

Introduzione a Scrum



*Corso di Ingegneria del Software
CdL Informatica
Università di Bologna*

Obiettivi della lezione

- Presentazione del metodo agile Scrum
- Ruoli – PO, SM, Dev
- Artefatti: backlog di prodotto e di sprint
- Rituali (riunioni)
- Varianti di Scrum

<https://www.scrum.org/resources/scrum-guide>

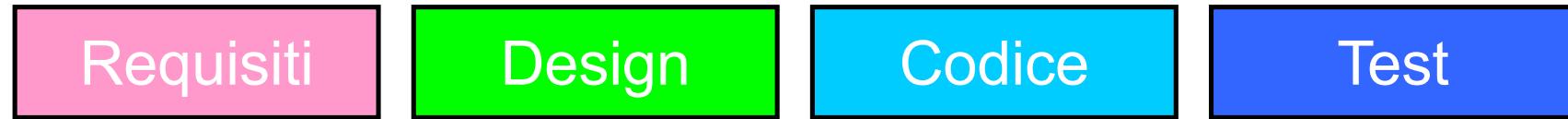
Staffetta o pacchetto di mischia?

“The... ‘relay race’ approach to product development...may conflict with the goals of maximum speed and flexibility. Instead a holistic or ‘rugby’ approach—where a team tries to go the distance as a unit, passing the ball back and forth—may better serve today’s competitive requirements.”

Hirotaka Takeuchi and Ikujiro Nonaka, “The New New Product Development Game”, *Harvard Business Review*, January 1986.

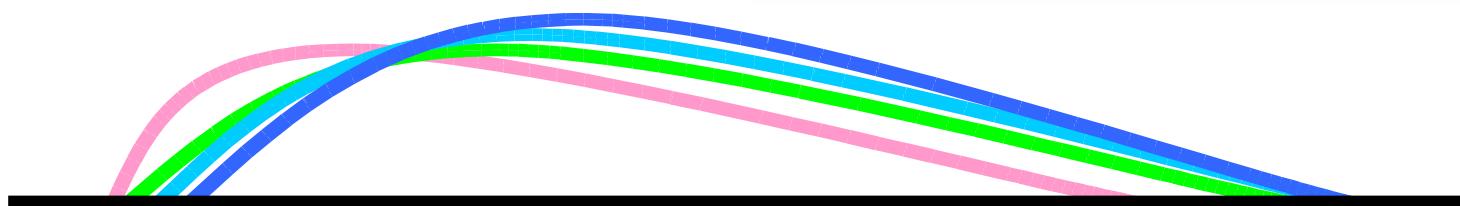


Dalla staffetta al pacchetto di mischia



Non una cosa alla volta

il team scrum fa un po' di tutto in ogni sprint

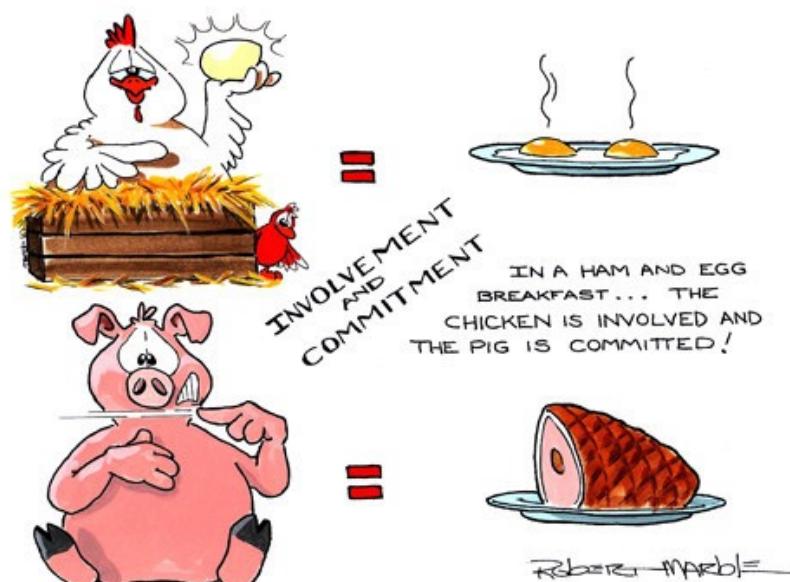


Source: "The New New Product Development Game" by Takeuchi and Nonaka. *Harvard Business Review*, January 1986.

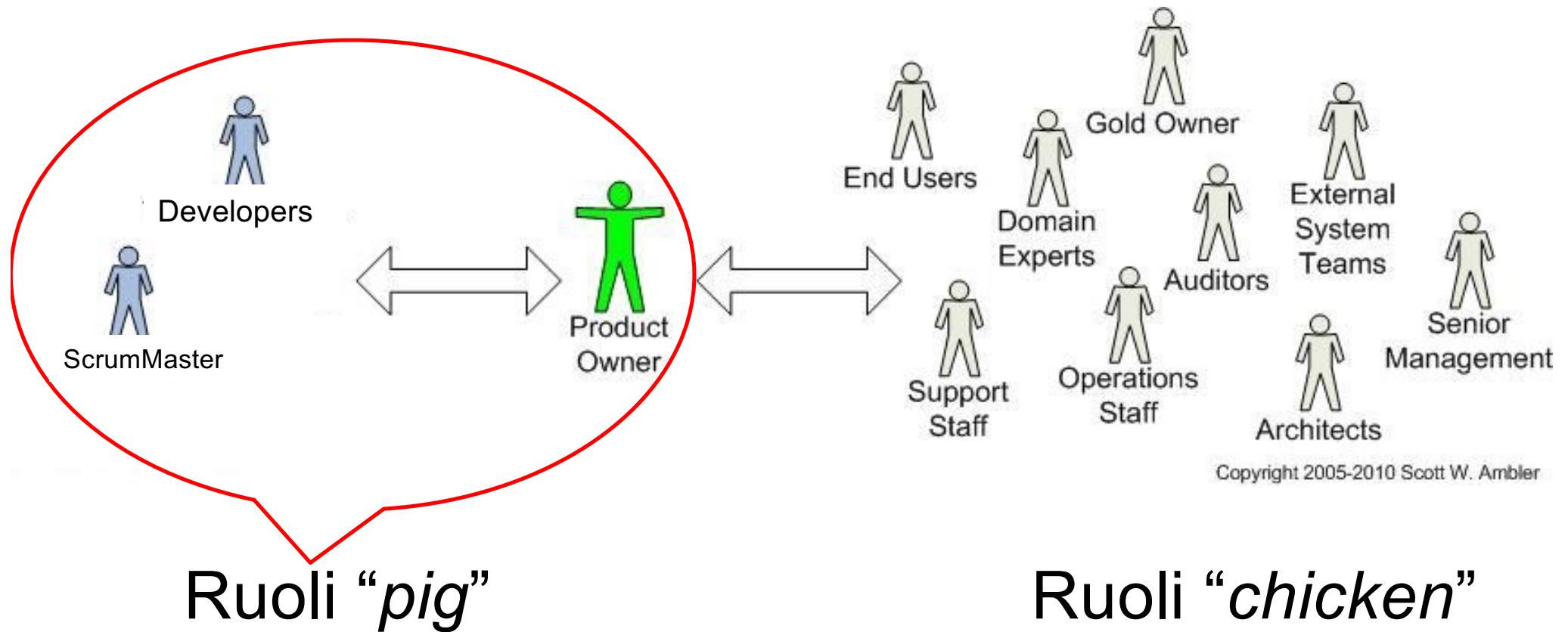
Scrum in 100 parole

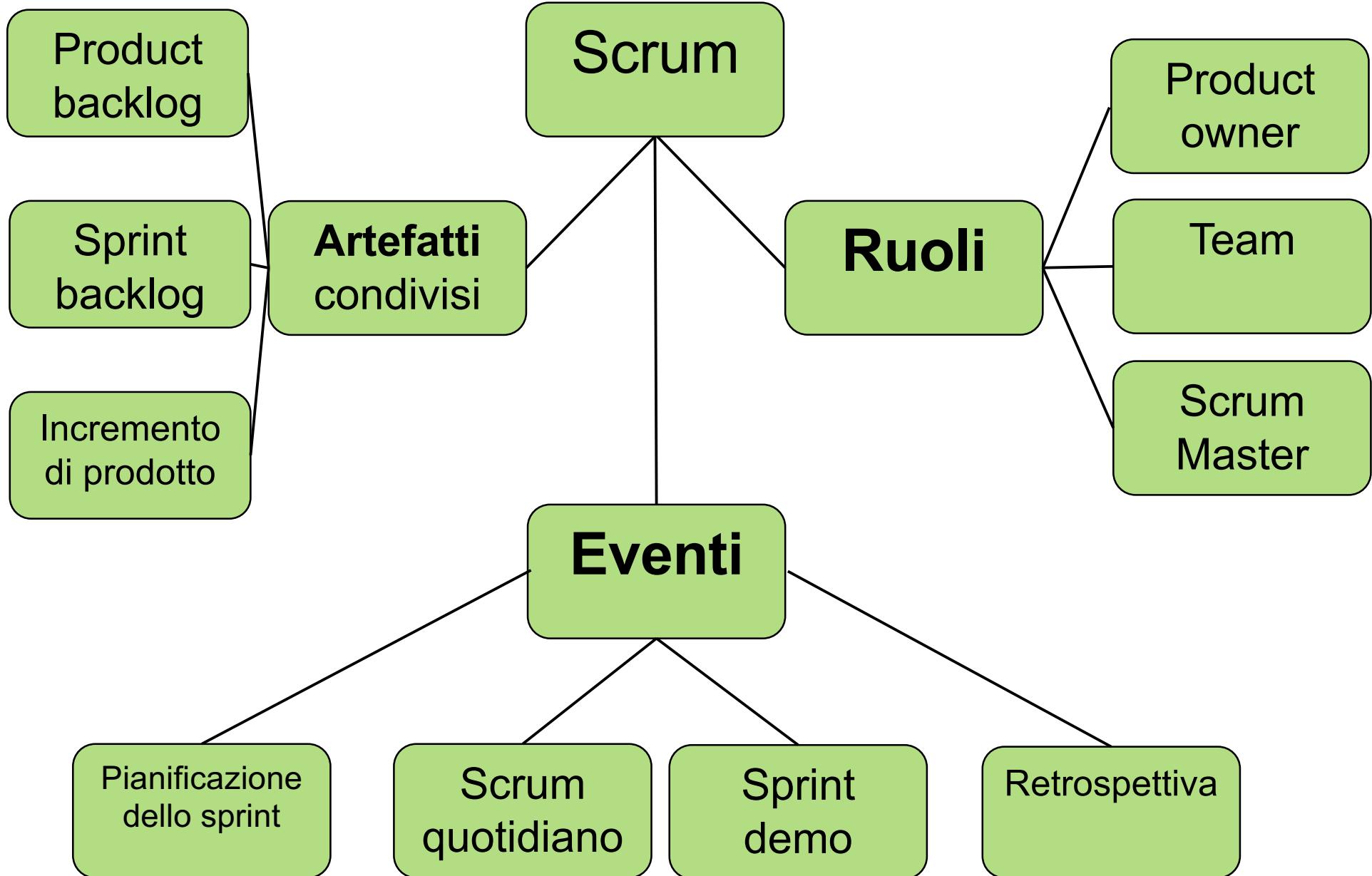
- Scrum è un modello di processo per produrre software ottenendo il massimo valore utile nel minor tempo
- Il cliente definisce le funzioni da realizzare e loro priorità
- Il team di sviluppo decide quotidianamente il modo migliore di produrre le funzioni di più alta priorità.
- Alla fine di ogni sprint nasce una nuova versione che viene esaminata dal cliente per decidere se continuare lo sviluppo con un altro sprint e/o produrre un rilascio

