizioni**, **attivazioni**, **retention**, **referenze**, **revenue**, e così via.

## Unfair advantage

Jason Cohen descrive un **unfair advantage** come “qualcosa che non può essere facilmente copiato o comprato”. Molte volte, individuare gli **unfair advantage** può essere

difficilissimo e, almeno nella fase iniziale non è sbagliato lasciare questa sezione del canvas in bianco, come suggerisce Maurya.

Per ora i lettori non si preoccupino eccessivamente di questo aspetto, tanto ci torneremo adeguatamente sopra nei prossimi capitoli, in cui vedremo l'uso del Lean Canvas in maniera più dettagliata.

## Conclusioni

Il Lean Canvas è certamente uno strumento che aiuta nell'affrontare un argomento complesso e importante, come quello dell'individuazione degli aspetti di business del prodotto che si deve realizzare; questo canvas si rivela molto utile per la sua capacità di focalizzare l'attenzione sulla definizione di alcune caratteristiche del prodotto nelle fasi iniziali.

Su di esso, e sul suo uso pratico, torneremo nei prossimi capitoli. La cosa fondamentale da capire è che un prodotto o un servizio devono nascere come risposta a una serie di bisogni o di problemi — siano essi preesistenti o indotti — che un utente può avere.

Una volta identificate le possibili soluzioni e il modo in cui sostenerle, arriva il momento in cui dobbiamo condividere e comunicare la vision del prodotto. Nel prossimo capitolo vedremo quali strumenti abbiamo a disposizione per farlo.

## Riferimenti

Ash Maurya, *Why Lean Canvas vs Business Model Canvas?*

<https://goo.gl/iHqTLV>

Ash Maurya, *Running Lean: Iterate from Plan A to a Plan That Works*. O'Reilly, 2<sup>nd</sup> edition, 2012

Business Model Canvas

<http://www.businessmodelcanvas.it>



# Capitolo 4

## Raccogliere le storie con la User Story Map

**Backlog**

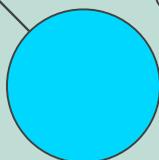
#planet

**US Workshop**

#satellite

**Story mapping**

#satellite



## Introduzione

Una celebre frase di Kent Beck recita: “Le roadmap dei prodotti dovrebbero essere liste di domande e non elenchi di funzionalità”. In quest’ottica, nel presente capitolo intendiamo parlare delle tecniche per arrivare a definire l’**elenco delle cose da fare**.

In **Scrum**, questo elenco di cose da fare è detto **Product Backlog Items**. Per arrivare a una prima versione del **PBI**, è necessario individuare la lista di bisogni e di domande a cui deve rispondere il sistema e trovare anche un **ordine** con cui metterle in fila. Fra le varie tecniche che si possono utilizzare, la **User Story Mapping** è una delle più efficaci e, proprio per questo motivo, la sua diffusione è sempre più ampia.

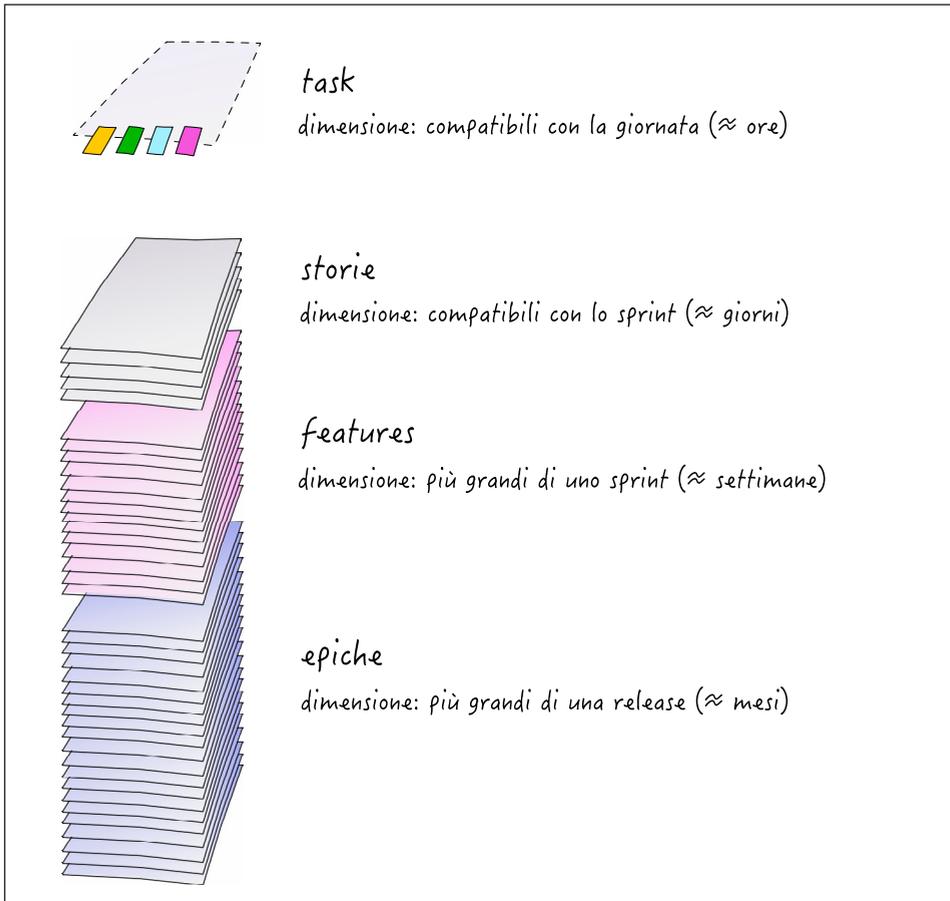


Figura II.6. Il Product Backlog di un progetto Scrum; contiene gli elementi che devono essere implementati per la realizzazione del prodotto. Sono ordinati per priorità (o valore) e sono disposti in modo che gli elementi in alto siano piccoli a sufficienza per poter essere messi in lavorazione, mentre in basso si trovano gli elementi più grandi. Normalmente il formato utilizzato è quello delle storie utente che diventano epic, feature, user story.

## Come è fatto un Product Backlog

Il **Product Backlog** è una **pila** che contiene l'**elenco** dei **bisogni** dell'utente che utilizzerà il prodotto ideato e da realizzare. Nella maggior parte dei casi, gli elementi sono descritti con il **formalismo** delle **storie utente**, ma si possono usare anche altri formati.

Nella parte alta di questa pila ci sono gli elementi ad **alta priorità**, quelli che si ritiene sia più utile realizzare per primi perché la loro implementazione fornisce all'utente un valore maggiore; nella parte bassa, invece, ci sono le funzionalità che sono spesso considerate “nice to have” ossia aggiunte utili al prodotto, ma **non indispensabili** per l'utilizzo.

## Contenuto del backlog e ordine delle storie

Arrivare a definire il **contenuto** e l'**ordine** del backlog è probabilmente una delle attività più importanti di un progetto agile basato su Scrum o su altra metodologia. Lo Story Mapping aiuta esattamente in questo compito: tramite un lavoro di gruppo, una sorta di discussione collaborativa, si arriva in genere non solo alla definizione del contenuto del **Product Backlog**, ma anche alla realizzazione di una **mappa** delle **cose da fare**.

Tale mappa, di fatto una sorta di **todo list** bidimensionale, rappresenta spesso un valido strumento **complementare** o **in sostituzione** del backlog, proprio grazie all'aggiunta di una seconda dimensione, garantendo una visione d'insieme spesso migliore di quella offerta dalla lista **monodimensionale del backlog**.

Avere a disposizione una visione bidimensionale del lavoro da svolgere agevola inoltre le attività di valutazione, prioritizzazione e raffinamento delle storie. Grazie alla sua immediata capacità di fornire una rappresentazione **visuale** delle cose da fare, la **Story Map** definisce quindi una sorta di linguaggio universale facilmente comprensibile anche da persone non facenti parte del team di sviluppo.

Il processo stesso di creazione e di gestione della mappa agevola la **discussione collaborativa** del gruppo; la tecnica di lavoro, iterativa e incrementale, si basa su un procedimento semplice e comprensibile anche da personale non tecnico o che non sia stato introdotto alle pratiche agili: per questo spesso viene utilizzata come strumento alternativo nelle sessioni di **project planning tradizionale**, dove sia necessario chiarire gli aspetti legati alle scadenze, alle tempistiche e allo scope delle varie release di progetto/prodotto.

## Come nascono le Story Maps: le considerazioni di Jeff Patton

Anche se le **story maps** sono utilizzate da molto tempo in varie forme, il primo che probabilmente le ha formalizzate e rese famose è stato Jeff Patton, che l'ha descritta nel suo libro *User Story Mapping* [1]. Fra le molte caratteristiche che secondo Patton rendono importanti le mappe, è la loro capacità di offrire una **rappresentazione persistente** del lavoro di **visioning** nonchè del contesto di lavoro.

Nel suo libro infatti Patton racconta:

«Trascurriamo molto tempo a lavorare con i nostri clienti e ci sforziamo di comprendere i loro obiettivi, i loro utenti e le parti principali del sistema che potremmo costruire. Infine arriviamo ai dettagli: le diverse parti delle funzionalità che ci piacerebbe costruire. Io immagino questo processo come se fosse un albero: il tronco è costituito dagli obiettivi o dai vantaggi desiderati per cui si decide di costruire il sistema; i rami grandi principali rappresentano gli utenti; i rami secondari più piccoli sono le capacità di cui essi hanno bisogno; infine, le foglie sono le storie utente, sufficientemente piccole da poter essere impiegate nelle iterazioni di sviluppo.

Dopo tutto questo lavoro, in cui si è creata tutta questa conoscenza condivisa, a me sembra che spesso si comincino a staccare le foglie dell'albero e a buttarle nei sacchi per lo smaltimento, e poi si proceda ad abbattere l'albero. Ecco come mi appare un backlog piatto: un sacco di paccime senza un contesto [...]

Le storie davvero grosse possono essere considerate come delle 'epiche': Mike Cohn le descrive per l'appunto in questo modo. Sono storie, solo che si tratta di storie veramente grandi: troppo grandi per essere valutate e sviluppate. Quando un'epica entra nel nostro backlog e arriva il momento di discuterla nel dettaglio, vedo spesso le persone che la rimuovono dal backlog stesso, la scompongono e sostituiscono i pezzi che riescono a identificare. Questa è la classica situazione in cui non posso trattenere una smorfia quasi di dolore: è esattamente il caso dell'abbattimento dell'albero del quale si mantengono le foglie dentro un sacco per il paccime. Infatti, quella storia grande costituiva comunque il contesto. Era il mio modo semplice di pensare a riguardo dell'intera attività che le persone stavano svolgendo; era il mio modo veloce di spiegare agli altri come vedo io il sistema».

A riprova che la **User Story Mapping** non è una tecnica inventata da Jeff Patton, c'è la stessa testimonianza dell'autore che racconta come spesso trovasse presso i clienti tecniche analoghe o del tutto simili. In questo senso Patton ci dice quindi che la **Story Mapping** non è una tecnica, ma piuttosto un **pattern** di lavoro:

“Se si spiega un concetto e si riceve come risposta ‘Che bella idea!’, non siamo di fronte a un pattern. Ma se le persone ci dicono ‘anche noi usiamo qualcosa di simile a questo strumento’, allora siamo in presenza di un pattern”.

### Story Map Workshop, il processo per arrivare alla mappa delle storie

In questo capitolo vediamo come si può arrivare alla definizione della mappa delle storie tramite un processo detto di **Story Map Workshop**: partendo dal lavoro descritto dallo stesso Jeff Patton nel suo libro, aggiungeremo alcune tecniche o variazioni che abbiamo sperimentato e trovato utili nel corso delle attività di coaching in progetti agili.

In particolare l'applicazione della tecnica del **buy an issue** per la valutazione del valore delle cose da fare è stata da noi introdotta — e siete quindi liberi di considerarla una stupidaggine — prendendo spunto da tecniche di **Value Stream Mapping** e **serious gaming**; si veda a tal proposito [2], al paragrafo intitolato “A tasty cupcase”.

In linea con la filosofia agile, la realizzazione della mappa viene fatta dalle stesse persone che poi effettueranno la lavorazione del progetto: essenziale la presenza del **Product Owner** — o figura analoga se il progetto non sarà organizzato secondo Scrum — e di qualche membro del **team di sviluppo** oltre agli altri **stakeholders** interessati allo sviluppo del prodotto. In questa prima fase non è essenziale che partecipi il team di sviluppo al gran completo: essere in pochi può risultare un buon modo per mantenere un clima intimo e riflessivo. Molto utile, in alcuni casi necessaria, la presenza di un **facilitatore**, un coach che guidi la riunione, scandisca i tempi e definisca le attività.

### Elenco degli utenti del sistema

Per identificare un primo **elenco** con le **macrofunzionalità**, si cambia la prospettiva, guardando il sistema dal **punto di vista** degli **utenti** e immaginando quindi quali saranno le **attività** che essi possono o desiderano svolgere all'interno del prodotto.

Per prima cosa quindi il gruppo di lavoro procede nell'individuare l'elenco degli **utenti** del sistema: questo lavoro può essere fatto in modo collaborativo (tutti di fronte alla lavagna) oppure isolatamente, tramite la tecnica del **silent brainstorming**, tecnica di collaborazione basata sull'uso di cartoncini attaccati alla parete. La tecnica del silent brainstorming è descritta in modo piuttosto chiaro da Steve Rogalsky [8]: in sintesi, potremmo dire che si tratta di una modalità di lavoro in cui ogni membro del team, lavorando **individualmente** e **in silenzio**, per prima cosa stila un proprio elenco

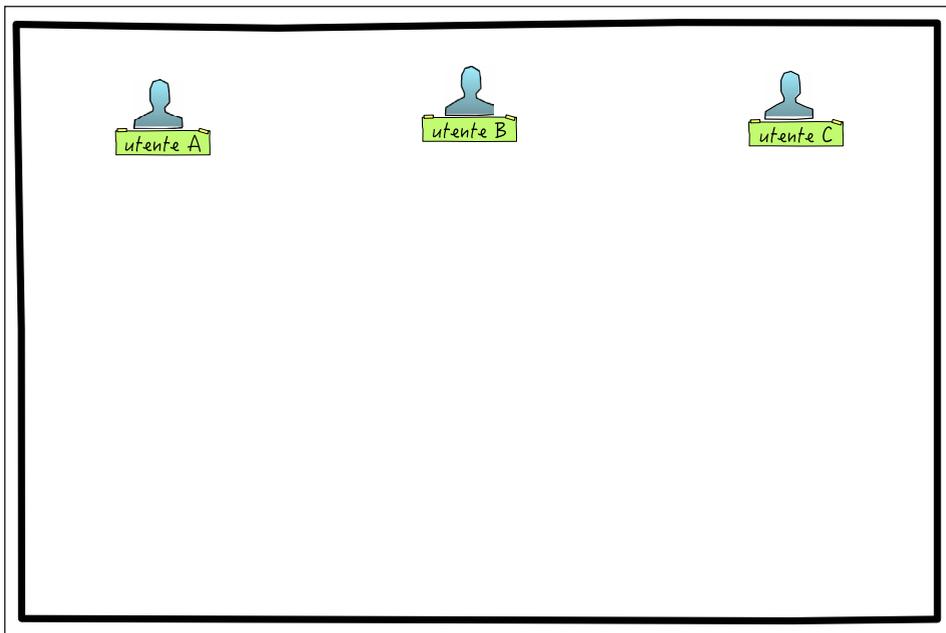


Figura 11.7. Prima fase del lavoro: si stila l'elenco delle personas.

di risposte o proposte. Successivamente **ogni elenco** personale verrà **presentato** agli altri, **discusso e confrontato**.

Non sempre è possibile nè semplice procedere in questo modo. A volte è preferibile partire definendo l'elenco delle classi di utenti dell'applicazione: una tecnica molto utile in questo caso è certamente quella della definizione delle **Personas**, di cui si è avuto modo di parlare sia in alcuni articoli pubblicati su MokaByte [3] dedicati allo sviluppo dell'esperienza utente, che in un articolo [4] che è stato opportunamente rielaborato per essere incluso in questa parte del libro.

L'obiettivo di questa fase è quello di arrivare con un elenco di nomi di ruoli o personas scritti ognuno su un cartellino e attaccati nella parte alta della parete o lavagna, come riportato nella figura 7.

## Le attività svolte dagli attori

Dopo aver individuato gli **utenti** del sistema, si procede a individuare i bisogni di business di ogni utente, in modo da individuare l'elenco delle operazioni che gli utenti svolgeranno utilizzando il prodotto, ovvero delle funzionalità che il prodotto dovrà implementare.

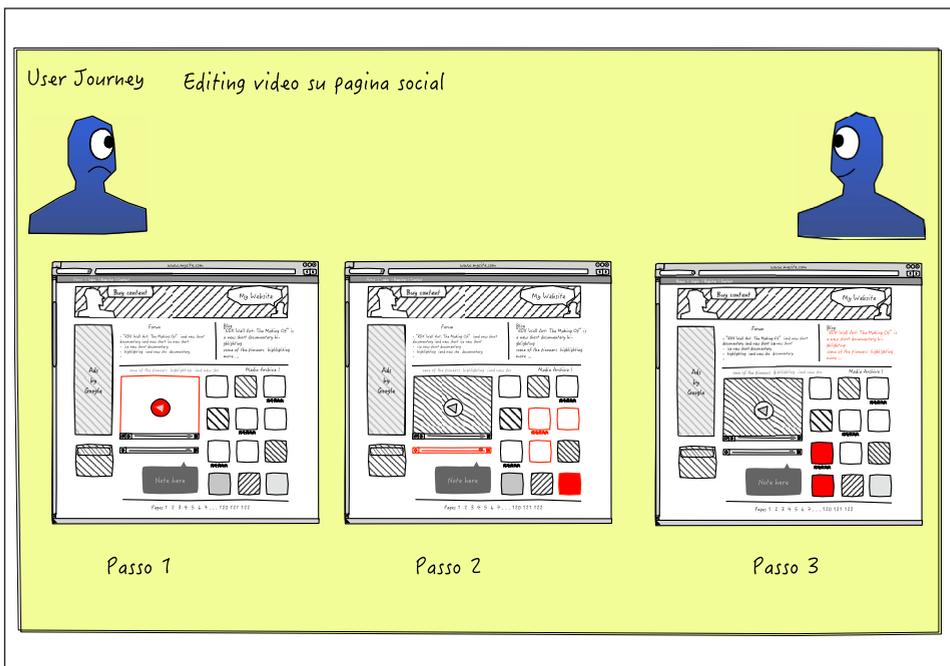


Figura 11.8. Uno user journey: il viaggio inizia fuori dal prodotto con l'utente che ha un bisogno irrisolto. L'utente "entra" nel prodotto effettuando alcune operazioni. L'utente esce dal prodotto con il suo bisogno soddisfatto.

Anche in questo caso si può lavorare in modalità **silent brainstorming**, in cui ognuno scriva su dei biglietti un elenco di funzionalità: ogni bigliettino dovrebbe esprimere un bisogno di business preciso, quindi è utile identificare il **verbo correlato** e concludersi eventualmente con un **complemento oggetto**; per esempio: “**creare nuovo utente**”, “**inviare mail**”, “**stampare report**” e così via.

Dopo che ogni partecipante al workshop ha stilato la propria lista di bisogni, ogni persona presenta i propri cartellini uno per uno attaccandoli alla lavagna sotto l’utente corrispondente.

In alternativa, se il gruppo di lavoro è numeroso, si possono creare tanti piccoli gruppi, ognuno dei quali si concentra su un utente specifico, identificandone i bisogni di business.

Un formalismo molto utile per descrivere meglio il bisogno utente e come questo possa essere soddisfatto dal prodotto, è quello di rappresentare l’**esperienza utente** nel sistema durante l’interazione dell’utente con il sistema. Si prova a disegnare un miniflusso basato su questo schema:

- utente è triste perché ha un bisogno irrisolto
- utente inizia ad utilizzare l’applicazione seguendo una serie di passi
- utente ha soddisfatto il suo bisogno ed è contento

Facciamo un esempio: Marco è un consulente che svolge attività presso aziende del settore telecomunicazioni. Marco per promuovere la sua attività spesso pubblica sui social le sue attività lavorative, le conferenze a cui partecipa, i corsi che eroga.

Ogni volta che parla a una conferenza, Marco alimenta la sua rete di contatti, pubblicando su LinkedIn, Twitter e Facebook un post con il titolo del suo talk, magari una foto e il link al sito della conferenza.

Il profilo di Marco e il suo bisogno quindi è piuttosto chiaro. Supponiamo che il progetto che si sta sviluppando, si prefigga di risolvere questa necessità, per esempio implementando un sistema multisocial che dallo smartphone permetta di pubblicare con un solo flusso di operazioni, sui vari social; oggi esistono diverse app che svolgono questo compito, ma non è importante, si ipotizzi che non ne esistano.

Marco è in procinto di partecipare a una importante conferenza del settore e quindi ha il bisogno di comunicarlo. Il flusso di operazioni di Marco potrebbero essere le seguenti:

- Marco ha un bisogno insoddisfatto e per questo è triste (la metafora è per rafforzare il bisogno) perché nessuno sa del suo talk;
- Marco prende l’app social che si sta progettando (ipotizzando che sia già realizzata);
- seleziona i social network su cui pubblicare il post;
- inserisce del testo;
- mette il link al sito della conferenza;
- allega una foto;
- associa degli hashtag;
- pubblica;
- Marco adesso è contento perché ha risolto il suo bisogno.

Questo flusso di operazioni spesso viene codificato tramite il formato degli **User Journeys**, un altro strumento preso in prestito dalla disciplina del Service Design.