Galline e maiali

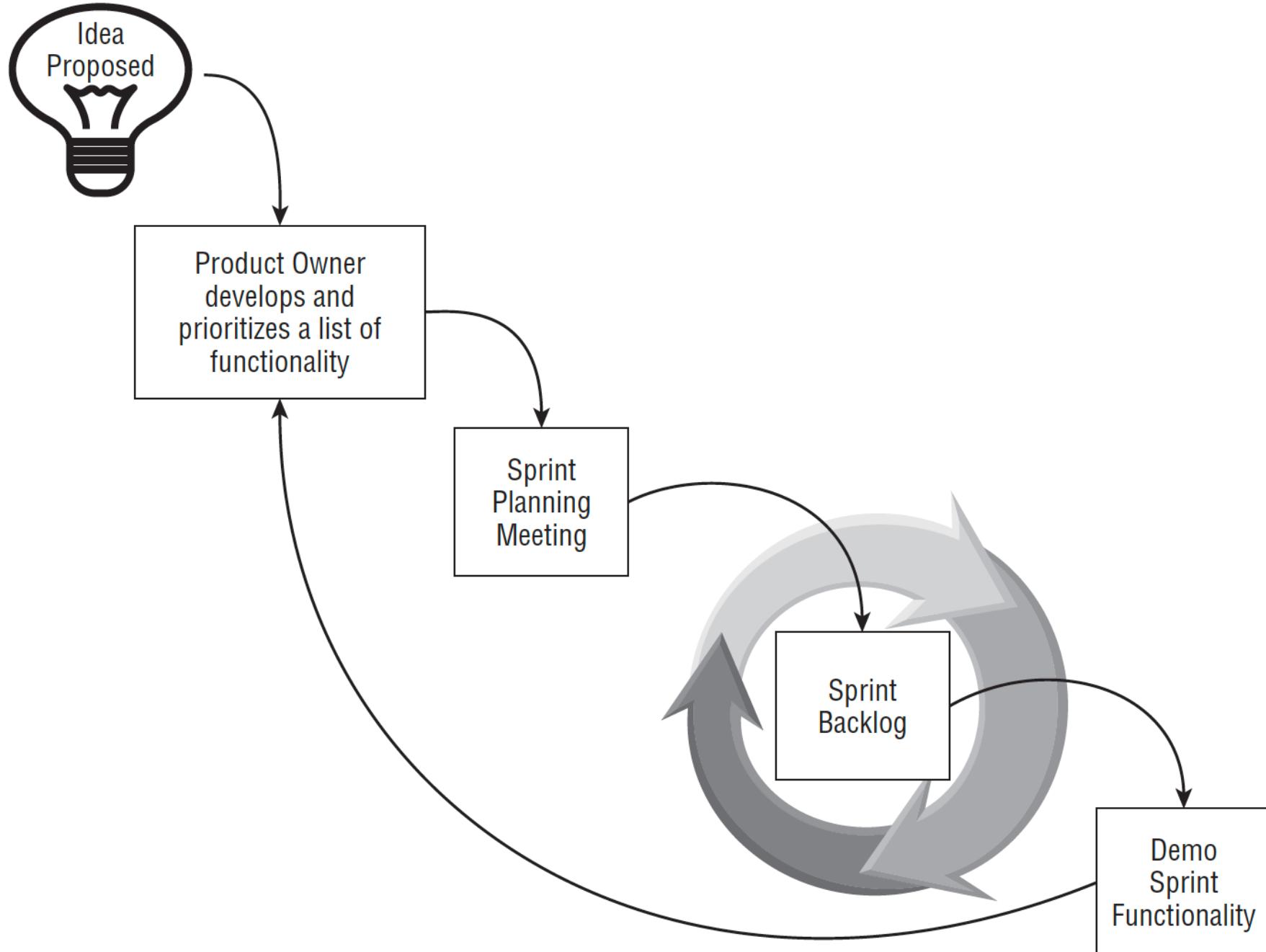


Scrum: ruoli pig e ruoli chicken

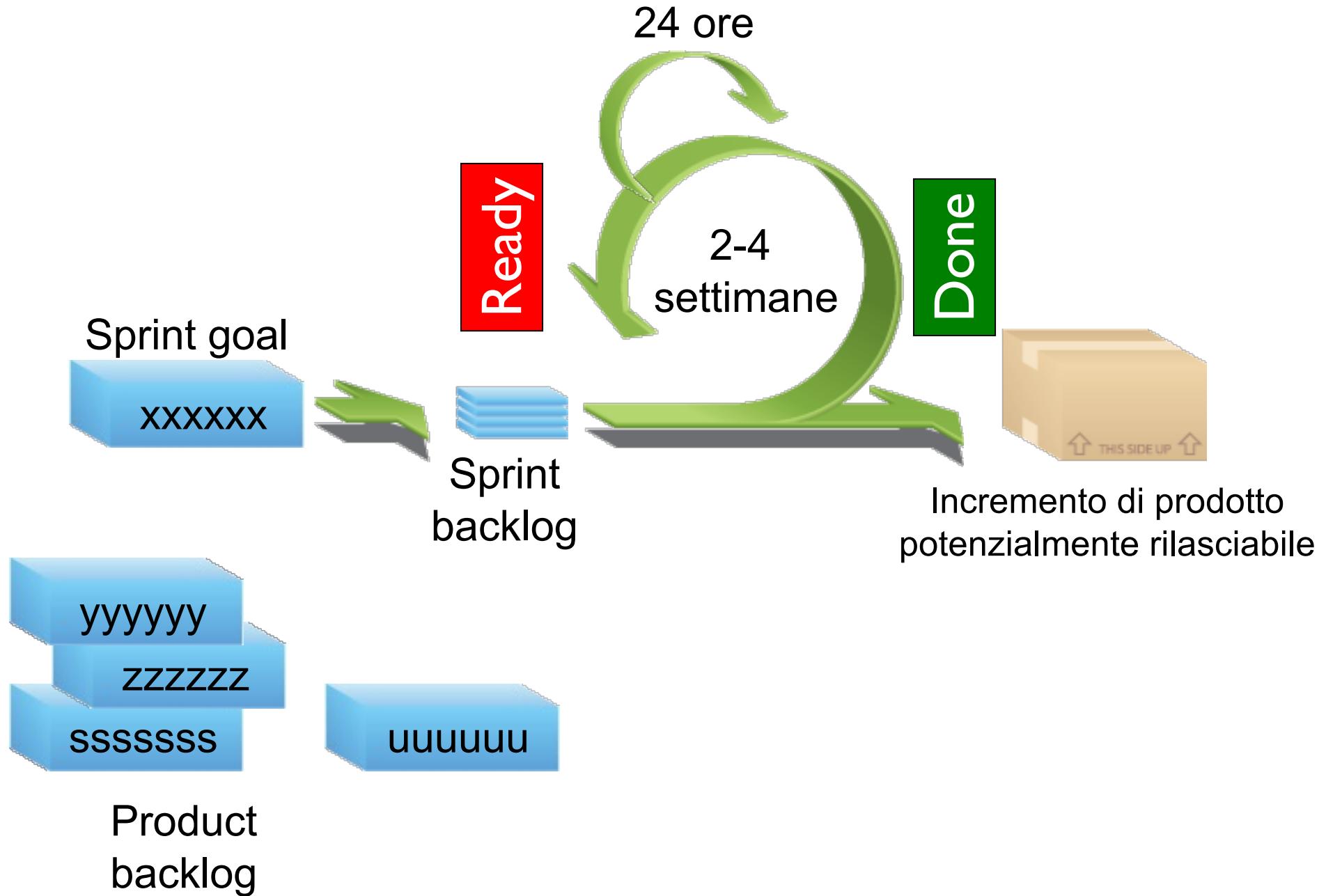




Il ciclo di base in Scrum



Un processo Scrum si compone di **sprint**



L'iterazione principale: lo sprint

- Un processo Scrum è una serie di sprint (iterazioni)
- Durata di ogni sprint da 2 a 4 settimane (a scelta del team+PO)
- Durata di ogni sprint: costante (migliora il ritmo del team)
- Ogni sprint include: design, codifica e test
- Ogni sprint estrae funzioni “**Ready**” dal product backlog e aggiunge codice “**Done**” al prodotto, da mostrare al cliente

Scrum: ruoli eventi artefatti

Ruoli

- Product owner
- ScrumMaster
- Team

Eventi

- Sprint planning
- Sprint review
- Sprint retrospective
- Daily scrum meeting

Artefatti

- Product backlog
- Sprint backlog
- Burndown charts

Scrum: i ruoli

Ruoli

- Product owner
- ScrumMaster
- Team

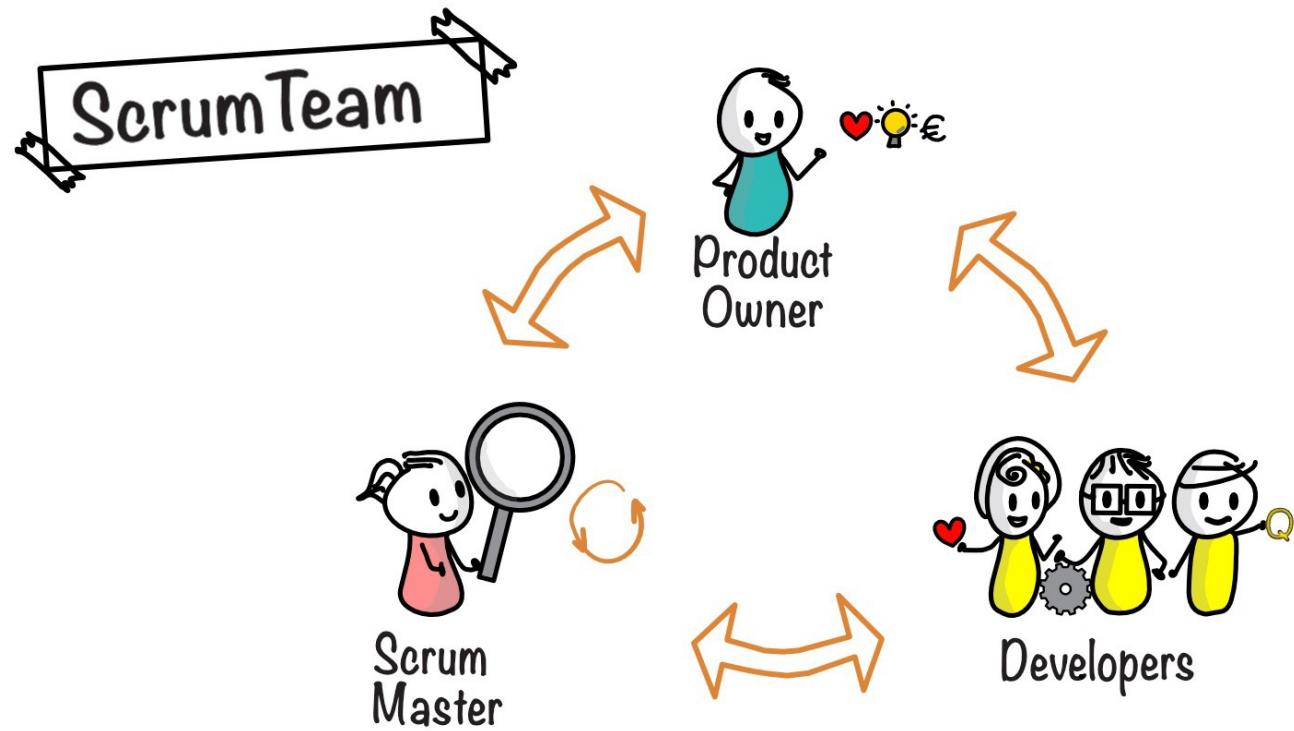
Rituali

- Sprint planning
- Sprint review
- Sprint retrospective
- Daily scrum meeting

Artefatti

- Product backlog
- Sprint backlog
- Burndown charts

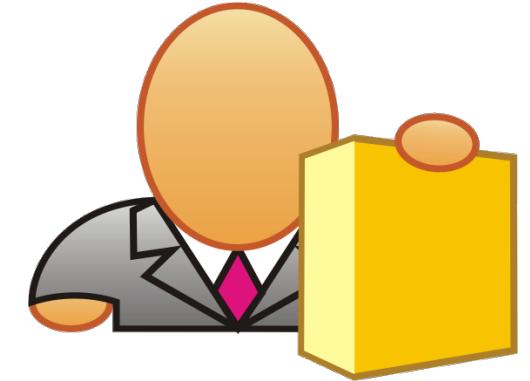
I ruoli principali in Scrum



I ruoli in Scrum

- **Product Owner (PO)**: rappresenta gli stakeholder (la voce del cliente), scrive il *product backlog* in forma di *user stories*; è responsabile della definizione di “Fatto”.
- **Team di sviluppo**: 3-9 membri con diverse abilità, collettivamente responsabili della consegna di un PSI (Potentially Shippable Increment)
- **Scrum Master (SM)**: facilita la corretta esecuzione del processo, elimina gli ostacoli
 - Product Owner e Scrum Master debbono essere persone diverse; possono collaborare allo sviluppo
 - SM non ha responsabilità di gestione del personale o del progetto in senso “tradizionale” – es. non dà premi o sanzioni

Product Owner (PO)



- Responsabile del valore del prodotto (Return On Investment - ROI)
- Definisce le feature (funzioni) del prodotto
- Decide i rilasci: date e contenuti
- Mette in priorità le feature rispetto al valore di mercato
- Per ogni iterazione rivede lista delle feature e loro priorità ove necessario
- Accetta o rifiuta i risultati dello sviluppo mediante la definizione di “Fatto”

Il Product Owner in Essence

«Possiede» il prodotto

Gestisce la visione del
prodotto e la sua
evoluzione

Mostra il prodotto agli
stakeholder

Acquisisce l'accettazione
del prodotto

Product Ownership Essentials

Own, evolve and communicate the
vision, and guide the evolution of the
product to achieve the vision.



Product
Ownership



Build
Stakeholder
Network



Stakeholder
Network



Evolve the
Product Vision



Product Vision



Demonstrate
the Product



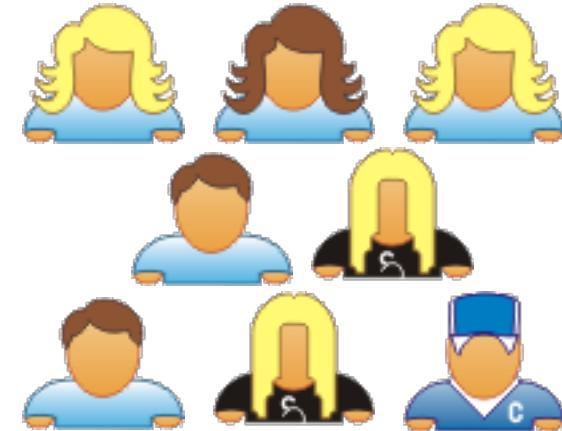
Achieve
Acceptance



Resources

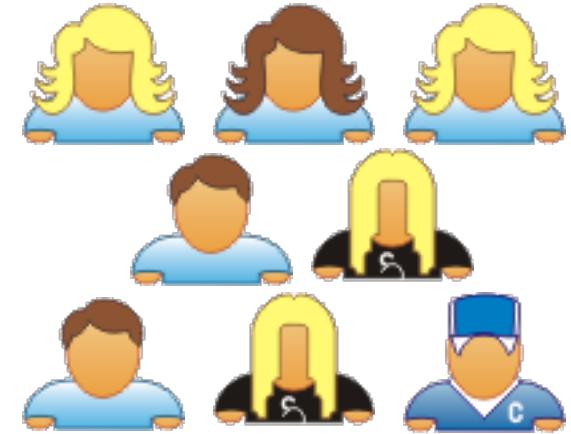


Il team di sviluppo



- Auto-organizzante
- Modifiche al team solo in sprint diversi: è stabile per tutto lo sprint
- Tutti presenti nello stesso spazio di lavoro
- Include uno Scrum Master come coach e mentore

I membri del team



- 5-9 persone
- Team piccolo comunica meglio
- Varie specialità:
 - Programmatori, testatori, user experience designers, database administrator, ecc.
- Impegno full-time
 - Con eccezioni (e.g., database administrator, costoso)

Caratteristiche del team Scrum

Il team include sviluppatori capaci di produrre un incremento del prodotto “Fatto” alla fine di ogni sprint.

Il team si autogestisce ed ha le seguenti caratteristiche:

- I membri si auto-organizzano; nessuno, nemmeno lo Scrum Master, può dire al Team come trasformare il Product Backlog in incrementi di funzionalità potenzialmente rilasciabili;
- I team sono cross-funzionali, cioè includono tutte le abilità necessarie a creare un incremento di prodotto;
- I membri del Team sono tutti Sviluppatori alla pari, senza nessuna eccezione;
- Il Team è unitario e non include sottogruppi, senza nessuna eccezione per attività o domini particolari;
- I membri del Team possono avere qualche specializzazione personale, ma l'intero Team resta responsabile dello sviluppo.

Il Team in Essence

Il team si autoorganizza

Include un facilitatore
(detto Scrum Master)

Sceglie una modalità di
lavoro

Collabora e condivide la
conoscenza

Tiene un passo sostenibile

Agile Teaming Essentials

A self-organizing team maximizes its performance by using a highly-collaborative teaming approach.