### Un accorgimento in più: la linea temporale

A mano a mano che tutti attaccano i propri cartellini, può essere utile che ogni persona disponga i propri in una sorta di **corsia orizzontale** cercando per quanto possibile di considerare la dimensione orizzontale come una sorta di **linea temporale** relativa alle operazioni che l'utente svolge all'interno del sistema (figura 9). Questa è una opzione che non si trova nel libro di Patton, ma che abbiamo sperimentato più volte e che può essere un valido ausilio per chiarire ulteriormente il flusso di interazione fra gli utenti e il sistema.

In questa fase i cartellini che sono organizzati in colonne rappresentano le funzionalità a supporto delle operazioni dei vari utenti. Sono possibili quindi dei duplicati, cosa che non costituisce un problema, potendo comunque mantenere un flusso *end to end*.

### Attività utenti e colonna portante

Al termine di questo lavoro, ogni **colonna** formata dovrebbe rappresentare una macroarea funzionale del flusso *end to end*.

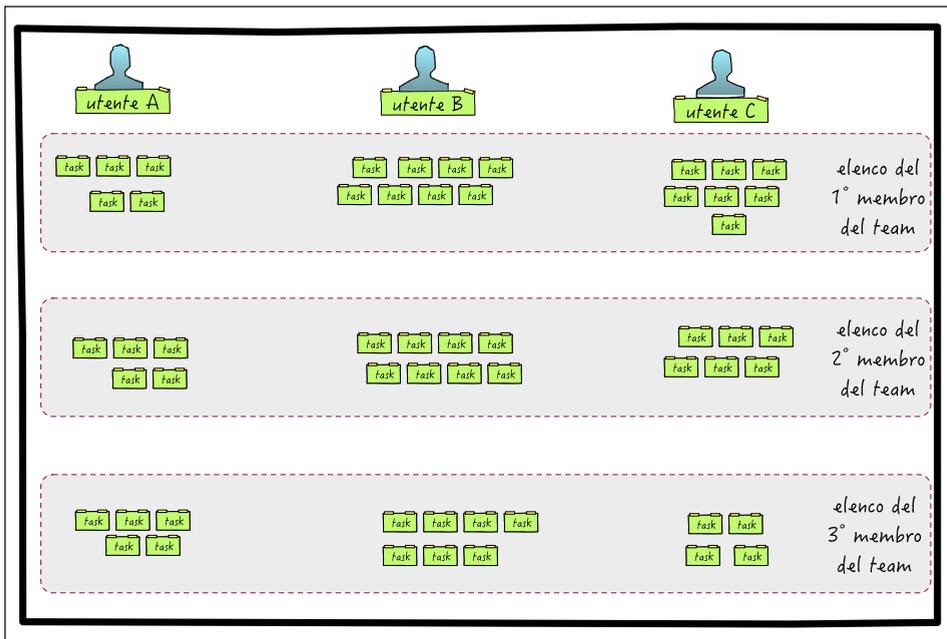


Figura 11.9. Dopo aver individuato le personas del sistema, si elencano le azioni che ogni utente svolge all'interno dell'applicazione. Ogni componente del team attacca alla parete i propri biglietti in modo da formare una sorta di corsia orizzontale.

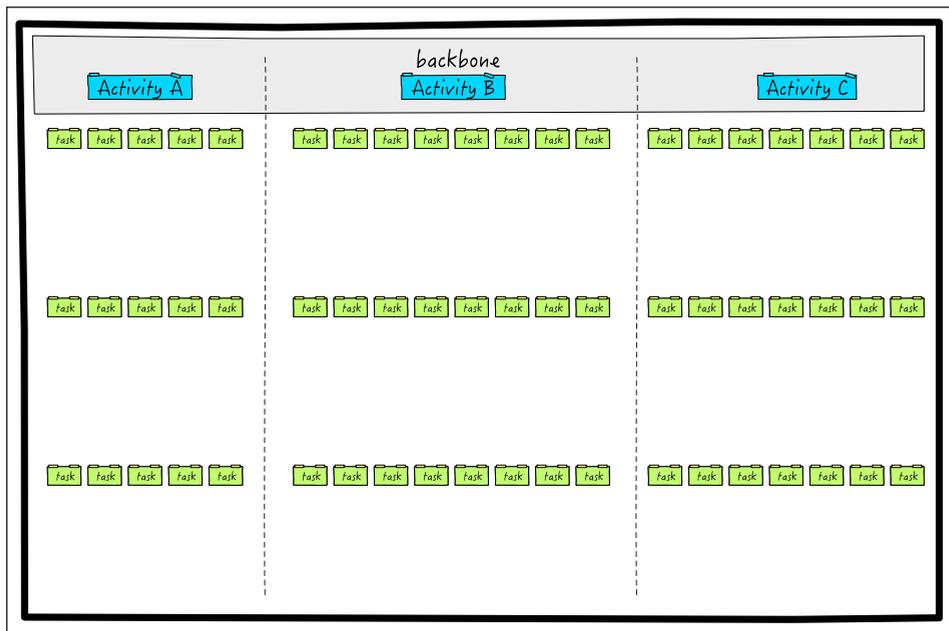


Figura 11.10. La backbone della mappa.

Queste macro aree funzionali sono quelle che Jeff Patton chiama **User Activities**: l'insieme delle attività da luogo alla **Backbone** dell'applicazione.

Durante l'attività di raggruppamento per identificare le **User Activities**, si consiglia di disporre i vari cartellini nella parte alta della board a formare, se possibile, un'unica riga che rappresenta la raccolta di **funzionalità ad alto livello**, che verranno chiamate **User Tasks** (un'altra parola presa in prestito dalla teoria della UX).

### Una visione comune sul valore delle attività

A questo punto, potrebbe essere utile analizzare se le varie persone (in particolare il **Product Owner** vs. il resto del team) hanno una **visione comune** circa l'importanza o il **valore** delle varie attività. Per fare questa valutazione, oltre a stimolare una discussione libera, si può utilizzare una tecnica presa in prestito dal **Value Stream Mapping**, che a volte viene presentata in forma di gioco dal nome **Buy An Issue**.

Supponendo di dare dei soldi **virtuali** ai partecipanti (potrebbero essere 100 € o 100 \$ o semplicemente **100 punti**) si chiede a ciascuna persona di disporre i questi soldi sulle varie attività in funzione del **valore**, dell'**utilità** o genericamente del **ROI** (indice di redditività) che ci si attende dall'implementazione di tale funzione.

Nella figura 12 è riportato il risultato ottenuto utilizzando per votare dei biglietti colorati: in questo caso ogni persona riceve un **numero prefissato** di biglietti che usa come gettoni o *fishes* per votare; la votazione avviene semplicemente attaccando i vari



*Figura 11.11. Con dei semplici pallini adesivi si possono far esprimere le preferenze sulle varie attività. Figura 21 – Il secondo livello intorno al nucleo centrale è quello del “come”. In che modo cambierà il comportamento dei vari attori e in che modo possono aiutarci a raggiungere gli obiettivi?*



*Figura 11.12. I vari partecipanti possono votare attaccando i pallini ai cartellini dei viaggi utente.*

biglietti ai task che si ritiene siano di maggior valore. Al termine della votazione, analizzando la disposizione dei biglietti o la macchia di colore, si può dedurre, in modo estremamente semplice ma efficace, quali sono i task il cui valore è ritenuto più elevato.

Raramente le varie persone danno gli stessi punteggi per le storie simili: proprio per questo motivo tale attività offre interessanti spunti di **discussione** finalizzata a capire se ci sia **intesa** su cosa si deve fare, su dove si produrrà **valore** e se sia chiaro del perchè si dovranno realizzare le varie funzionalità.

Dopo che il team ha discusso sulle varie differenze sui punteggi, si può quindi provare a unire le varie soluzioni, accorpendo, sostituendo, cambiando.

A questo punto il team si muove di fronte alla board camminando da destra a sinistra osservando le **varie attività** e cercando per un'ultima volta di vedere se ci si è dimenticati di qualcosa, se è necessario nuovamente spostare qualche task da una colonna a un'altra. Si possono prendere **appunti** sulla mappa, segnalare i cartellini corrispondenti alle parti più **rischiose** o indeterminate e provare a stimolare la discussione in modo informale sui vari cartellini appesi. Una conferma che il lavoro svolto è utile viene data se nascono

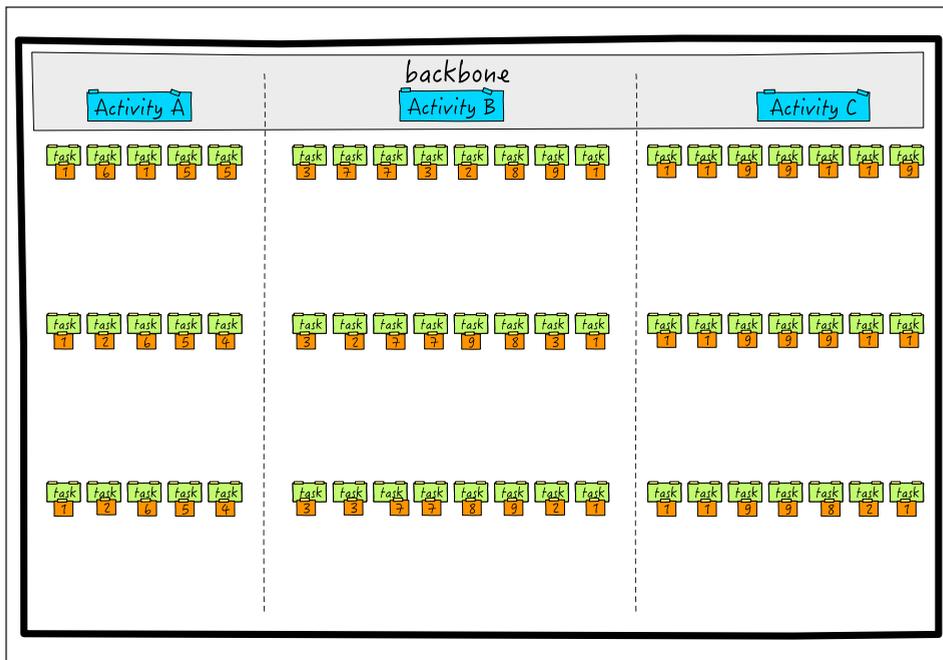


Figura 11.13. Tramite il gioco Buy An Issue, si può stimolare in modo estremamente rapido ed efficiente il confronto fra le persone del team. In questo esempio, ogni persona riceve un numero di bigliettini colorati prefissato e uguale per tutti. Questi bigliettini servono per votare i task scegliendo quelli che si ritiene abbiano maggior valore. In base alla disposizione dei biglietti (pattern analysis) e alle macchie di colore, si può rapidamente dedurre quali sono i task ritenuti più importanti.

commenti del tipo “mi era nota l’importanza di questa funzione, ma in questo contesto appare chiaro che sia ancora più importante e rischioso non implementarla in modo corretto”, oppure “questa cosa non l’avevo minimamente presa in considerazione”.

## Priorizzazione dei task

Il passo successivo è quello di introdurre la **dimensione temporale** rispetto alla quale il gruppo implementerà le funzionalità del sistema. In tal caso ha certamente poco senso ordinare le colonne con le **User Activities**, visto che queste sono funzionalità di alto livello delle quali tutte conterranno al loro interno alcune parti più urgenti insieme ad altre che potranno essere realizzate in un secondo momento.

Riprendendo l’esempio portato da Jeff Patton nel suo libro, si potrebbe immaginare un progetto che debba ideare, disegnare e poi costruire un’automobile. In questo caso le varie macrofunzionalità potrebbero essere le **User Activities** “impianto frenante”, “sistema di propulsione”, “impianto sterzante”, “carrozzeria”, “struttura portante” etc.

In questo caso è veramente difficile, se non inutile, utilizzare del tempo per capire se sia più importante la activity **impianto frenante** o quella relativa al **motore** o alle **ruote**. Dette in questo modo, sono tutte importanti e non è possibile stabilire in alcun modo a cosa convenga dare priorità: un’eventuale implementazione **minimum viable product (MVP)** di automobile che includa solo una o l’altra sarebbe comunque inutile se non addirittura pericolosa.

## Priorità dei sottocomponenti

Entrando invece nel dettaglio delle varie **Activity** è evidente come sia possibile optare per una qualche forma di priorizzazione basata sull’importanza dei **sotto-task**: per esempio, dell’impianto frenante potremmo rimandare a un secondo momento l’implementazione di un sistema di antibloccaggio delle ruote; l’ABS è importante ma può essere considerato un meccanismo da introdurre in un secondo momento, certamente dopo che si sia realizzato il prototipo da far vedere agli utenti finali. Discorso analogo per la carrozzeria o per il motore: ci sono molte parti e funzionalità che potrebbero essere inserite in un secondo momento, dopo la realizzazione del prototipo.

Tornando alla mappa che si sta realizzando, si può quindi introdurre un **asse verticale** sul quale spostare i vari task in base all’urgenza con la quale si vorranno implementare. Questa attività di ordinamento è ovviamente in carico al **Product Owner**, anche se potrà essere aiutato da altre persone; è comunque sua la **responsabilità ultima** dell’**ordine** delle cose da fare.

## La prima riga: MVP

A questo punto, osservando la **prima riga** che si forma in alto vicino alla linea della **backbone**, si potrà constatare che essa contiene le storie che possono dar vita a una **versione minimale** dell’applicazione con tutte le **funzionalità essenziali** per gli utenti finali. La prima riga rappresenta quindi la versione **Minimum Viable Product**

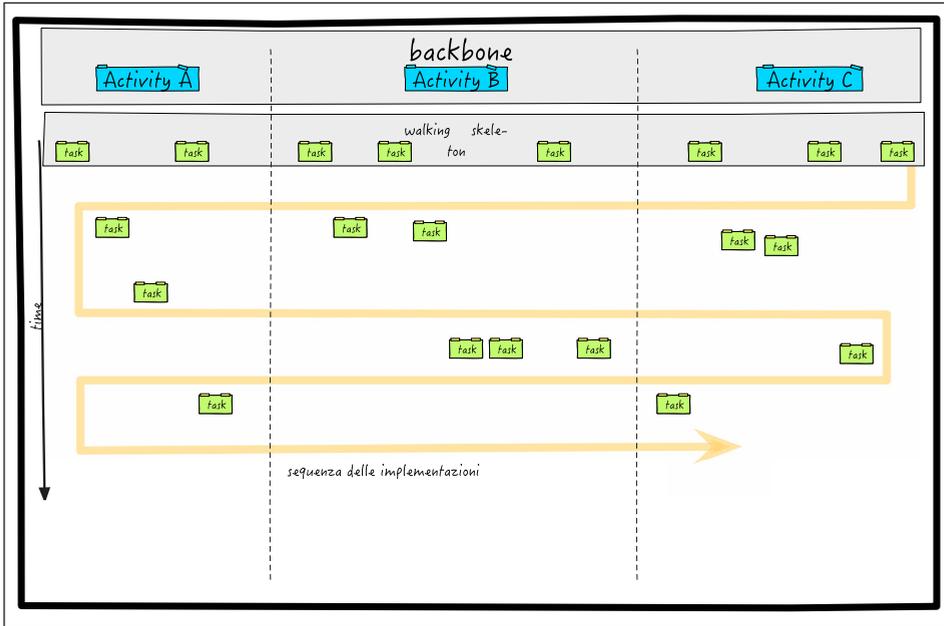


Figura 11.14. La sequenza di implementazione delle storie compone un ordinamento verticale con uno orizzontale, e può essere letta con uno schema che ricorda quello della scrittura bustrofedica.

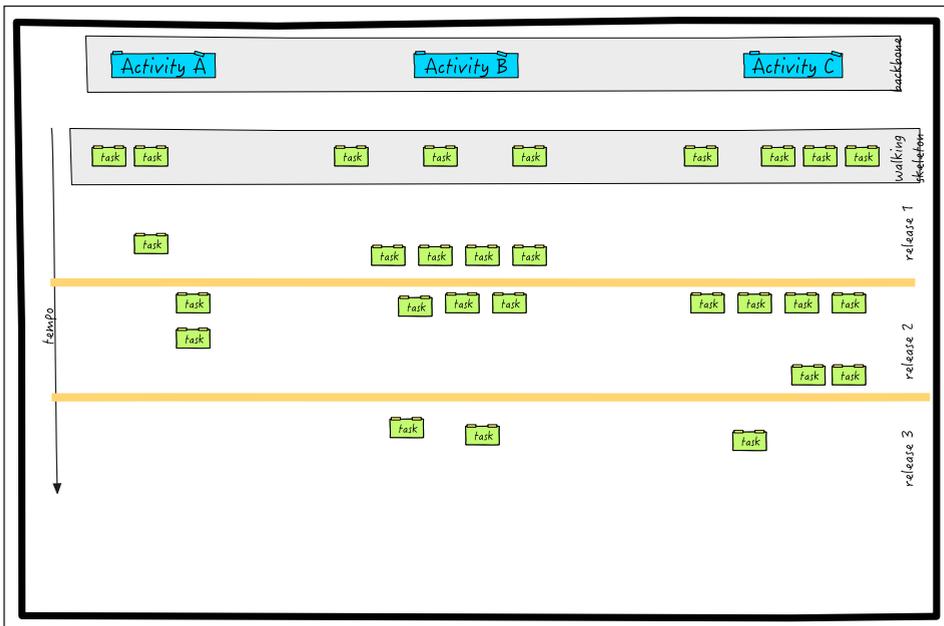


Figura 11.15. Prioritizzazione verticale dei task: nasce il walking skeleton. Tracciando delle linee orizzontali si possono creare delle corsie corrispondenti alle varie release.

(MVP) del prodotto: prendendo in prestito una definizione di Alistair Cockburn, è detta **Walking Skeleton**, con una immagine molto eloquente di uno scheletro (quindi solo la struttura portante) in grado comunque di camminare (quindi di svolgere alcune funzioni fondamentali). Sul sito di Cockburn si trova questa interessante definizione [5]:

«[il walking skeleton] è una implementazione minimale del sistema in grado di svolgere una piccola funzione end-to-end. Non necessita di usare l'architettura finale, ma dovrebbe collegare insieme i principali componenti architetturali. La parte architettonica e la parte funzionale possono poi evolvere in parallelo».

## Release

Dopo aver individuato la priorità circa l'implementazione dei vari task, grazie all'introduzione della dimensione **temporale**, si può raffinare questa suddivisione introducendo il concetto di **release**, disegnando delle strisce in orizzontale (figura 15).

## Conclusioni

La **User Story Map** è uno strumento molto utile e versatile che richiede alcune conoscenze e un po' di pratica, ma che risulta molto chiaro e capace di irradiare conoscenza a chi lo utilizza.

Consente una definizione **collaborativa** dei vari task, una loro **organizzazione** in macroaree funzionali, una prioritizzazione in termini di **valore** delle varie attività e di **tempi** di realizzazione delle varie funzioni.

Insieme agli altri strumenti visuali analizzati nei capitoli precedenti, la User Story Map contribuisce in maniera strutturata ma flessibile ad affrontare il percorso che porta dalla visione al prodotto finale.

## Riferimenti

[1] Jeff Patton, *User Story Mapping: Discover the Whole Story, Build the Right Product*. O'Reilly, 2014

[2] Giovanni Puliti, *#Play14. Reportage dalla unconference sul serious game*. MokaByte 194, aprile 2014

[3] Hoang C. Huynh, *Prototipare con MEAN.io. II parte: Le cose importanti prima di tutto*. MokaByte 198, settembre 2014

[5] Walking skeleton

<http://alistair.cockburn.us/Walking+skeleton>

[6] Adzic G. – Evans D. – Korac N., *Fifty Quick Ideas To Improve Your User Stories*. Neuri Consulting LLP, 2014

[7] How to split a user story

<http://goo.gl/0k21t0>

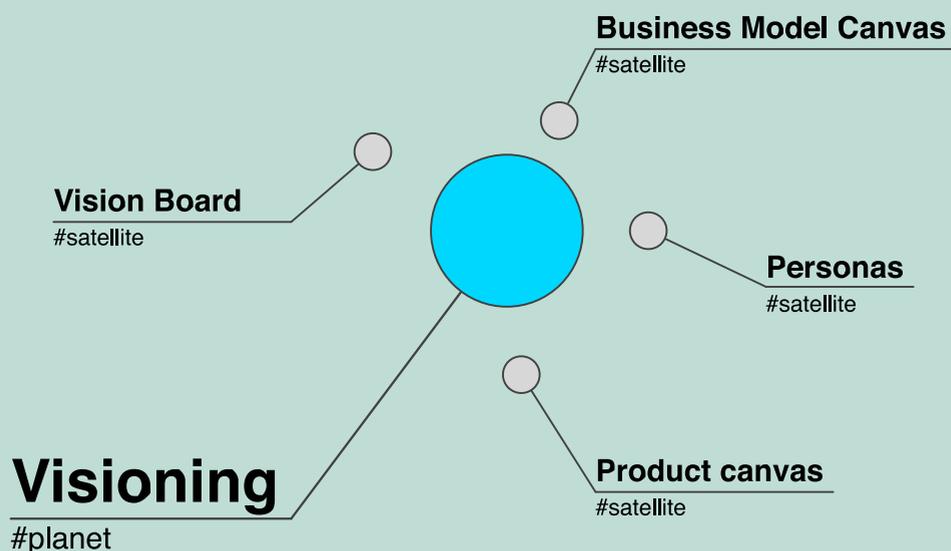
[8] Silent Brainstorming: A Guide To Using Post-its

<http://goo.gl/65WdeZ>



# Capitolo 5

## Raccogliere i requisiti con il Product Canvas



## I requisiti con il Product Canvas

Una volta definita e condivisa la **Vision Board**, si può provare a stilare un primo **elenco** delle cose da fare creando una prima versione del **Product Backlog**; per svolgere questo compito si può utilizzare la tecnica vista in precedenza dello **Story Mapping**, oppure sfruttare un altro strumento come il **Product Canvas**. Nella figura 16 è mostrato il tipico impiego della **Product Canvas**, che si inserisce all'interno di un processo **iterativo e incrementale** e che consente di trasformare un'idea in un elenco di **funzionalità**, in questo caso **storie utente**, da implementare nella fase di lavorazione vera e propria.