ScrumMaster

- Serve il team, non lo dirige
- Responsabile dei valori e pratiche agili
- È il “Process Owner”: coordina i meeting e rimuove gli impedimenti al processo Scrum
- Assicura il benessere del team
- Supporta la cooperazione di ruoli e funzioni
- Protegge il team da interferenze esterne
- Non è detto che sia un programmatore



I ruoli Scrum: le responsabilità

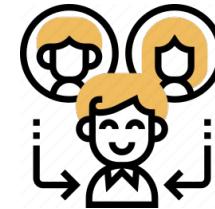


Product owner



Costruire
il prodotto
bene

Sviluppatori



Evitare gli
inciampi e
assicurare
la velocità

Scrum Master

Scrum: gli eventi

Ruoli

- Product owner
- ScrumMaster
- Team

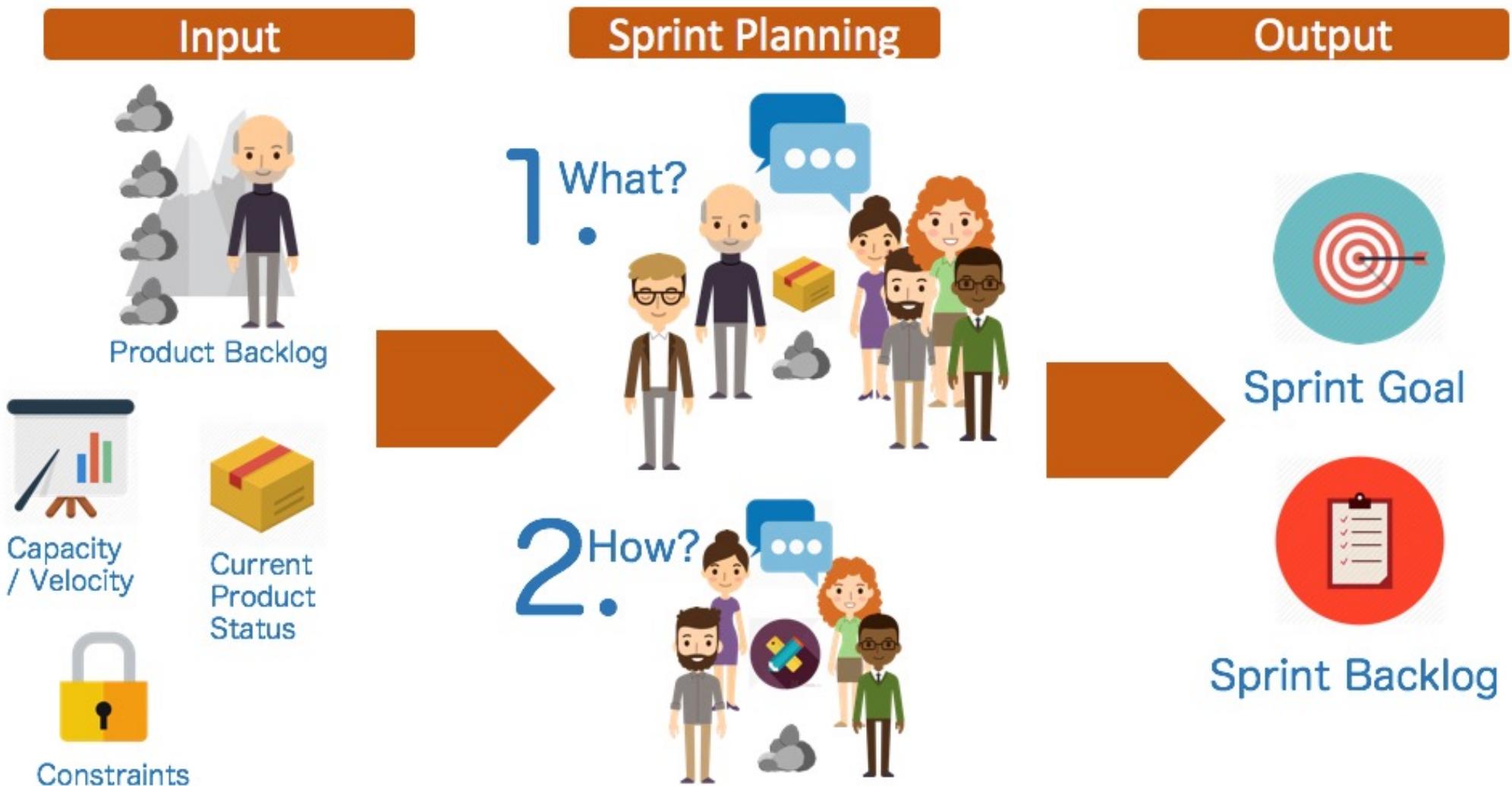
Eventi

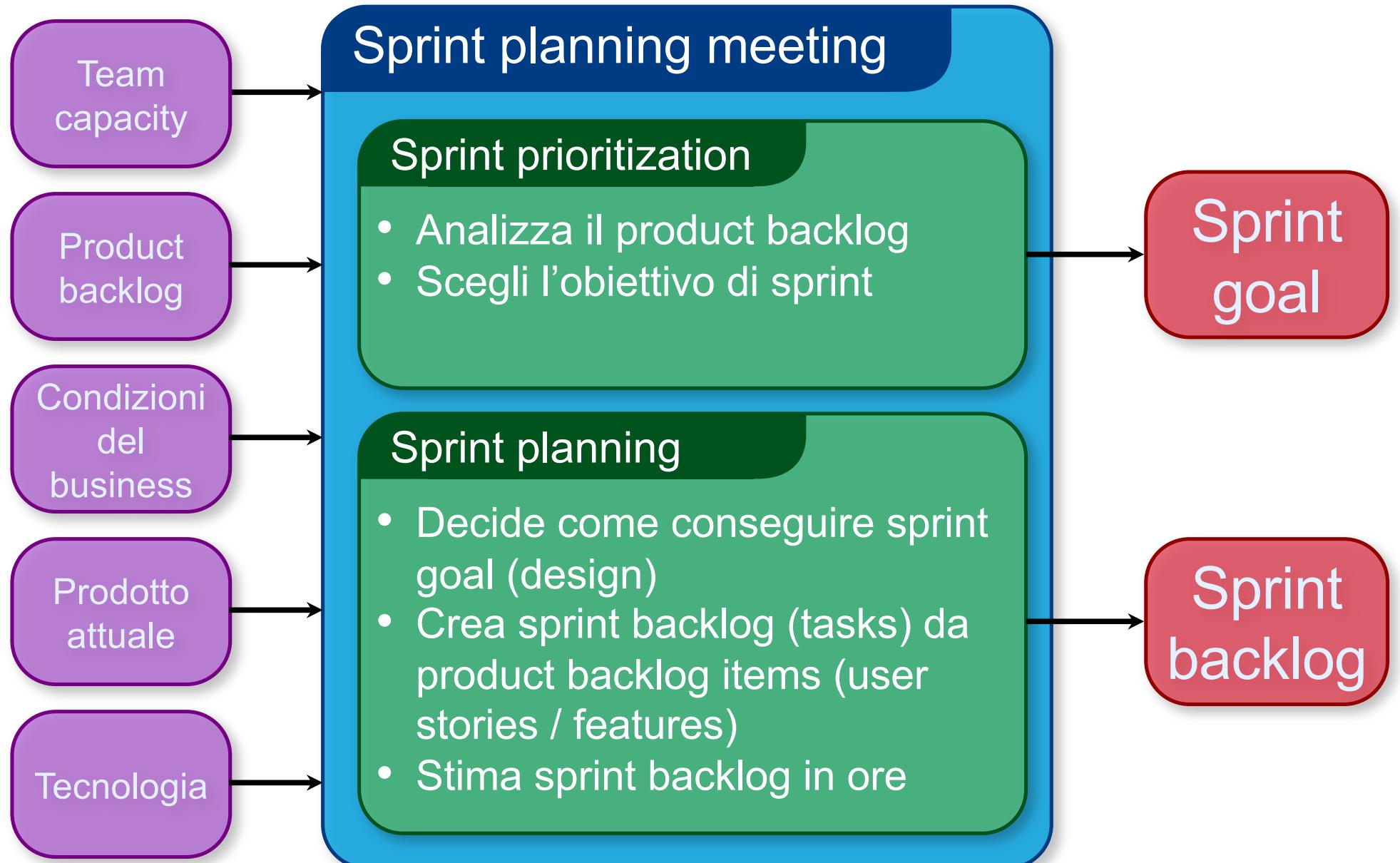
- Sprint planning
- Sprint review
- Sprint retrospective
- Daily scrum meeting

Artefatti

- Product backlog
- Sprint backlog
- Burndown charts

Il meeting di pianificazione dello sprint





The scrum board



La definizione di Pronto (definition of Ready)

Una storia è Ready quando:

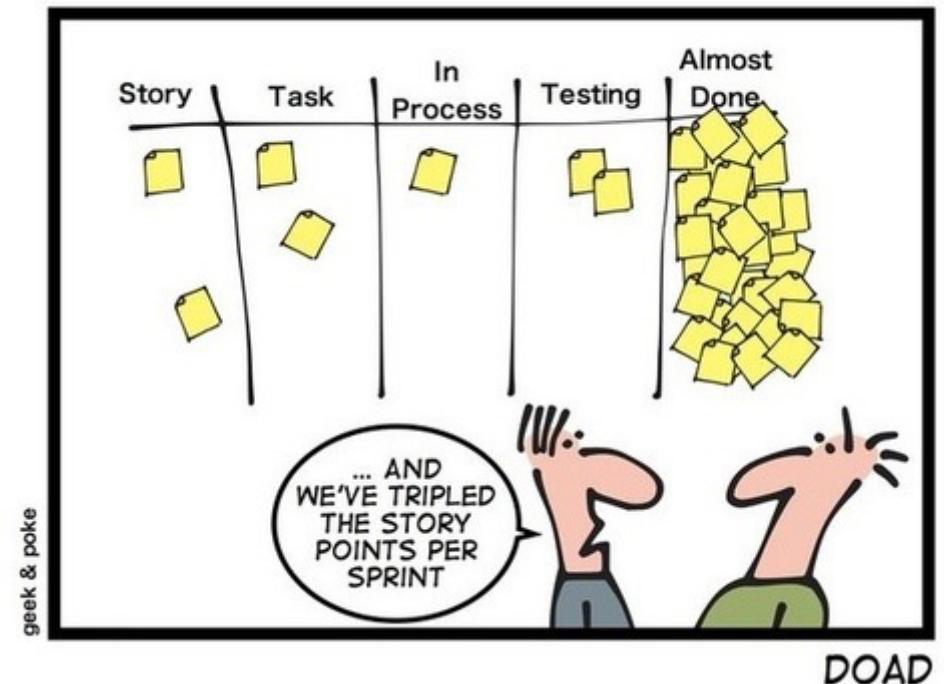
- Tutti i membri del team la capiscono
- È dettagliata abbastanza da poter essere poi testata
- Il PO deve comprenderne il valore per poterla ordinare in priorità con le altre
- Il Team dev'essere in grado di farne una stima in modo da completarla entro uno sprint
- La storia è indipendente dalle altre: il Team può lavorarla avendone il pieno controllo

La definizione di Fatto (Definition of Done)

La Definition of Done descrive lo stato dell'Increment quando questo soddisfa le metriche di qualità richieste per il prodotto.

La Definition of Done crea trasparenza fornendo a tutti una comprensione condivisa di quale lavoro è stato completato come parte dell'Increment.

Se un elemento non soddisfa la Definition of Done, non può essere rilasciato e nemmeno presentato durante la Sprint Review: ritorna nel Product Backlog



Esempio di definizione di “Fatto”

La lavorazione di una funzione che realizza una user story non è finita a meno che non soddisfi il PO: la definizione di “Fatto” è concordata dal Team col PO all'inizio del processo

Una possibile definizione di Fatto:

- i. Codifica della funzionalità richiesta: completata
- ii. Test di unità: scritti ed effettuati
- iii. Test di integrazione: superato
- iv. Test prestazionale: superato
- v. Documentazione (minimale): scritta
- vi. Approvata: dal PO

Pianificazione dello sprint

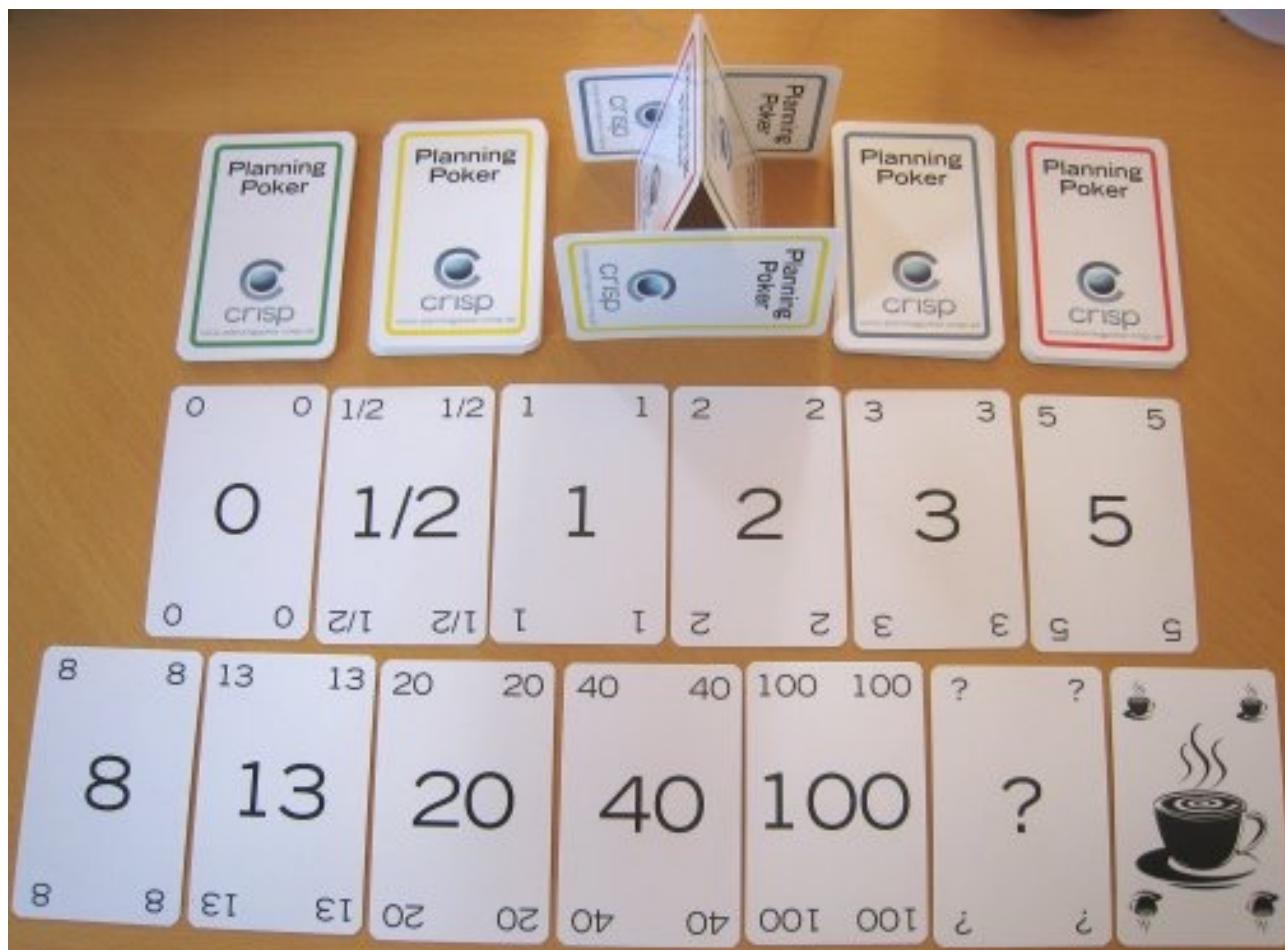
- Il team sceglie dal product backlog gli elementi che verranno certamente realizzati nello sprint
- Studio dell'architettura di alto livello
- Si crea lo sprint backlog: i compiti da fare
 - Identificazione e stima di ciascun compito (1-16 hours)
 - Uso di Planning Poker

As a vacation
planner, I want to
see photos of the
hotels.