Per comprendere meglio l'utilizzo di questo strumento, conviene entrare nel dettaglio di come è fatta la "lavagna". La figura 17 riprende un modello proposto da Roman Pichler [2] e ci mostra le varie sezioni della Product Canvas.

La **Product Canvas** normalmente viene "compilata" in un **workshop** in cui il team arriva a completare le varie parti della board in forma di prima realizzazione. In genere, la durata di tale evento è quello di una giornata completa.

Vediamo di seguito le diverse sezioni e le informazioni che andranno inserite.

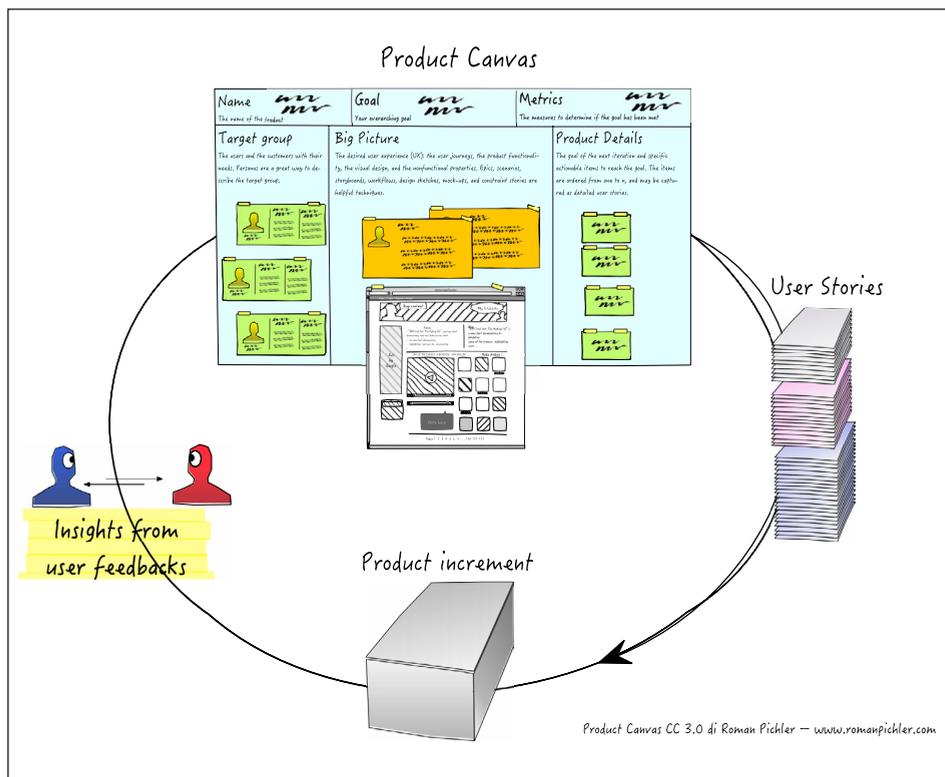


Figura 11.16. Lo schema di lavoro iterativo e incrementale all'interno del quale si inserisce l'uso della Product Canvas.

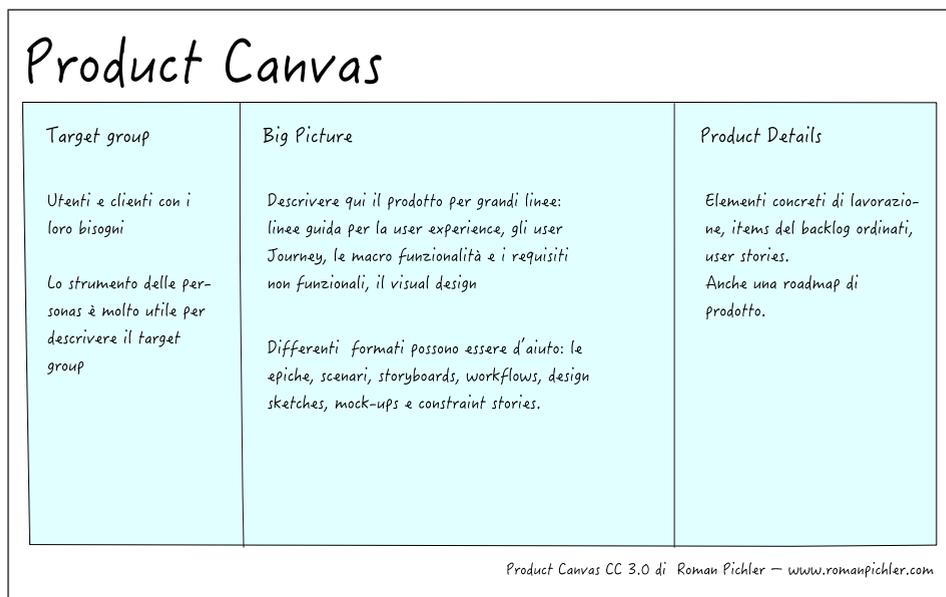


Figura 11.17. La Product Canvas con le sue sezioni.

## Nome

Il **nome** del prodotto che si deve realizzare. Specificare anche la versione del prodotto, quando è un redesign o la nuova implementazione di qualcosa che già esisteva.

## Goal

Il **goal** rappresenta l'obiettivo che si vuole raggiungere con la realizzazione di questo prodotto o tramite questa nuova release di un prodotto già esistente: trovare nuovi utenti, soddisfare nuove richieste di mercato e così via. Roman Pichler, nel suo sito, propone l'uso della **Go Product Roadmap** [3] come strumento di pianificazione del prodotto orientata alla definizione degli obiettivi. Nel caso in cui si sia scelto di utilizzare questo strumento, per approfondire l'analisi del tema "obiettivi", si potranno riportare nella Product Canvas le parti corrispondenti.

## Metriche

La sezione **metriche** serve per specificare dei parametri di valutazione che possano permettere di capire quanto quello che si sta realizzando rispetta gli obiettivi prefissati. Anche in questo caso la Go Product Roadmap può essere di aiuto.

## Target Group

La definizione del **gruppo di utenti tipo** serve per individuare chi potrebbero essere gli utilizzatori tipici di questo prodotto, e capire successivamente i loro bisogni. Per

compilare questa parte si può usare il formalismo delle **personas** e per questo si potrebbero utilizzare le personas individuate durante la fase di visioning e che potrebbero essere qui riutilizzate in forma sintetica prendendo direttamente spunto da quanto inserito nella **Vision Canvas**. La Product Canvas offre infatti una visione di sintesi da un differente punto di vista, per cui non deve meravigliare che si ricavano in questa fase informazioni differenti a complemento di quanto visto durante la vision.

Visto che nella **Product Canvas** il focus è sulle funzionalità del sistema, potrebbe essere utile in questo momento provare a **ordinare** le **personas** in **funzione dell'importanza** dell'utente: quello che paga o uno sponsor importante, o un'altra categoria importante. Si può anche ordinare le personas in funzione dell'importanza dei bisogni: urgenza, funzionalità "rivendibilità", ROI maggiore o altro. In questo modo sarà certamente più semplice poi stabilire la priorità delle funzionalità da implementare. Alle **Primary Personas** così individuate dovrebbero infatti corrispondere funzionalità che poi saranno in cima al backlog di prodotto.

### Big Picture

La **Big Picture**, vale a dire il "quadro d'insieme", descrive cosa è necessario fare per soddisfare i bisogni delle personas: dovrebbe fornire la visione di insieme del prodotto ad alto livello; concettualmente ricorda l'**indice di un libro**: ne descrive il contenuto senza entrare nel dettaglio.

Spesso in questa parte si inseriscono gli **User Journeys** ovvero la descrizione delle varie operazioni che l'utente svolge quando utilizza il prodotto: si collega, apre il menu, cerca e sceglie, compra, stampa, esce etc. Per la definizione dei "viaggi" possono usare tecniche come gli **Scenari** [4] o lo **User Story Mapping** che abbiamo visto al Capitolo 4.

Durante questa fase, è utile anche la raccolta di informazioni relativamente alla parte non funzionale, ai requisiti tecnici e alle cosiddette **constraint stories** che possono in qualche modo impattare sulla esperienza utente o sulla architettura software.

Per comprendere meglio l'esperienza utente spesso si realizzano in questa fase delle immagini di interfaccia utente, che poi si "attaccano" direttamente alla board. Per questo lavoro si possono usare diverse tecniche come **design sketches** e **mock-up** [5], oppure applicazioni per la prototipazione rapida delle interfacce che permettano di provare una versione semi funzionante in tempi rapidissimi [6]. In questa fase, è importante concentrarsi sugli aspetti strutturali del design grafico (layout di massima) e sugli aspetti critici legati all'architettura dell'informazione.

### Product Details

Infine, nella sezione **Product Details**, si inseriscono gli elementi che dovranno essere messi in lavorazione nella prossima iterazione (lo sprint zero se siamo a inizio progetto).

Questo elenco può essere prodotto analizzando i risultati del punto precedente. Per esempio i vari passi dei diversi "viaggi utente" possono essere visti come gli elementi a

grana grossa (**epiche** o **features**) del **Product Backlog**. Se si è utilizzata la tecnica dello **User Story Mapping**, il set delle storie individuate per il prossimo sprint potrebbe essere inserito così com'è in questa parte.

Molto utile usare la Product Canvas in congiunzione con la tecnica dello User Story Mapping perché consente di produrre contenuti interessanti tramite un lavoro strutturato e rigoroso. Nella figura 18, infine è riportato un esempio di una Product Canvas per un progetto mobile.

### L'utente prima di tutto

La canvas è progettata per permettere che il processo relativo alla generazione delle varie parti si traduca in un **flusso** di informazione da sinistra a destra, ossia dalle personas ai dettagli del prodotto. In questo, modo si pone sempre **l'utente** al **centro** del proprio lavoro concentrando quindi gli sforzi sui suoi bisogni e sui suoi desideri, permettendo la realizzazione di un prodotto che li possa soddisfare.