Code the middle tier (8 hours)
Code the user interface (4)
Write test fixtures (4)
Code the foo class (6)
Update performance tests (4)

Planning poker



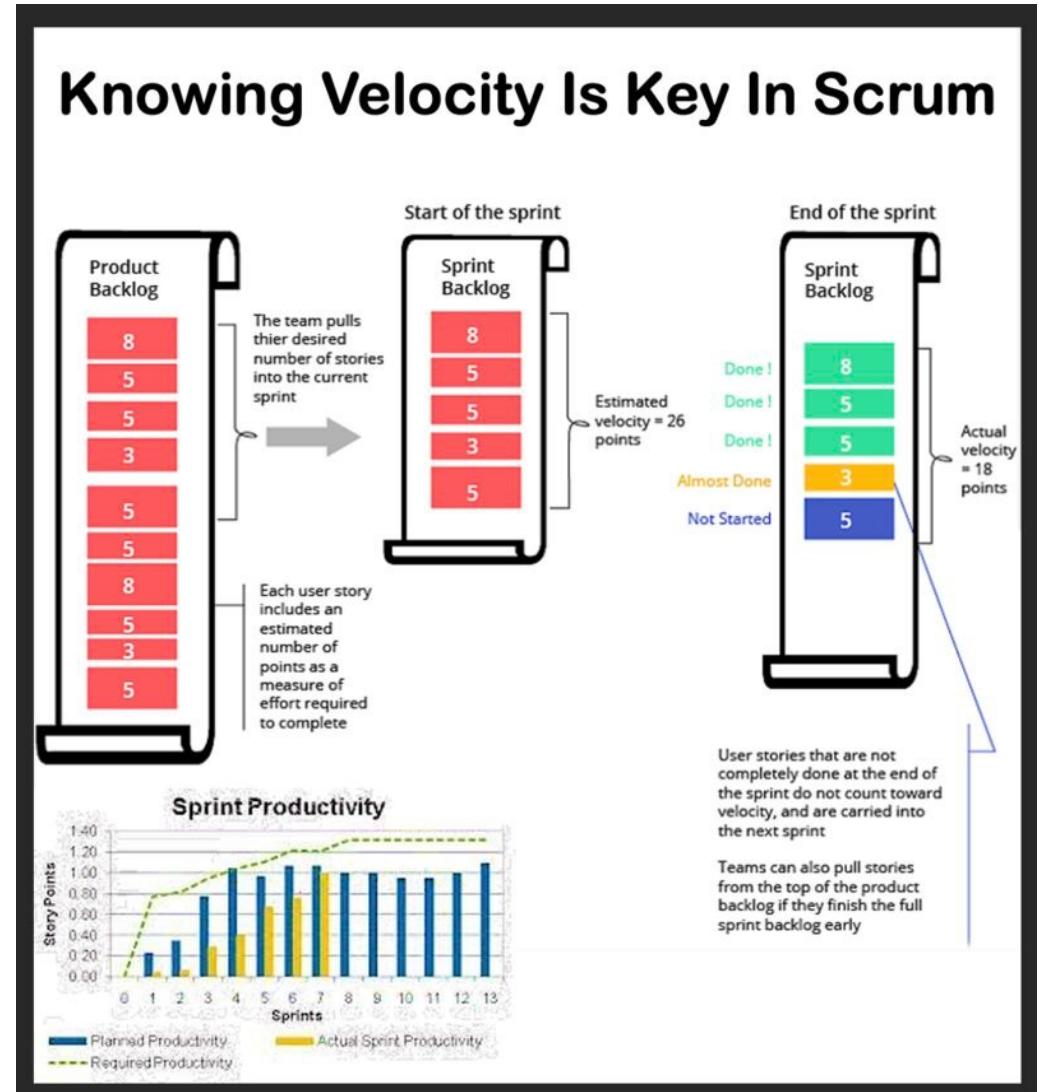
Pianificazione dello sprint

La “velocità” è la quantità di story point che il team ha completato durante uno sprint

La velocità si calcola alla fine di uno sprint ed è utile per prevedere (stimare) quanto lavoro si riuscirà a fare nello sprint successivo

La velocità si può misurare

- Per ciascun singolo task
- Per ciascuno sprint



Scrum quotidiano

- Modalità
 - Guardando il board
 - 15-minuti max
 - In piedi
 - Chiunque può assistere
 - Possono parlare solo i pig: membri del team, ScrumMaster, Product Owner
 - Perché in piedi: evitare durata troppo lunga



Daily scrum: Tre domande a tutti

1

Che hai fatto ieri?

2

Che farai oggi?

3

Prevedi problemi?

Le risposte sono promesse ai compagni di team

Il PO nel daily scrum

Il PO partecipa al meeting quotidiano, ma

- Non importa se il PO è assente (il meeting inizia comunque)
- Il PO non può inserire nuove US (questo si fa solo all'inizio dello sprint)
- Il PO non pretende di avere chiari gli stati di avanzamento: gli sviluppatori si autogestiscono
- Il PO partecipa per dare suggerimenti costruttivi

(Scherzo)

How to make meetings shorter:



Sprint review: la demo del prodotto

- Il team presenta al PO ciò che ha ottenuto con lo sprint: occorre mostrare come il prodotto è “cresciuto”
- È una demo delle nuove funzioni
- Serve per verificare lo stato del product backlog
- Riunione informale
 - 2 h di preparazione, niente slide
- Partecipa tutto il team
- Invitare tutti



Sprint retrospective

- La retrospettiva riguarda il processo (non il prodotto): come sta andando lo sviluppo?
- Occorre analizzare periodicamente cosa va e cosa non va in ciascuno sprint
- Quando: alla fine di ciascuno sprint, dopo la review
- Partecipano tutti:
 - ScrumMaster (coordinatore della retrospettiva)
 - Product Owner
 - Team
 - Forse commitenti, utenti e altri stakeholders

Retrospettiva: Start / Stop / Continue

Il team discute cosa vorrebbe fare:

Start doing

Stop doing

Ci sono vari modi alternativi di definire una retrospettiva

Continue doing

Lo scopo della retrospettiva

- Esaminare come è andato l'ultimo sprint riguardo a persone, relazioni, processi e strumenti;
- Rivedere i principali driver architetturali rispetto alle attese degli stakeholder
- Identificare e ordinare gli elementi principali che sono andati bene e le migliorie potenziali;
- Creare un piano per attuare i miglioramenti al modo di lavorare del Team

Cancellare uno sprint

- Il Product Owner può cancellare uno sprint durante il suo svolgimento se l'obiettivo dello sprint diventa obsoleto
 - per es. se sono cambiate le condizioni di mercato o se l'organizzazione subisce una modifica
- Di solito non ha senso cancellare uno sprint

Scrum: gli artefatti

Ruoli

- Product owner
- ScrumMaster
- Team

Eventi

- Sprint planning
- Sprint review
- Sprint retrospective
- Daily scrum meeting

Artefatti

- Product backlog
- Sprint backlog
- Burndown charts

Product backlog



Questo è il
product backlog

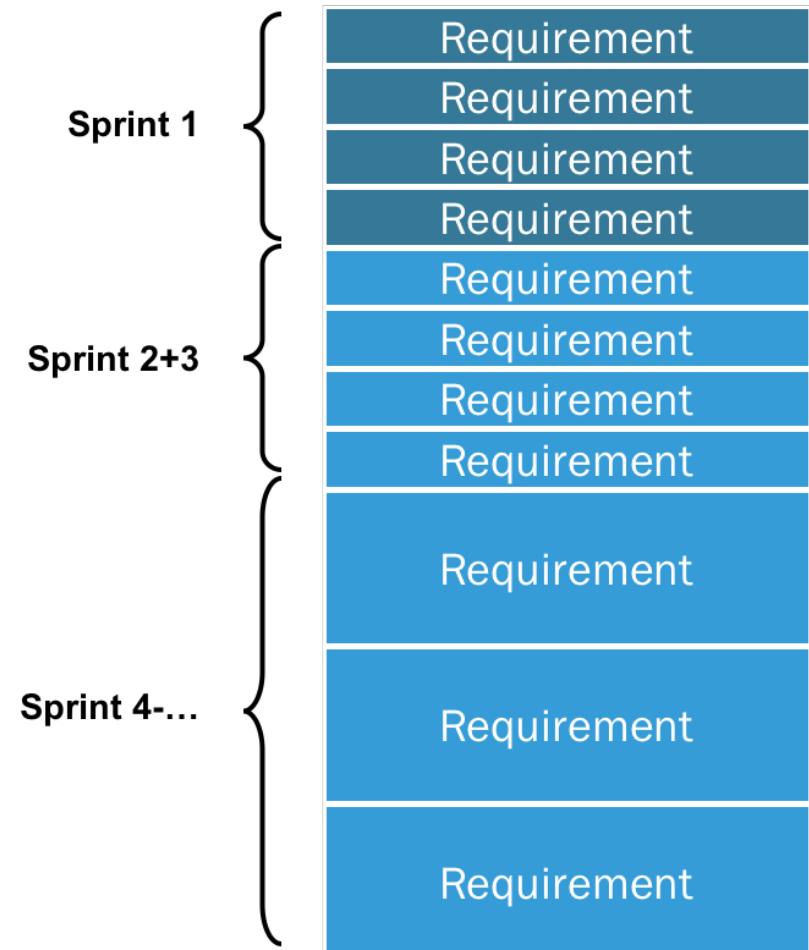
- Product backlog = Lista dei requisiti in forma di user stories
- Definita in modo tale che ciascun elemento abbia valore per gli utenti o i committenti del prodotto
- Messa in priorità dal Product Owner
- Priorità ridefinite all'inizio di ogni sprint

Il backlog di prodotto è una lista di storie ordinate dal PO e stimate dal team

ToDo List			
Story	Estimation	Priority	
As a user I want to be able to reset my password	1	1	1
As a user I want to edit items	3	2	2
As a user I want to export data	2	3	3
As an administrator I want to define KPI's for my sales team	4	4	4
As a user I want to view my data on mobile	5	5	5
As an administrator I want to send alerts when new leads come in	2	6	6
As a user I want to create a report of my data	5	7	7
As a user I want to update my reminder settings when a date is added	3	8	8
As a user I want filtering enhancements	4	9	9
As an administrator I want to configure views of data	5	10	10
Total	34		

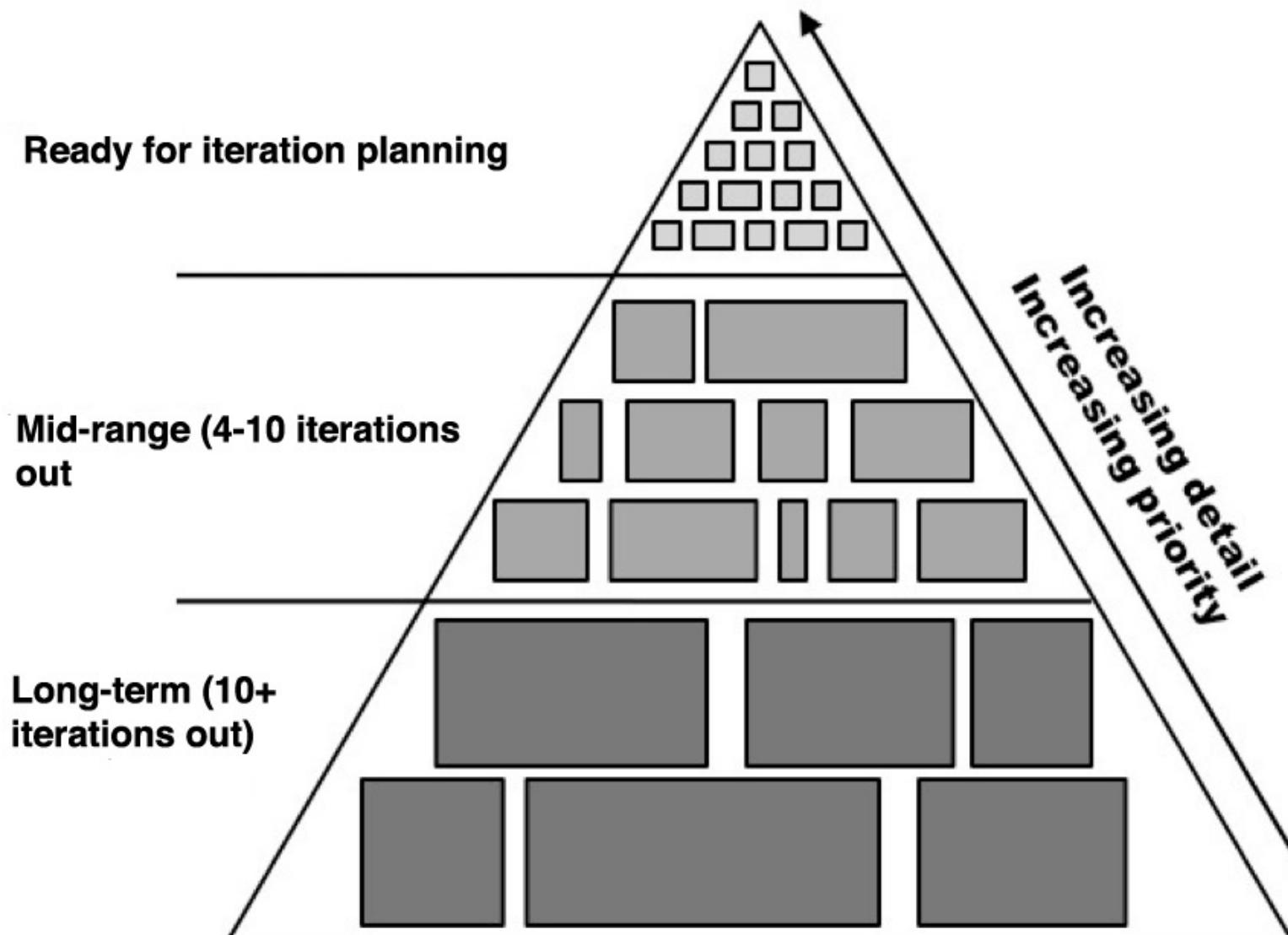
Il backlog di prodotto è ordinato

- L'ordinamento (priorità) delle user story è compito del PO
- La suddivisione di quali storie verranno realizzate nel prossimo sprint è compito del team, sulla base delle stime di sviluppo
- (le user story sono requisiti)