### Il flusso di informazioni

Interessante notare come questo **flusso di informazioni** non si realizzi solamente all'interno della board, ma sia lo stesso tipo di percorso che si ritrova più in grande nel processo di lavorazione (figura 16): in quel caso, grazie al processo iterativo e incrementale messo in atto, ogni incremento nella realizzazione del prodotto finito permette

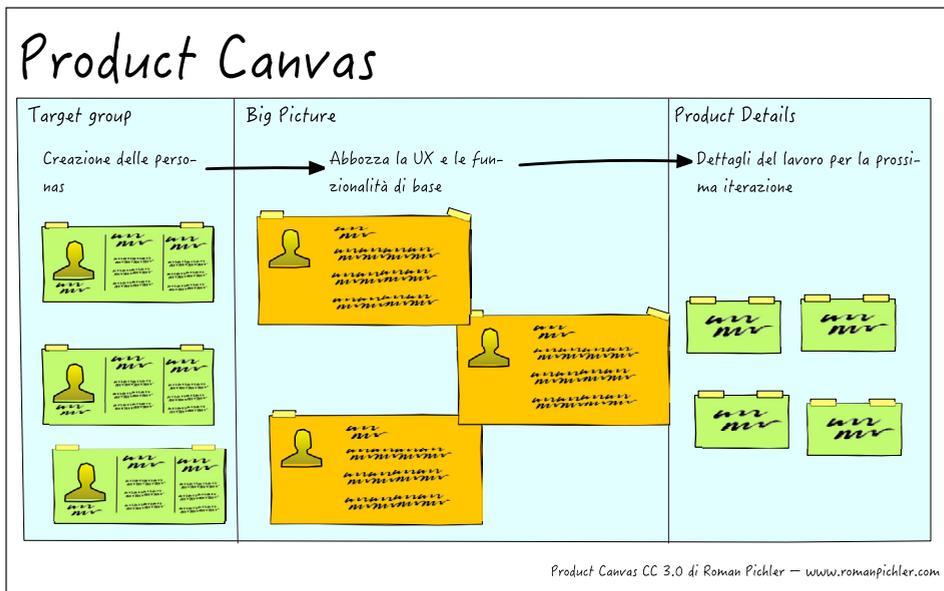


Figura 11.18. Il processo di produzione delle informazioni segue un flusso da sinistra a destra ed è incentrato sul concetto di utente del prodotto.

di ricavare informazioni direttamente dagli utenti finali e quindi di migliorare la conoscenza del dominio, delle esigenze da soddisfare, ossia, in definitiva di quello che si deve fare. Si realizza quindi un circolo virtuoso in cui la Product Canvas può essere vista come una sorta di **strumento di apprendimento** utile sia per abbozzare le idee iniziali che per raffinarle in un secondo momento.

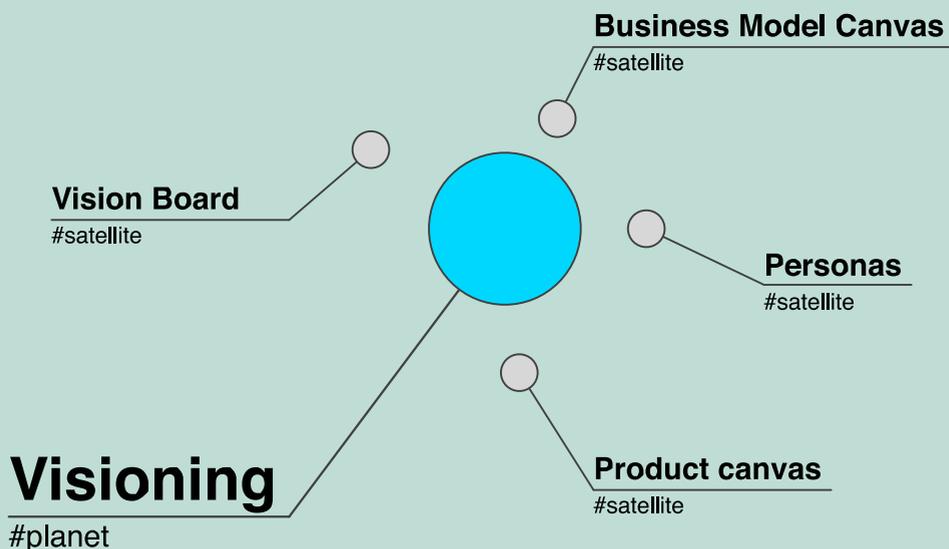
Per questo motivo non deve stupire se la Product Canvas subisce un processo di **correzione** e **modifica** per inserire nuovi dettagli, nuove informazioni o per cambiare quanto già inserito, man mano che si scoprono nuove cose. Per questo spesso le board sono piene di foglietti, scritte, scarabocchi: sono la brutta copia del tema in classe, dove continuamente si fanno correzioni, si aggiungono parti e se ne cancellano altre.





# Capitolo 6

## Organizzare i deliverables di progetto tramite le Impact Maps



## Un'alternativa intuitiva

Per arrivare a una organizzazione delle storie utente e al relativo raggruppamento per funzionalità e necessità utente, un utile strumento, da usare in alternativa ai vari story mapping e product board, è la **impact map**, introdotta da Gojko Adzic [1]: si tratta di un tool dall'approccio estremamente intuitivo — si ispira alle **mappe concettuali** o *mind maps* — che connette in maniera visuale i vari *deliverable* di prodotto e gli obiettivi di business.

## Come funziona una impact map

Una **impact map** è di fatto una mappa mentale (**mind map**) che offre una visualizzazione dello *scope* di un prodotto e mette in relazione le varie ipotesi connesse per la sua realizzazione.

La sua realizzazione avviene in modo collaborativo e incrementale, durante la discussione che viene condotta e in un certo senso facilitata rispondendo a 4 semplici domande:

- **Perché?** Per quale ragione dovremmo realizzare questo prodotto? Qual è lo scopo? Quali sono gli obiettivi che ci poniamo? Quale la vision?
- **Chi?** Quali sono gli attori che potrebbero abilitare/influenzare il raggiungimento dell'obiettivo?
- **Come?** In quale modo potrebbe cambiare il comportamento degli attori una volta realizzato il prodotto? Ma anche, come team di sviluppo in che modo possiamo abilitare questo cambiamento?
- **Cosa?** Che cosa potremmo realizzare, sempre dal punto di vista del team di sviluppo.

## Perché: definire l'obiettivo

Rappresenta il nodo centrale della mappa e permette di rispondere alla domanda forse più importante: **perché** dovremmo fare questo prodotto/servizio/iniziativa?



Figura 11.19. Il nodo centrale della Impact Map è rappresentato dalla risposta al “perché” si vuol realizzare il nostro prodotto.

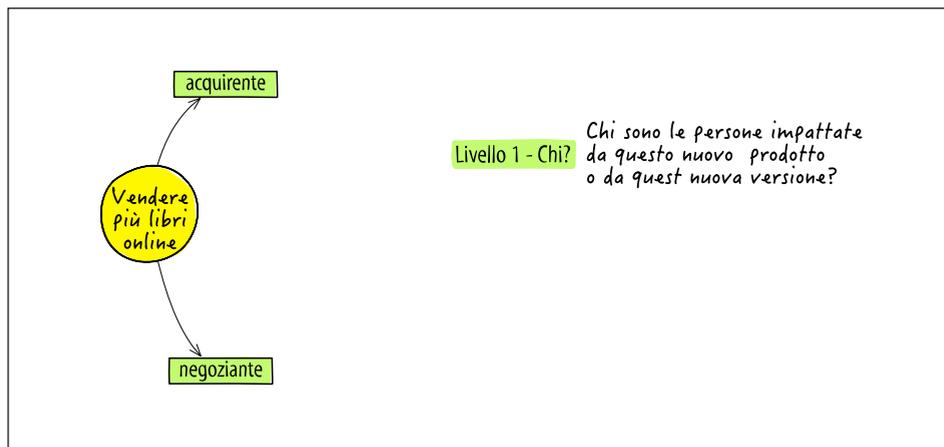


Figura 11.20. Intorno al nucleo centrale, sviluppiamo un primo livello con le risposte alla nostra domanda “chi?”, riguardante sia chi è il destinatario impattato dal prodotto che vogliamo sviluppare, sia chi può produrre l'effetto desiderato.

Rispondendo a questa domanda si riesce a descrivere l'obiettivo che si intende raggiungere. Una **impact map** è pensata per descrivere un prodotto nella sua **interezza**: nel nodo centrale della mappa troviamo quella che forse è la parte più importante e che permette spesso di descrivere in modo sintetico la vision di prodotto, o meglio **una parte della vision** (obiettivi, scope).

Il focus dovrebbe sempre essere sull'individuare ed evidenziare le **necessità** degli utenti, più che sul fornire soluzioni: pensare alle necessità che si vogliono soddisfare, invece che al modo in cui fornire un supporto. Per raggiungere questo obiettivo si potrebbe utilizzare una frase, uno slogan, o anche qualcosa di più strutturato come un **elevator pitch**.

Spesso non è detto che la visione sia completa a inizio progetto: per questo può valere la pena iniziare a compilare la mappa inserendo una versione abbozzata della visione per poi raffinarla e rivederla man mano che si chiariscono meglio gli obiettivi tramite l'analisi e la compilazione degli altri livelli della mappa.

**Scope** e **vision** di un progetto dovrebbero essere sempre ben chiari a tutti i membri del team, e non solo di questo. Pertanto spesso è utile rendere la mappa — ma soprattutto il suo nodo centrale — accessibile e visibile, per esempio appendendola a una parete.

È infatti estremamente importante che tutti sappiano cosa fare e come muoversi nel caso si debbano prendere decisioni in modo rapido, intervenire in emergenza, apportare correzioni al progetto, gestire cambi di rotta o tamponare emergenze.

### Chi: per chi stiamo sviluppando il prodotto

Questo livello della mappa serve per rispondere alla seguente domanda: **chi può produrre** l'effetto desiderato? Chi potrebbe impedirlo? **Chi** sono gli **utenti/destinatari** a

cui ci vogliamo rivolgere? **Chi** ne sarà **impattato**: l'uso di questa parola non è casuale, essendo alla base del concetto di **impact map**.

A differenza di altri strumenti, in questo caso si prendono in considerazione non solo gli utenti finali, ma anche quelle persone/realità che possono **influenzare** la realizzazione del prodotto con le loro decisioni.

Un modo per definire il concetto di qualità — molto usato all'interno della comunità agile — è “valore rilasciato all'utente finale” che quindi deve essere ben compreso e descritto; devono essere comprese le sue necessità, i suoi obiettivi e le sue preferenze per indirizzare meglio le varie componenti o declinazioni del prodotto finale. Di fatto la maggior parte dei tool o tecniche di Product Definition insistono molto su questo aspetto: si pensi alle **Personas**.

Nel processo di creazione di una Impact Map la definizione degli attori può essere utile per comprendere le loro necessità e quindi riuscire a prioritizzare meglio le cose da fare.

Un modo per qualificare gli attori potrebbe essere per esempio di suddividerli in base al coinvolgimento con il prodotto che si deve realizzare. Alistair Cockburn suggerisce questa semplice classifica:

- **Attori primari** i cui obiettivi devono essere soddisfatti pienamente.
- **Attori secondari** che non sono coinvolti direttamente ma possono fornire un servizio a supporto del prodotto che si deve realizzare. Sono coinvolti o comunque possono essere un valido supporto.