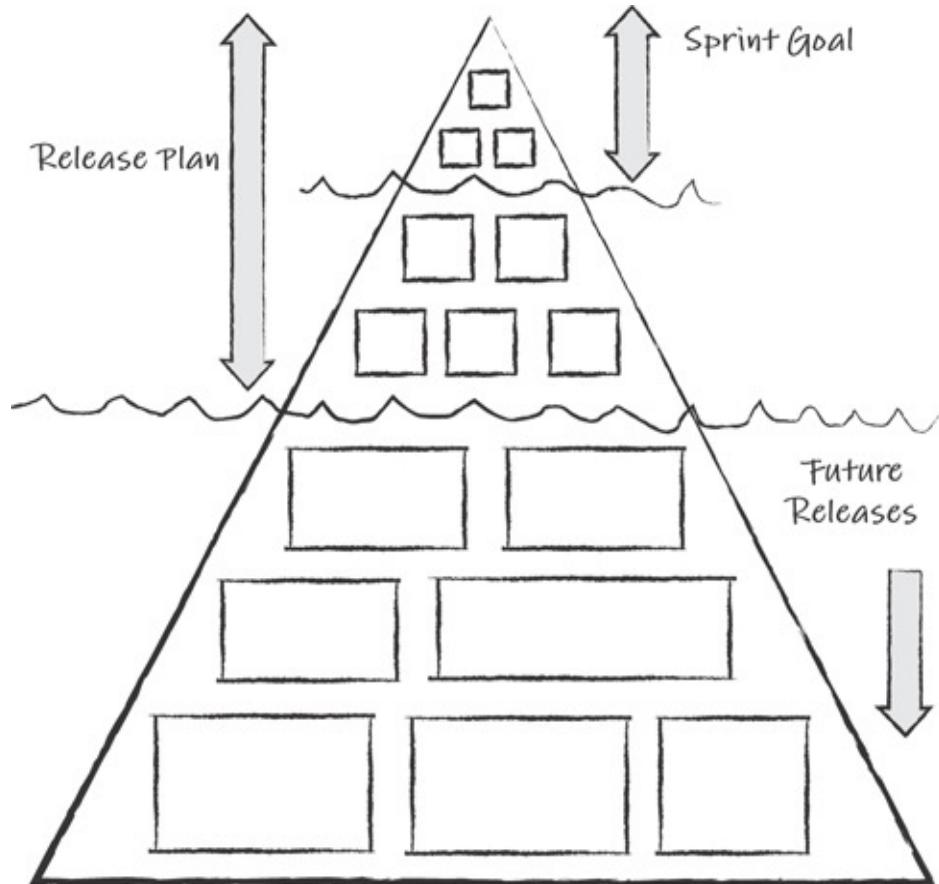


La struttura del product backlog

Break down stories as they move up the backlog

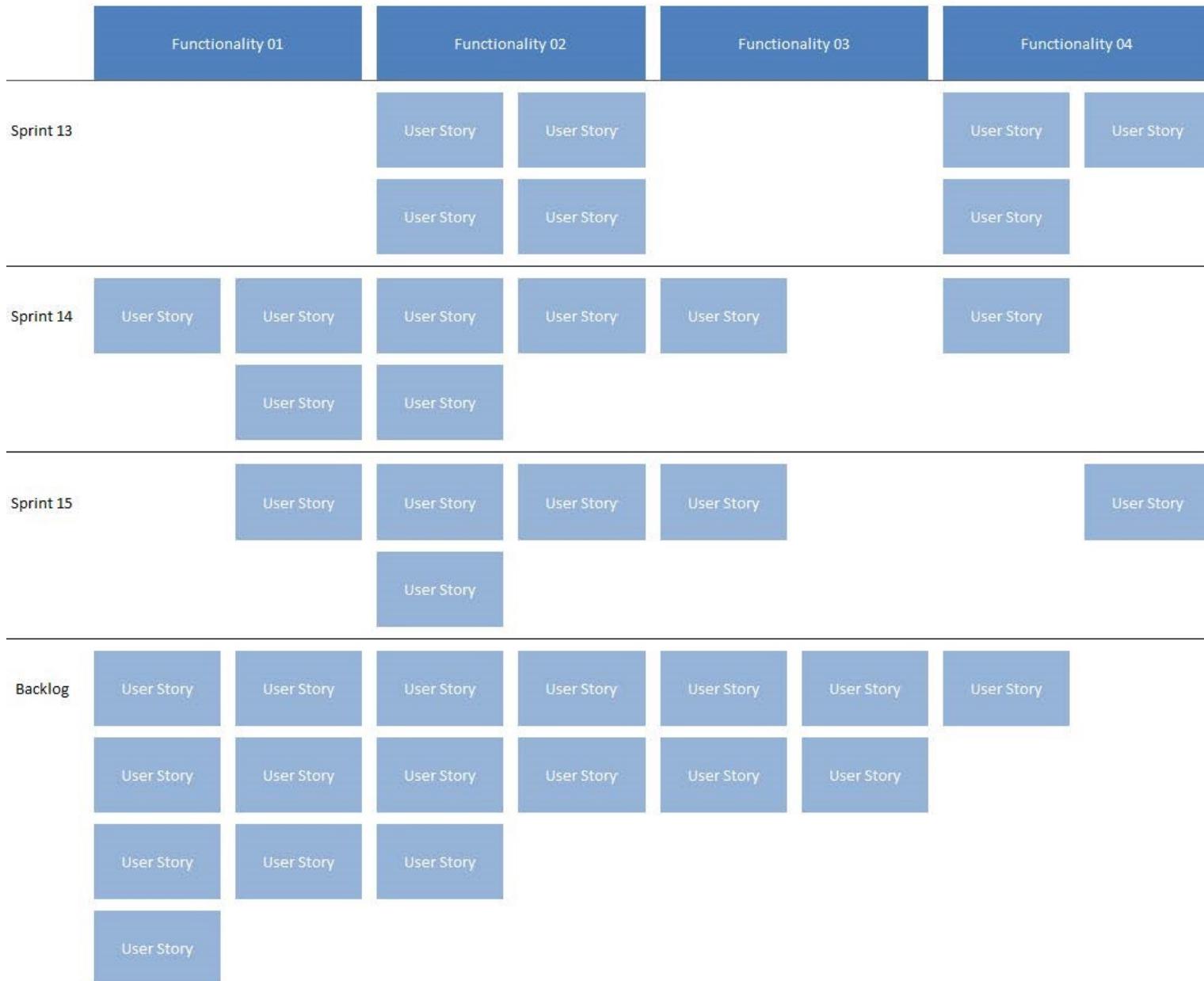


Il product backlog è un iceberg



- La cima dell'iceberg sono le storie pronte per essere realizzate
- Le storie sotto la cima sono previste nel piano dei rilasci, anche se sono ancora poco chiare
- Sotto la superficie ci sono i rilasci futuri, ancora non concordati o nemmeno immaginati

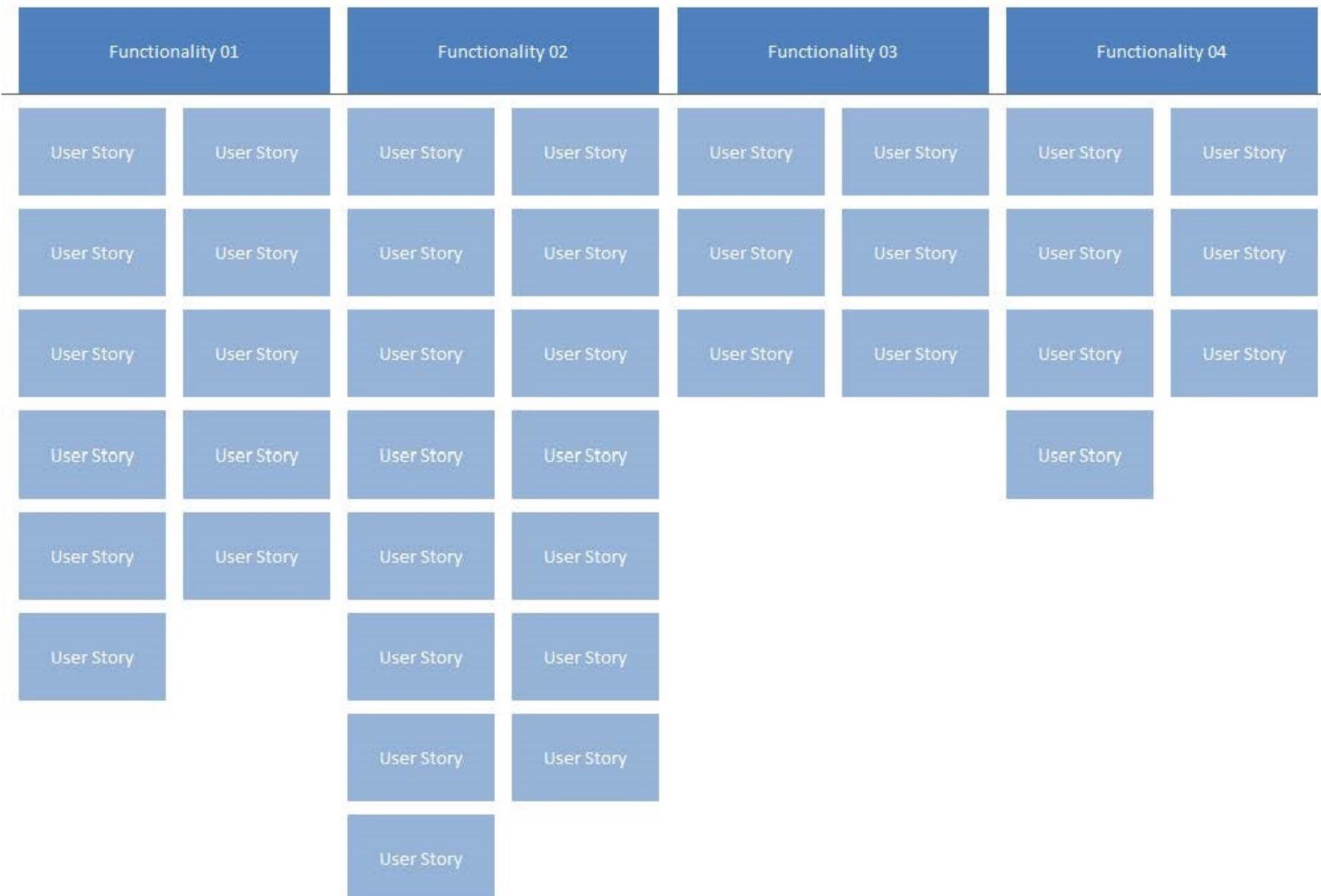
Mettere in priorità le user story



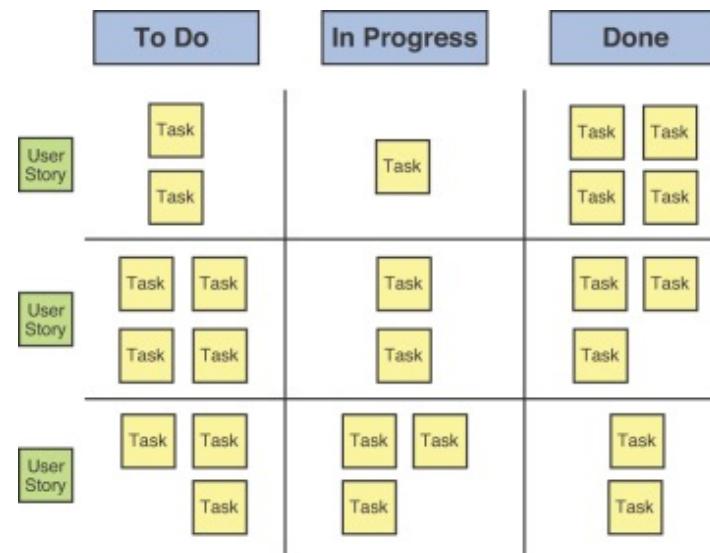
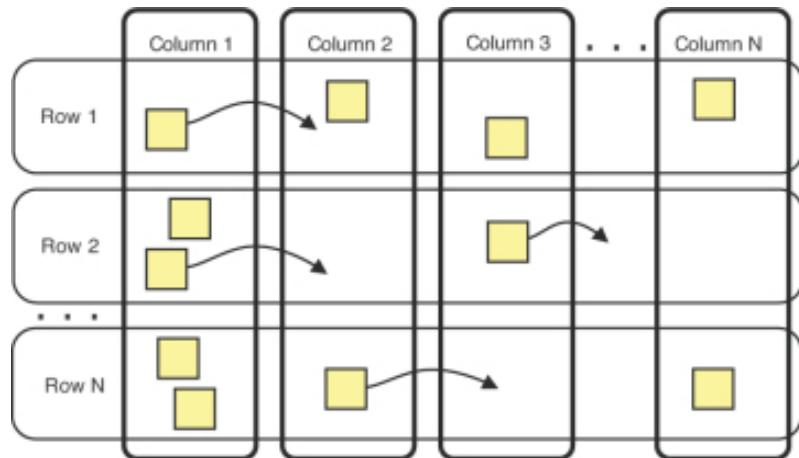
User story map: come affettare il product backlog



Mappare le user story sulle feature



Come si usa il backlog



Obiettivo dello sprint (sprint goal)

Breve descrizione del lavoro da fare durante lo sprint. Esempi:

Database Application

Make the application run on SQL Server in addition to Oracle.

Life Sciences

Support features necessary for population genetics studies.

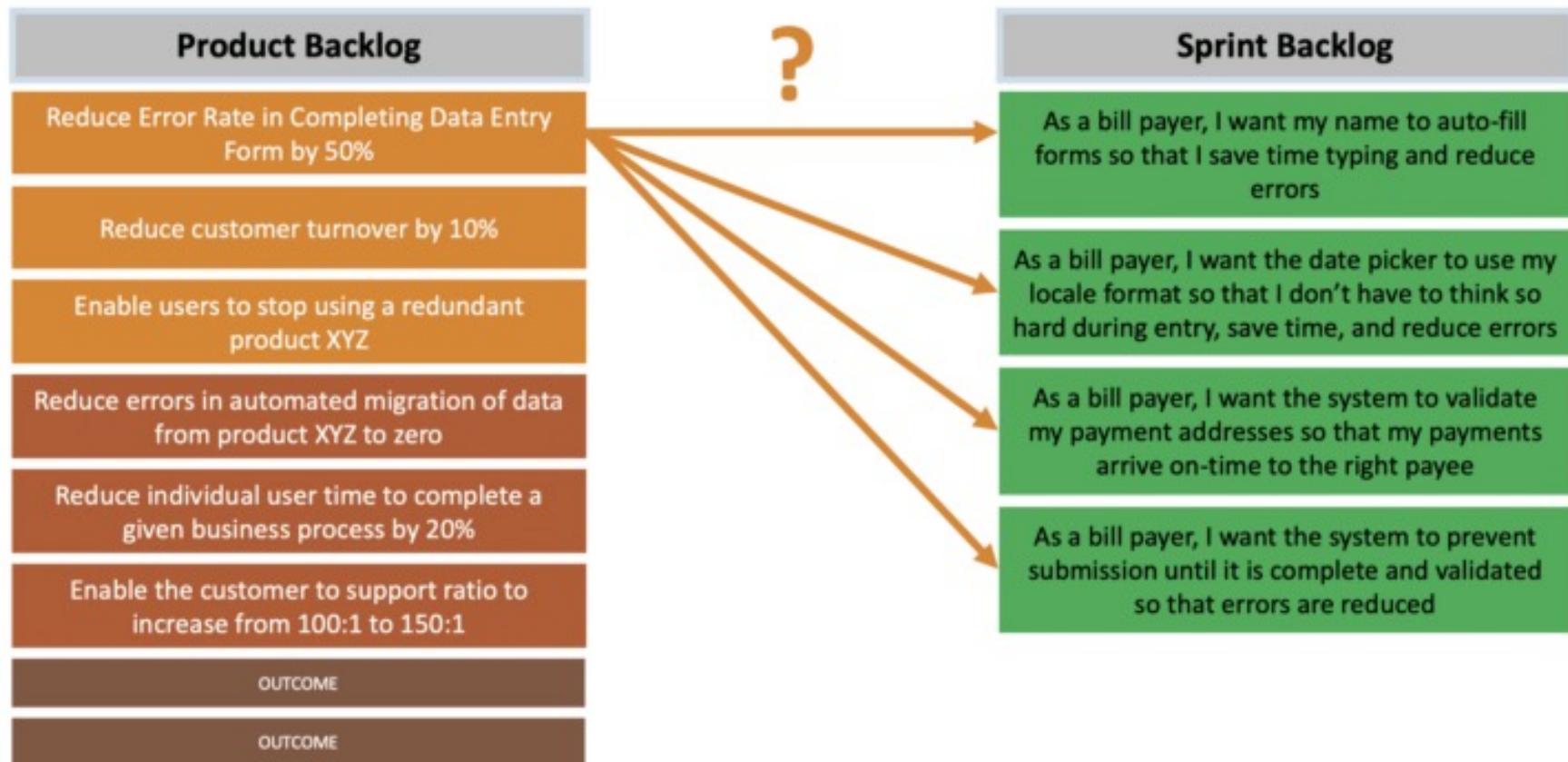
Financial services

Support more technical indicators than company ABC with real-time, streaming data.

Gestione dello sprint backlog

- I membri del team prenotano lavoro da fare su scelta personale
- Il lavoro non viene assegnato, ma richiesto su base “volontaria”
- La stima del lavoro da fare in termini di effort viene aggiornata quotidianamente

Dal backlog di prodotto a quello di sprint



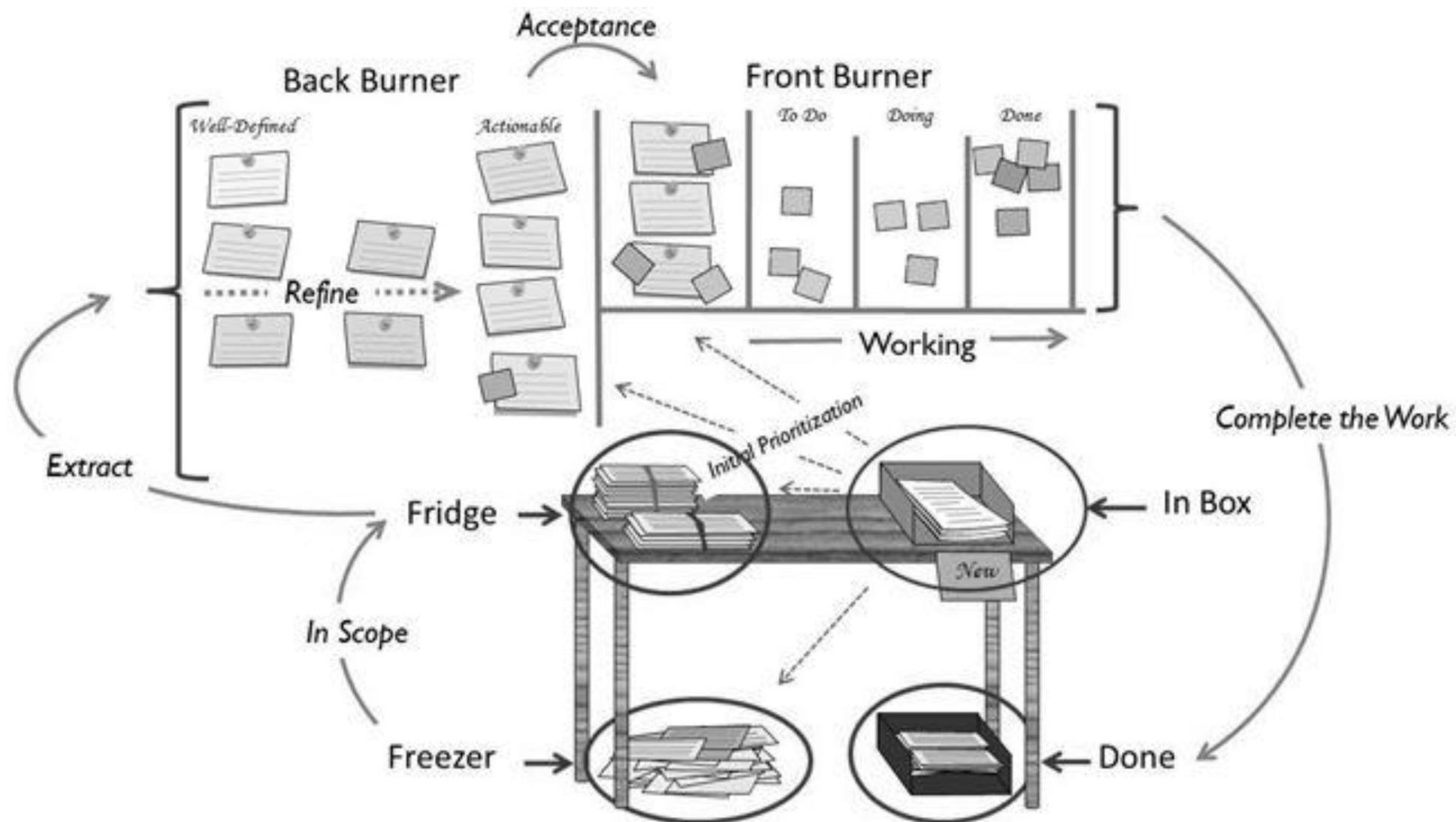
La decomposizione delle US in task

USER STORY	TASKS	DAY 1	DAY 2	DAY 3	DAY 4	DAY 5	...
As a member, I can read profiles of other members so that I can find someone to date.	Code the...	8	4	8	0		
	Design the...	16	12	10	4		
	Meet with Mary about...	8	16	16	11		
	Design the UI	12	6	0	0		
	Automate test...	4	4	1	0		
	Code the other...	8	8	8	8		
As a member, I can update my billing information	Update security tests	6	6	4	0		
	Design a solution to...	12	6	0	0		
	Write a test plan	8	8	4	0		
	Automate tests...	12	12	10	6		
	Code the...	8	8	8	4		

Gestione dello sprint backlog

- Ogni membro del team può modificare lo sprint backlog, che di solito viene conservato in un tabellone detto “taskboard” o “kanban”
- Il lavoro da fare in ogni sprint “emerge” sul tabellone
- Se il lavoro da fare è poco chiaro, conviene definire uno sprint backlog item con una stima maggiore e decomporlo più tardi
- Occorre aggiornare il lavoro da fare man mano che viene fuori

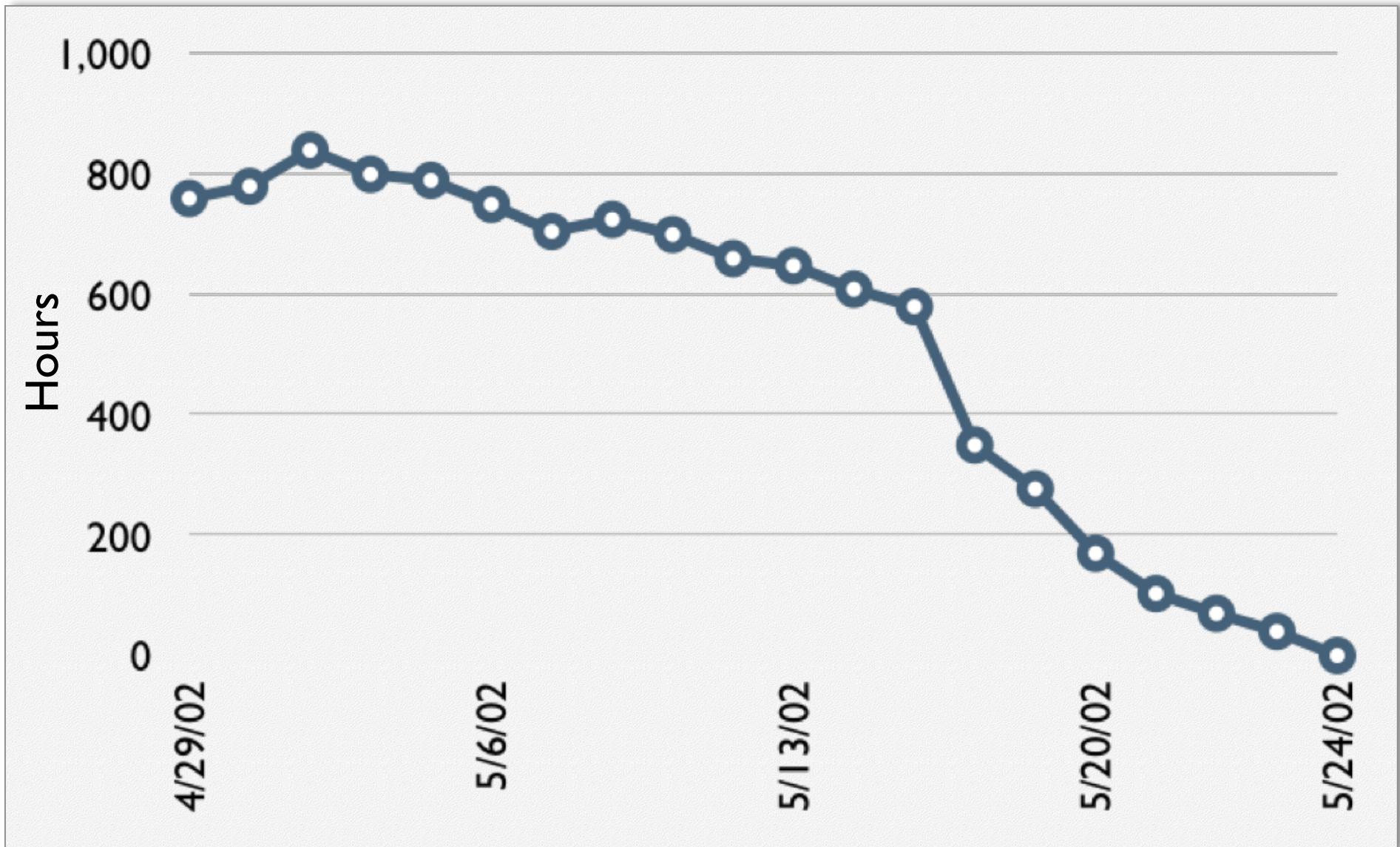
Il ciclo dei compiti in uno sprint



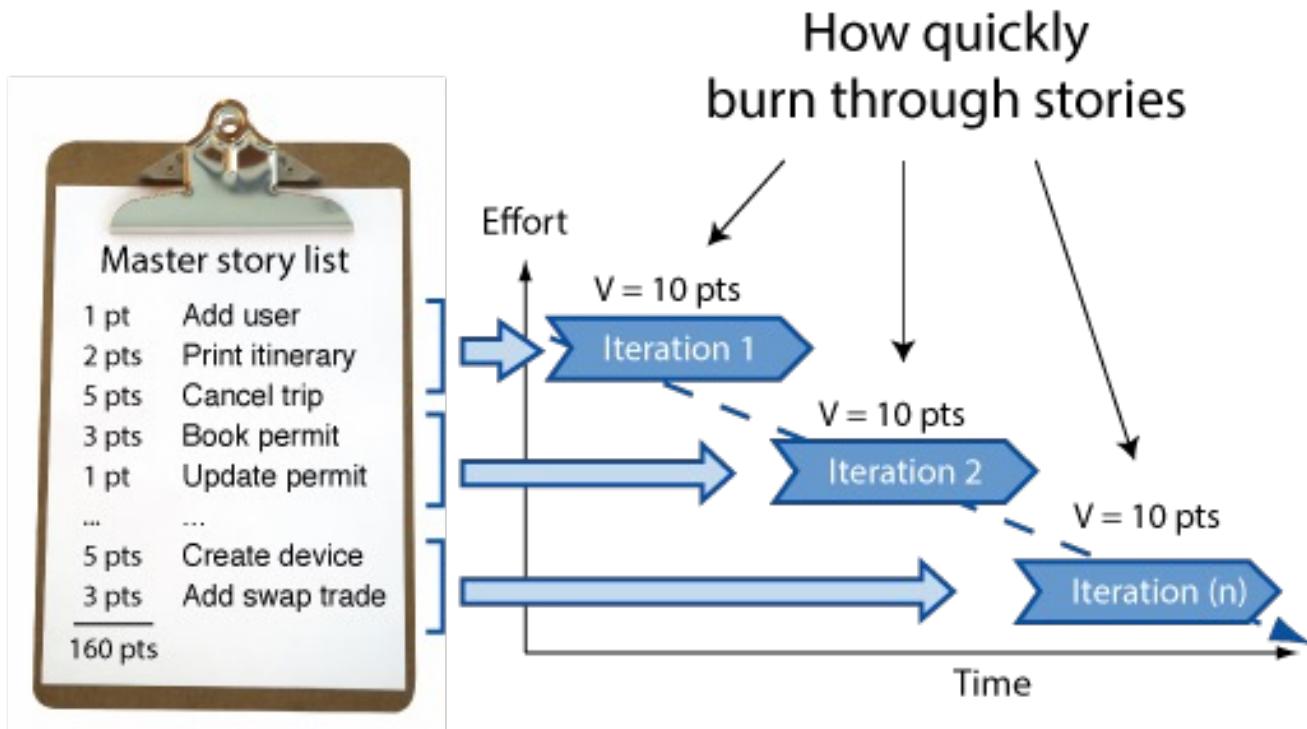
Uno sprint backlog

Tasks	Mon	Tues	Wed	Thur	Fri
Code the user interface	8	4	8		
Code the middle tier	16	12	10	4	
Test the middle tier	8	16	16	11	8
Write online help	12				
Write the foo class	8	8	8	8	8
Add error logging			8	4	