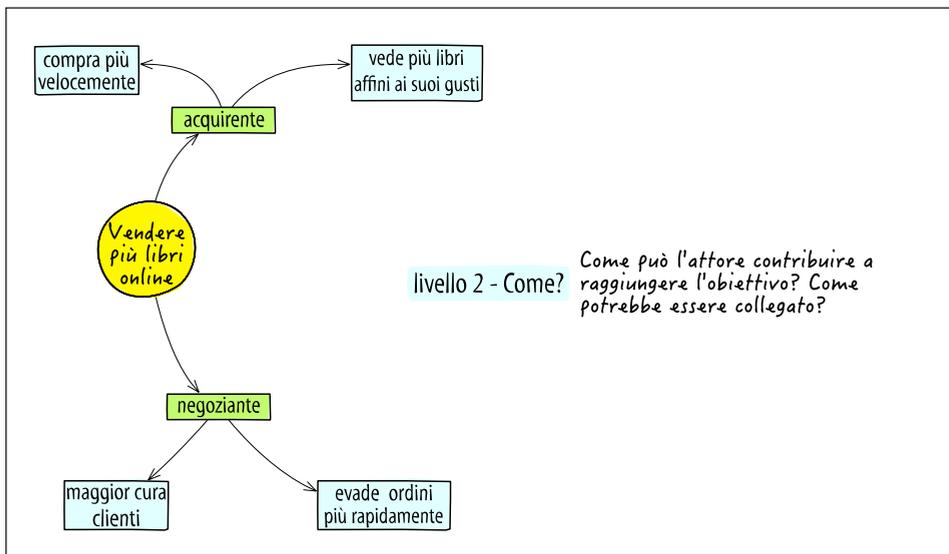


Figura II.21. Il secondo livello intorno al nucleo centrale è quello del “come”. In che modo cambierà il comportamento dei vari attori e in che modo possono aiutarci a raggiungere gli obiettivi?

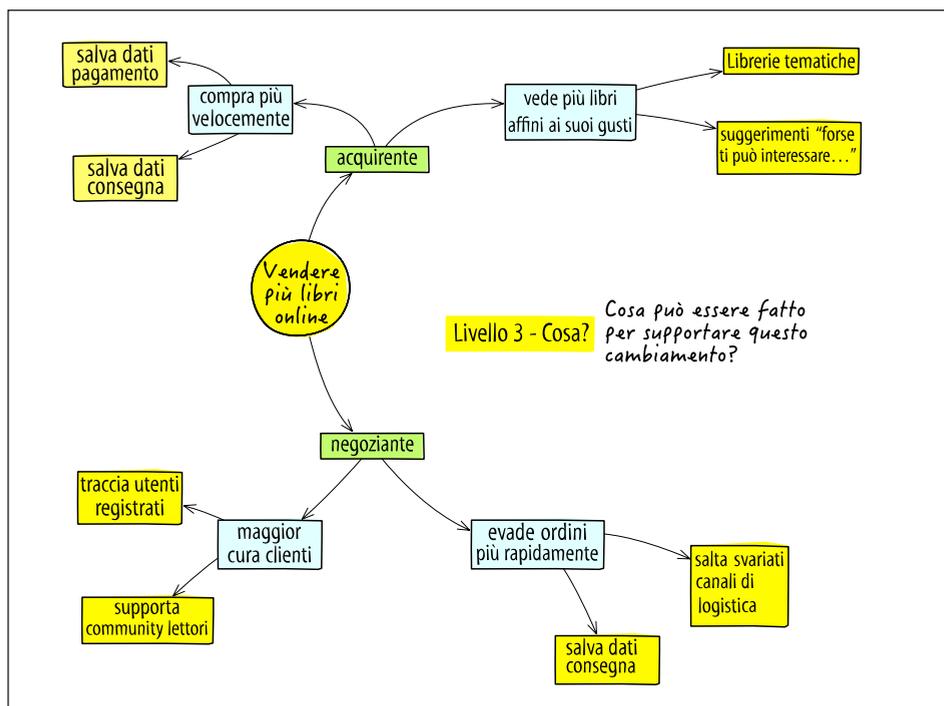


Figura 11.22. Con l'aggiunta del terzo livello, quello del "cosa", si arriva a individuare elementi che rappresentano i deliverables, le caratteristiche del prodotto ma anche attività relative all'organizzazione.

- **Off stage actors**, i quali sono interessati di riflesso dal sistema ma non lo usano direttamente o non forniscono un servizio.

## Come: in che modo realizzare il prodotto

Questo livello suggerisce il "come" si potrebbe realizzare il prodotto e risponde alla domanda: "In che modo dovrebbe cambiare il comportamento degli attori? Cosa ci aspettiamo che succeda? Come possono concretamente aiutarci ad ottenere gli obiettivi? Come possono invece impedire od ostacolarci?"

Questo livello della mappa quindi ci aiuta a capire gli obiettivi degli utenti finali, quali sono le cose che vogliono ottenere; ci permette quindi di descrivere il cosiddetto **change behaviour** dell'utente e quindi quali dovranno essere gli **impatti** del lavoro che si sta realizzando.

Spesso un impatto evidenzia un **cambiamento di comportamento**: per esempio potrebbe essere una buona descrizione "vendere libri online in modo più semplice e rapido", mentre genericamente "vendere libri online" potrebbe essere troppo generico e non introduce nessuna variazione: ci sono già molti e-commerce che lo fanno.

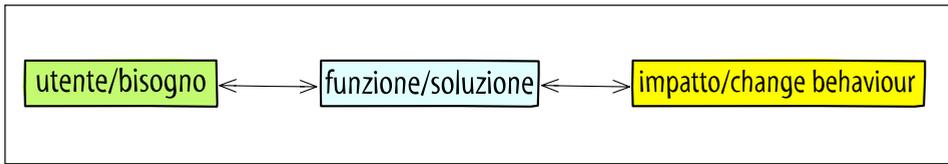


Figura II.23. Nella stesura dei requisiti è necessario concentrarsi non su un solo aspetto, ma collegarli nella “terna” rappresentata da “utente / bisogno”, “funzione / soluzione” e “impatto / change business behaviour”.

Interessante notare come le differenze fra gli attori evidenziate durante la compilazione del livello precedente (quello del chi), in questo caso si potranno tradurre in impatti che potrebbero essere legati da un legame di priorità, o potrebbero essere complementari o del tutto incompatibili fra loro.

Per non rischiare di divagare troppo, si dovrebbe cercare di focalizzare l’attenzione solo su quegli impatti che possono essere utili a muovere il progetto nella direzione giusta. In pratica, un trucco potrebbe essere di rimanere sulle attività di business da implementare, non genericamente divagando su tutte le idee che ci passano per la testa; altra indicazione fondamentale è di non cadere nel tranello di elencare un set di funzionalità del prodotto finale.

### Cosa: cosa facciamo per portare gli impatti

Questa sezione della mappa permette di rispondere alla domanda: “Come team di sviluppo/azienda/organizzazione, **cosa possiamo fare** per supportare o implementare il gli impatti desiderati? Ossia cosa possiamo fare per abilitare il cambio di comportamento?”.

Gli elementi qui presenti corrispondono ai cosiddetti *deliverables*, le funzionalità del software, oppure semplici attività organizzative.

Dovrebbe ormai essere chiaro quanto sia importante collegare i deliverables di progetto con i corrispondenti obiettivi di business e per ogni collegamento argomentarne le implicazioni (ossia gli impatti) di business: detto in altro modo è necessario definire la terna illustrata nella figura 23.

Collegando i deliverables con gli obiettivi e le conseguenze, è come se la mappa ci aiutasse a definire il filo logico che collega una scelta di business con le eventuali soluzioni che possono risolverlo. L’**obiettivo** di una **Impact Map** è esattamente questo, ossia creare un **collegamento** fra **quello** che deve essere fatto, **perché** deve essere fatto e **per chi**.

Grazie alla struttura organizzativa a grafo, una impact map è di grande aiuto per organizzare gli eventuali rilasci intermedi: i rami (suddivisione verticale) o i livelli (suddivisione orizzontale) potrebbero essere associati alle milestone o release. Anche le eventuali priorità che si instaurano fra gli attori sono di grosso aiuto per definire la sequenza delle cose da realizzare.

Questo livello della mappa è il più importante e per questo dovrebbe essere completato in modo incrementale e iterativo.

Man mano che si rilasciano nuove parti, si potrebbero aggiornare le parti ancora poco complete. I deliverables sono opzioni, non tutto quello che verrà qui rappresentato potrebbe essere implementato.

Non conviene fissarsi fin da subito sui dettagli, ma è preferibile partire dalla definizione di alto livello dei deliverables; successivamente si potranno raffinare tramite le consuete tecniche di splitting come si fa comunemente sugli elementi di un backlog.

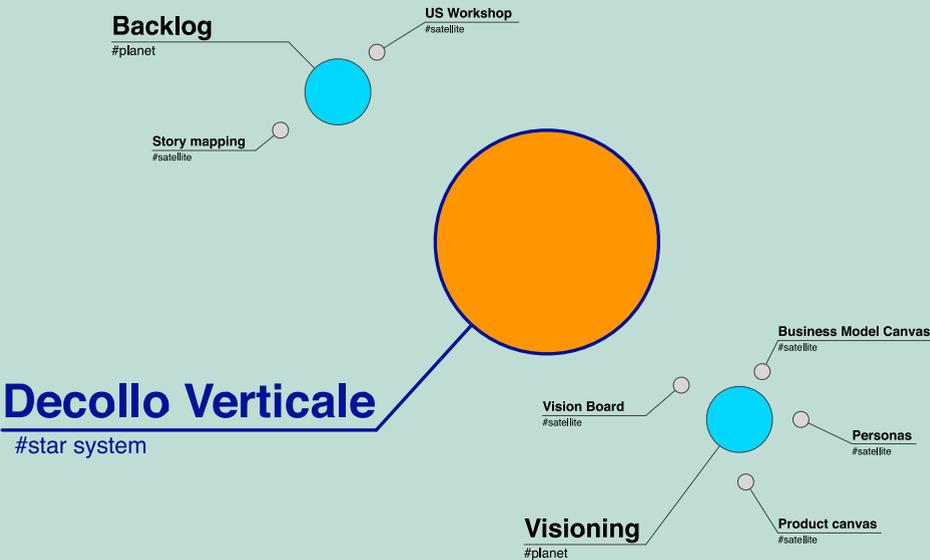
## Riferimenti

- [1] Gojko Adzic, *Impact Mapping: Making a big impact with software products and projects*. Provoking Thoughts, 2012



# Capitolo 7

## Mettiamo tutto insieme



## Strumenti, ma non solo...

Fin qui, abbiamo illustrato **strumenti** per svolgere uno dei compiti più difficili: **trasformare un'idea in un prodotto**. E se li abbiamo visti all'opera nell'arte di sviluppare software, non va dimenticato che molto di quanto raccontato in questa Parte 2 del libro è applicabile anche in altri contesti produttivi. Si tratta di un lavoro che comprende la definizione dell'**idea di base**, la **formalizzazione dei requisiti** e l'**impostazione del progetto**.

La **condivisione** e la **comprensione comune** dell'idea sono forse l'ostacolo più grande per far partire un progetto. Si possono trascorrere giorni e giorni in interminabili riunioni per condividere l'idea a tutti i livelli: possiamo coinvolgere ruoli direttivi e finanziari, project manager, sviluppatori, marketing, ufficio vendite e supporto clienti... per poi scoprire di avere una idea che è solo **superficialmente condivisa** e che il consenso sul progetto è dato più sulla fiducia che sul contenuto. Spesso la **comunicazione verbale** non aiuta più di tanto, perché molti aspetti dell'idea del prodotto in molti casi **non** sono facilmente condivisibili a parole.