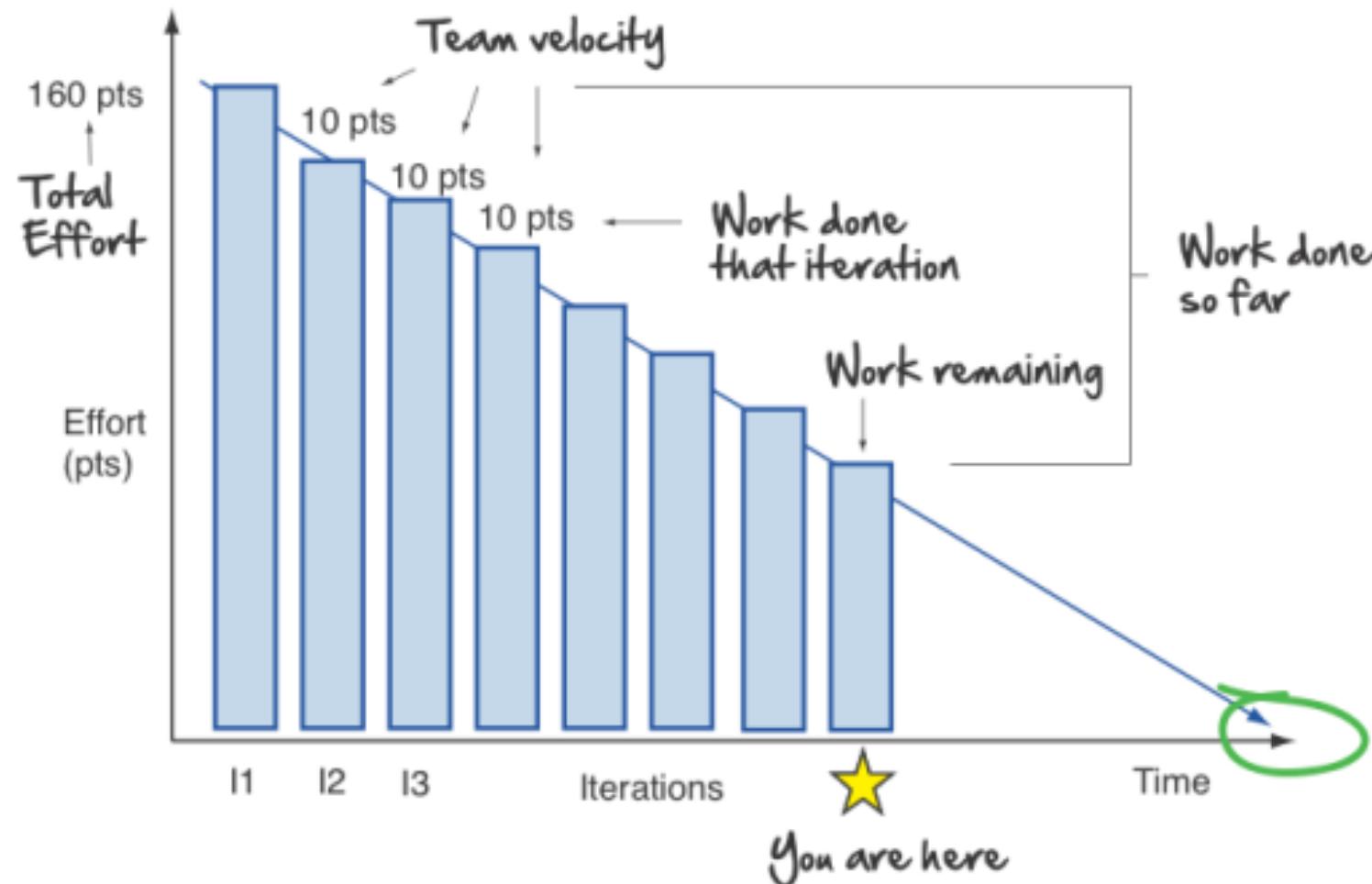
Un burndown chart di uno sprint



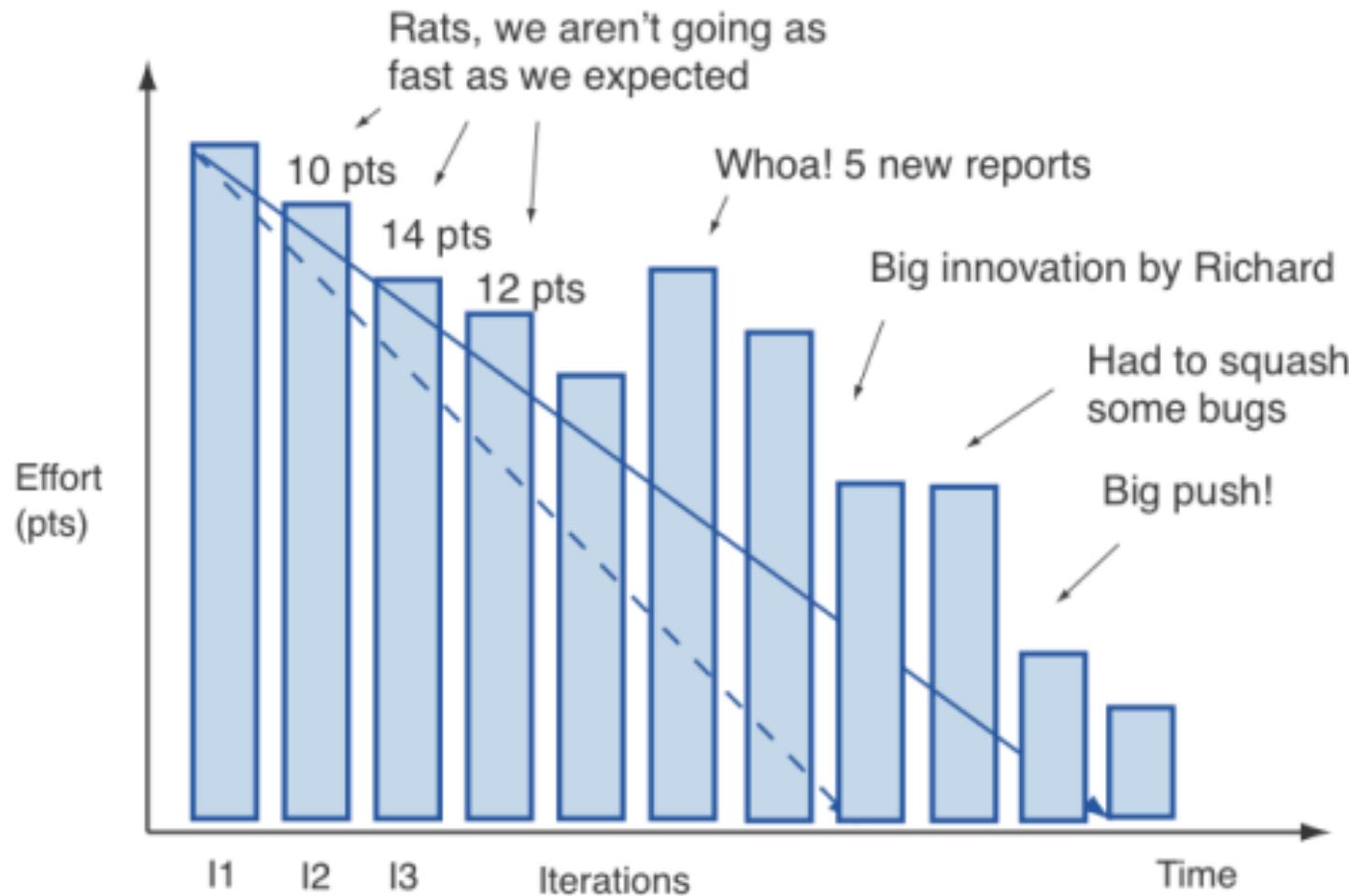
Burndown charts



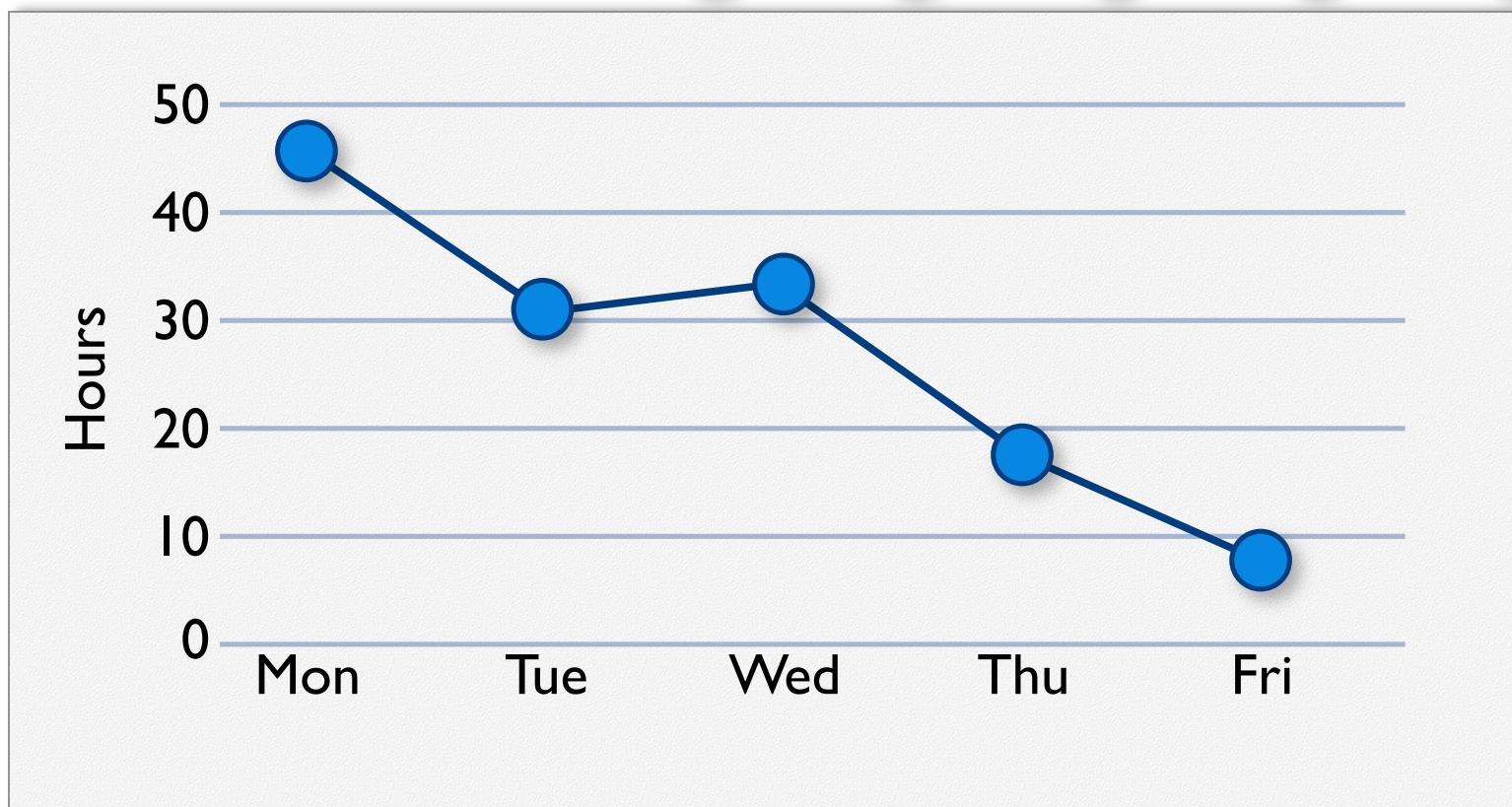
Velocity ideale sul burndown chart



Velocity reale sul burndown chart

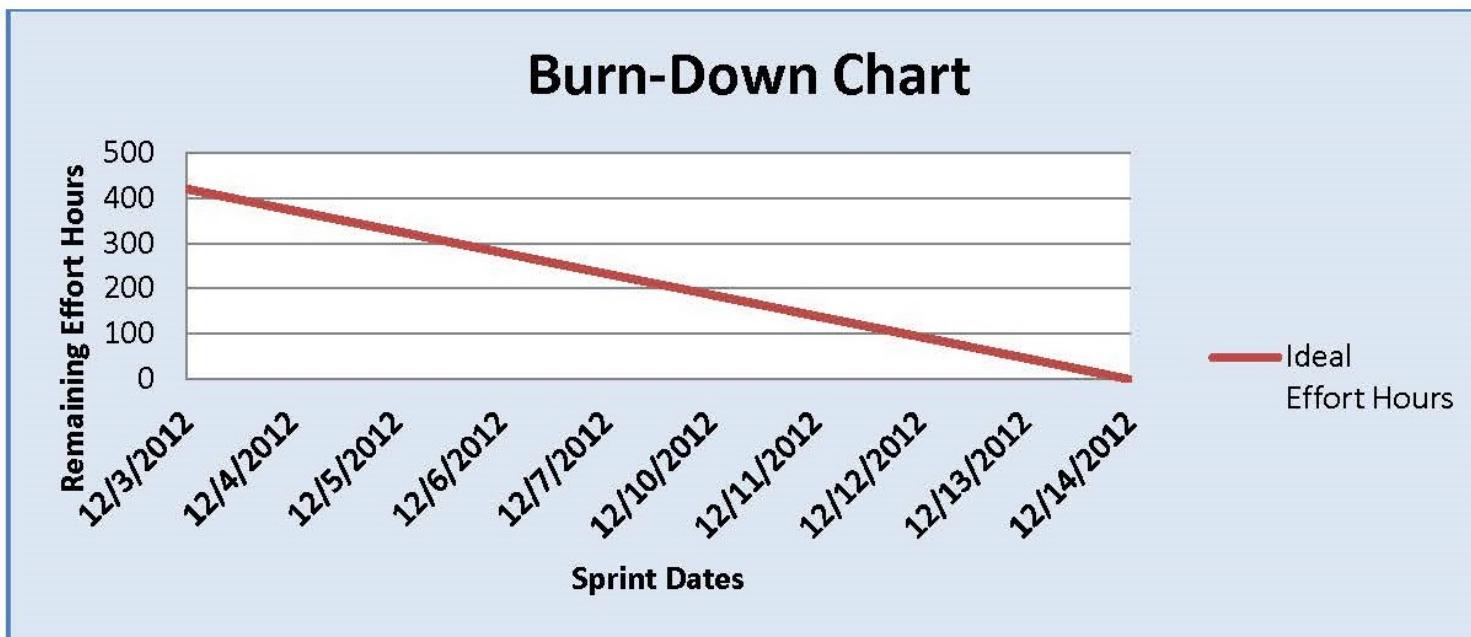


Tasks	Mon	Tues	Wed	Thur	Fri
Code the user interface	8	4	8		
Code the middle tier	16	12	10	7	
Test the middle tier	8	16	16	11	8
Write online help	12				