## Vantaggi e criticità della documentazione scritta

Trasmettere le informazioni **per iscritto** è una prassi molto comune che ha indubbi vantaggi:

- **formalizza** il pensiero e la comunicazione verbale, **organizzando** in modo logico i **contenuti**;
- consente di mantenere uno **storico** delle scelte adottate, e magari anche delle ragioni alla base di tali scelte;
- il documento **scritto** ben si presta a inserirsi in un **processo di approvazione**.

A fronte di questi vantaggi, quando si decide di scrivere un **documento testuale**, indipendentemente dal formalismo utilizzato, spesso si corre il rischio di concentrare l'attenzione sul "formato" e sulla compilazione del documento piuttosto che sul produrre **valore** per il cliente.

Il secondo punto dell'Agile Manifesto ("Il software funzionante più che la documentazione esaustiva") [1] diventa sia un faro guida sia un punto di discussione forte. Come faccio a **non** scrivere documentazione? In Agile si scrive **documentazione**? Quanta? Quale?

Prima di rispondere a queste domande, conviene anzitutto capire quale sia l'**utilità** che risiede nella documentazione dal punto di vista del cliente finale.

## Il valore del documento

In tal senso la documentazione prodotta, soprattutto quella realizzata nelle fasi iniziali di progetto, ha valore se è **condivisa** e **compresa** da tutte le persone coinvolte: lo scopo è quello di permettere di condividere l'idea del prodotto o servizio che si vuole realizzare.

Come già più volte detto, solo la **condivisione** della **visione** di prodotto, degli **obiettivi** e delle **modalità** per raggiungerli, consente di realizzare un prodotto che

soddisfi i bisogni di business dell'utente, cosa che in definitiva è il vero scopo di un progetto software.

Quindi, lo **scopo della documentazione** è **condividere conoscenza** e permettere la collaborazione. Guardando le cose da questo punto di vista, l'obiettivo di produrre documentazione è certamente nobile e utile. Quello che ci si potrebbe chiedere è se ci possano essere **strumenti più efficaci rispetto** alla scrittura di **documenti testuali** che poi spesso non catturano la curiosità degli interessati e finiscono per rimanere in qualche remota cartella.

### Canvas come documentazione

Probabilmente l'uso di **lavagne** per disegnare ma anche e soprattutto per **parlare insieme** potrebbe essere una valida alternativa. In questa parte del libro abbiamo già abbondantemente affrontato questo tema, proponendo una serie di strumenti utili per la fase di definizione dei requisiti nonché per la successiva validazione.

Abbiamo parlato di come **identificare** e **formalizzare** la visione di prodotto (**Vision Board**) e di come **individuare** e validare successivamente il **modello** di **business** (**Lean Canvas**); abbiamo affrontato il tema della **traduzione** dei bisogni utente in un **elenco di funzionalità** da implementare (**User Story Mapping** e **Product Canvas**).

In questo ultimo capitolo della Parte II, ricapitoliamo il flusso della lavorazione nel suo complesso, ossia come i vari tool visti fin qui possano essere utilizzati per passare dall'idea al prodotto.

### Un riassunto sulle canvas: “from vision to backlog... and back”

Il lavoro di costruzione di un prodotto complesso è frutto di un processo iterativo e incrementale in cui continuamente si parte dalle premesse per realizzare una parte del prodotto finito e, tramite la prova sul campo delle funzionalità inserite, si prova a rivedere, aggiungere e modificare le premesse.

La frase “from vision to backlog... and back” rappresenta questo approccio: la parte del **back**, ossia l'approccio **iterativo** e **incrementale**, permette di trasformare un processo tipicamente **sequenziale** — dalla vision al modello di business alla definizione dell'elenco delle cose da fare — in un processo **iterativo incrementale** tipicamente agile.

Di fatto senza il cortocircuito che, dalla fase di realizzazione, ci fa ritornare ciclicamente a quella di vision e raccolta dei requisiti, quanto visto finora non sarebbe altro che una forma rivista e corretta del caro vecchio waterfall.

### Le varie canvas in azione, ossia un approccio iterativo, incrementale e complementare.

Come riportato nella figura 1 al Capitolo 1 di questa Parte II, per passare **dall'idea al prodotto**, la prima cosa da fare è lavorare sulla **visione** per esempio tramite il workshop con la **Vision Canvas**: lo scopo è capire perché si vuole fare questo prodotto, per chi è il prodotto, quali bisogni si vogliono risolvere etc.

Lo scopo in questa fase, lo ricordiamo, è capire in che modo il nostro prodotto “funzioni” in termini di sostenibilità economica e possibilità di guadagno, e come gli utenti saranno interessati a continuare a utilizzarlo, magari pagando per certe funzioni, è compito della **Business Model Canvas**, che si occupa, appunto del **modello di business**.

Quando infine si arriva a entrare nello specifico delle **funzionalità** del prodotto, la **Product Canvas** può rappresentare un interessante punto di congiunzione fra le due attività che si sono svolte parallelamente in precedenza.

Da un lato la **Product Canvas** permette di approfondire in modo efficace un aspetto solamente accennato nella **Business Model Canvas**, ossia quello della **Value Proposition**; dall'altro la parte di raccolta delle varie **personas** offre una visione di sintesi di quello che nella **Vision Canvas** è invece analizzato nell'elenco degli utenti e dei bisogni.

### Tre strumenti, un solo processo

**Vision Canvas**, **Business Model Canvas** e **Product Canvas** sono quindi tre strumenti che offrono una visione **complementare** del sistema che si deve realizzare. Tipicamente, la corretta maniera di usarli è in modo continuativo, iterativo e incrementale:

- in genere si parte dall'impostazione della **vision**;
- si passa subito dopo alla definizione del **modello di business**;
- appena si hanno sufficienti informazioni, si può iniziare a impostare lo studio del **prodotto** tramite la **Product Canvas**;
- da questa, direttamente o tramite l'ausilio dello **User Story Mapping**, si può ricavare una prima versione del **Product Backlog Items**;
- infine, grazie a una metodologia iterativa e incrementale come Scrum, si potrà arrivare alla realizzazione di una prima versione del prodotto che implementi in forma minimale una prima risposta ai bisogni degli utenti individuati (il cosiddetto MVP);
- a questo punto, grazie alla raccolta del feedback degli utenti che hanno modo di provare una prima release di prodotto, si potrà procedere alla **revisione** o al **completamento** delle varie **canvas**.

Se il progetto è gestito tramite Scrum, ad esempio, le informazioni raccolte durante le sprint demo potranno fornire spunti per **completare** o **modificare** sia la parte di **vision**, sia quella di **business**. E, molto probabilmente forniranno dati interessanti anche per completare la raccolta delle **funzionalità** da implementare o per apportare modifiche alla modalità di interazione dell'utente con il sistema (i cosiddetti **user journeys**).

Questo flusso di lavorazione iterativo e incrementale, in cui i diversi strumenti sono messi in relazione complementare, sono mostrati in figura 24. L'idea si condivide e si elabora con l'aiuto della **Vision Canvas**, e una volta che la vision è chiara e condivisa si possono creare in parallelo **Business Model Canvas** [7] e **Product Canvas**; a partire da quest'ultimo, poi si può creare il **Product Backlog** del prodotto. E poi, con il feedback degli utenti, è possibile apportare le necessarie modifiche ai vari strumenti.

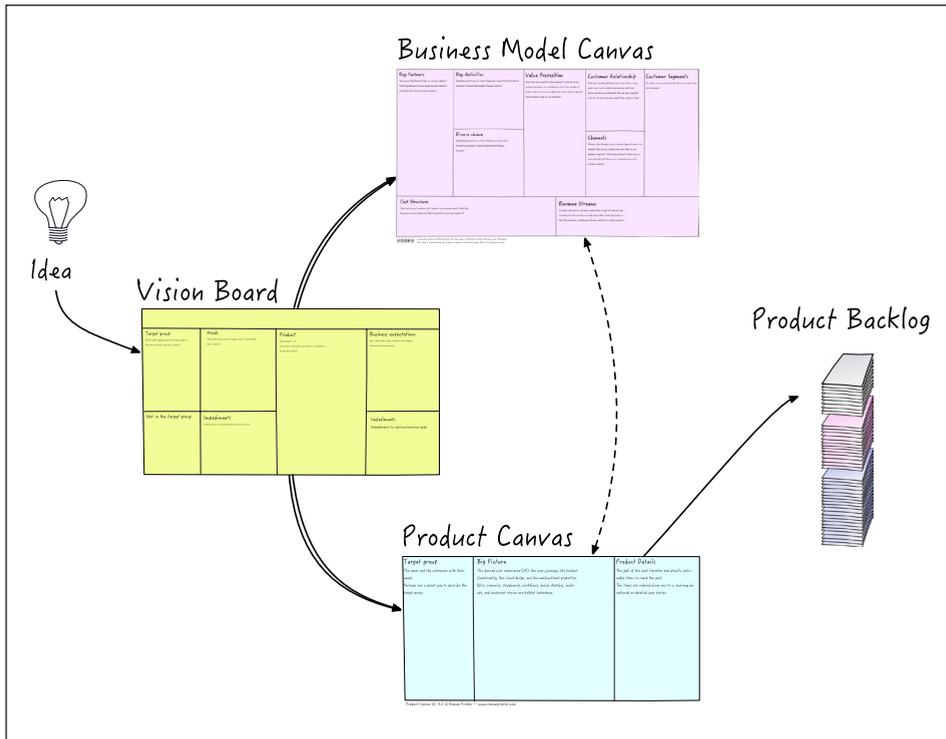


Figura 11.24. Visione di insieme degli strumenti: il processo è iterativo e incrementale e si usano tutti gli strumenti per facilitare il percorso da Idea a backlog.

## Conclusioni

Termina qui la parte dedicata al tema “Dall’idea al prodotto”. Questo argomento spesso viene trattato con titoli differenti, come **Lift Off**, **From Vision to Backlog**, **Inception** e altri ancora, ma i concetti e gli strumenti sono quelli trattati in questi capitoli.

## Riferimenti

[1] Manifesto per lo sviluppo agile di software

<http://agilemanifesto.org/iso/it/>

[2] Pichler Consulting

<http://www.romanpichler.com>

[3] La GO Product Roadmap

<http://www.romanpichler.com/tools/product-roadmap/>

[4] Gli Scenari secondo Pichler

<http://www.romanpichler.com/blog/agile-scenarios-and-storyboards/>

[5] Alcuni strumenti per il design agile di interfacce utente

<http://www.romanpichler.com/blog/agile-user-interface-design/>

[6] Prototyping On Paper, un'app per la prototipazione rapida di app mobile

<https://popapp.in>

[7] Value Proposition Design

<http://www.businessmodelgeneration.com/>