Esempio

Durata sprint: 2 settimane, 7 persone, 6h/giorno. Totale sforzo ideale 420 ore



<https://www.scrumalliance.org/community/articles/2013/august/burn-down-chart---an-effective-planning-and-tracki>

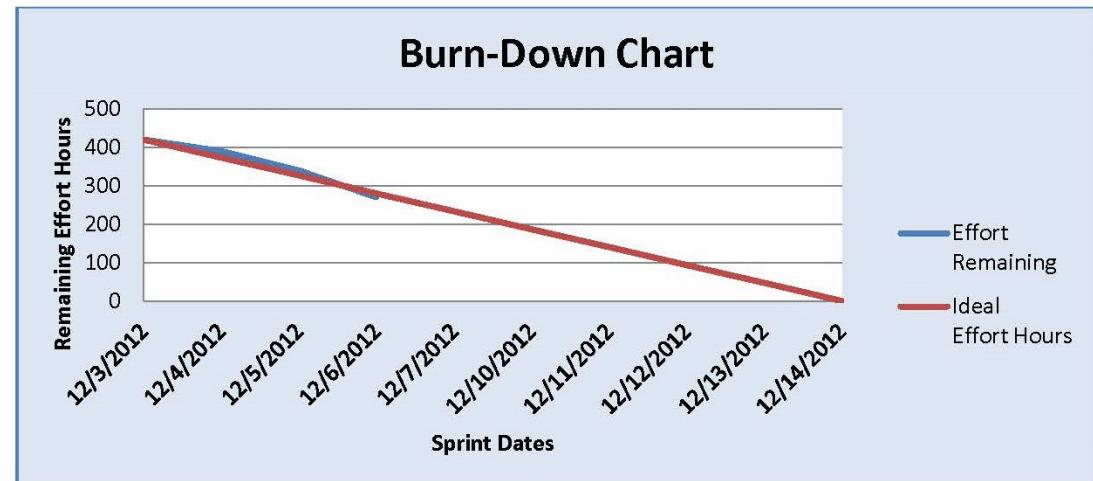
Stima continua dello sforzo rimanente

Ogni partecipante sceglie un compito e stima il tempo rimanente.

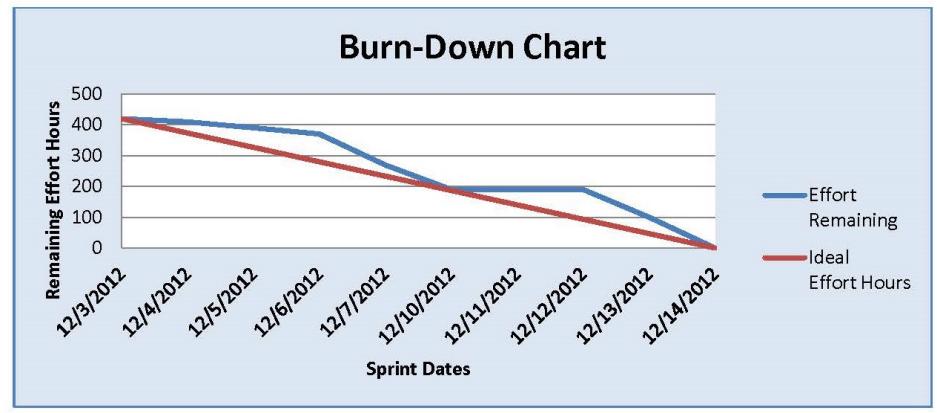
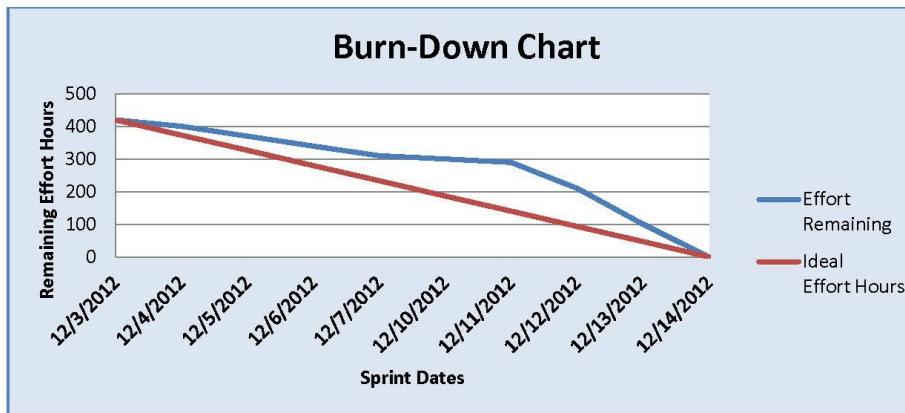
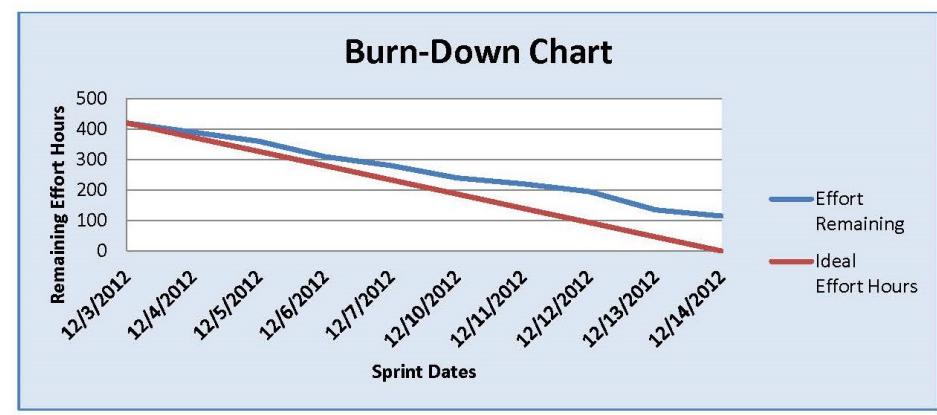
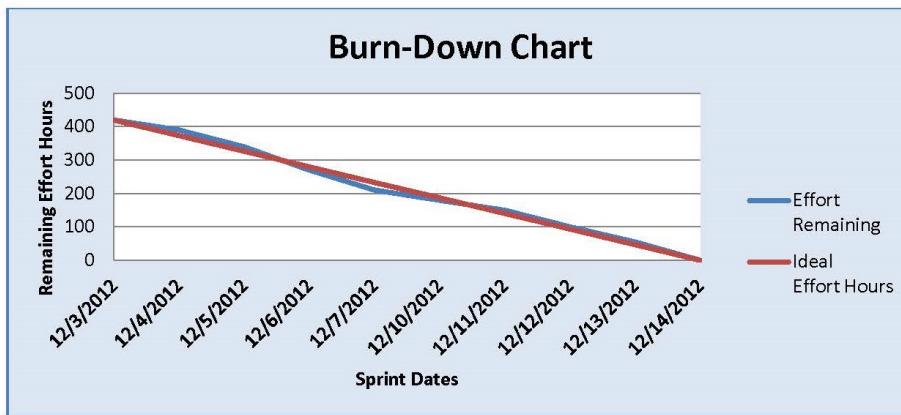
Per es. dopo il primo giorno sono state spese 6 h sul compito 1. Il programmatore stima che rimangano altre 6 ore (quindi 2h oltre la stima di 10h).

Il burn-down chart viene aggiornato e confrontato con l'ideale (blu vs rosso)

Story Name	Task No	Task Description	Status	Owner	Estimated Effort (in Hours)	Effort Remaining (in Hours)
Story 1	1	POC for Story 1	In Progress	Developer 1	10	6
	2	Requirement Clarification with PO	Closed	BA	8	0
	3	Develop modules	Open	Developer2	12	12



Varie situazioni

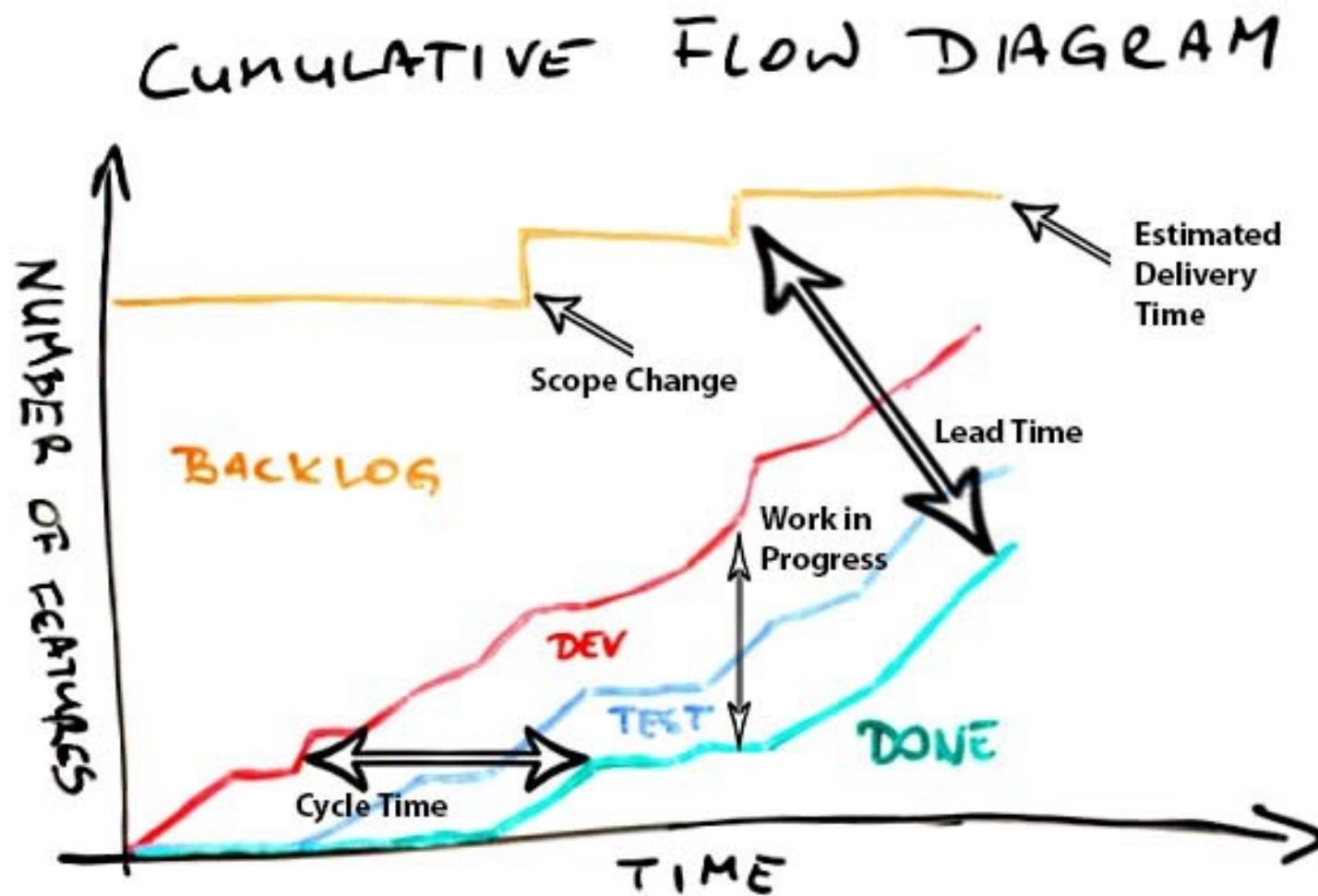


Altre metriche agili

Oltre burndown e velocity, sono rilevanti:

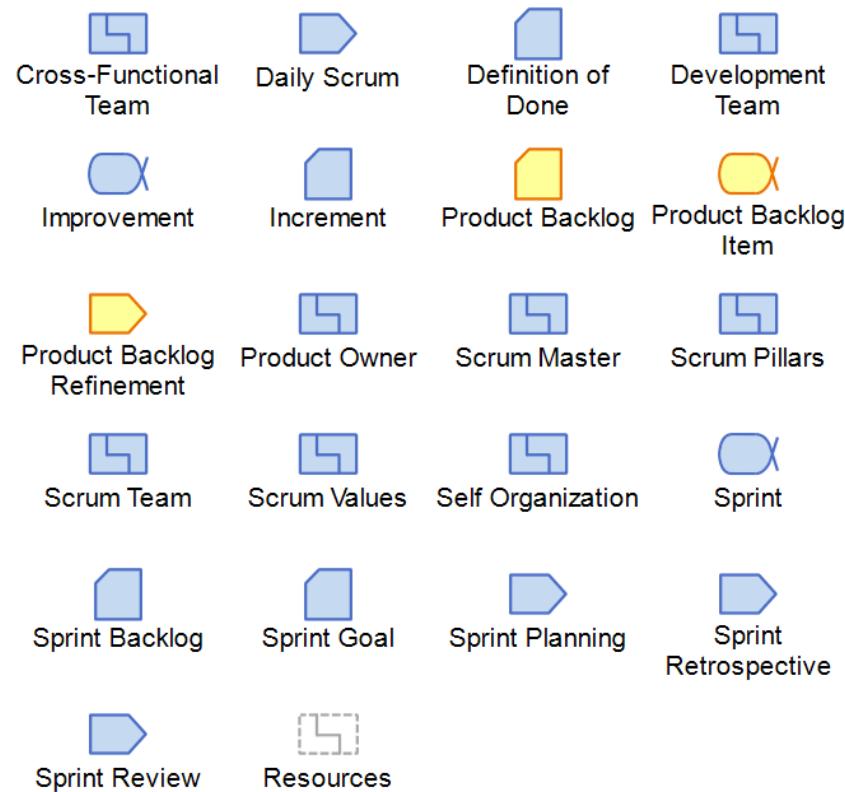
- Release burndown
- Control chart
- Cumulative flow diagram

Cumulative flow diagram



Scrum Essentials

Scrum is a framework for developing, delivering, and sustaining complex products.

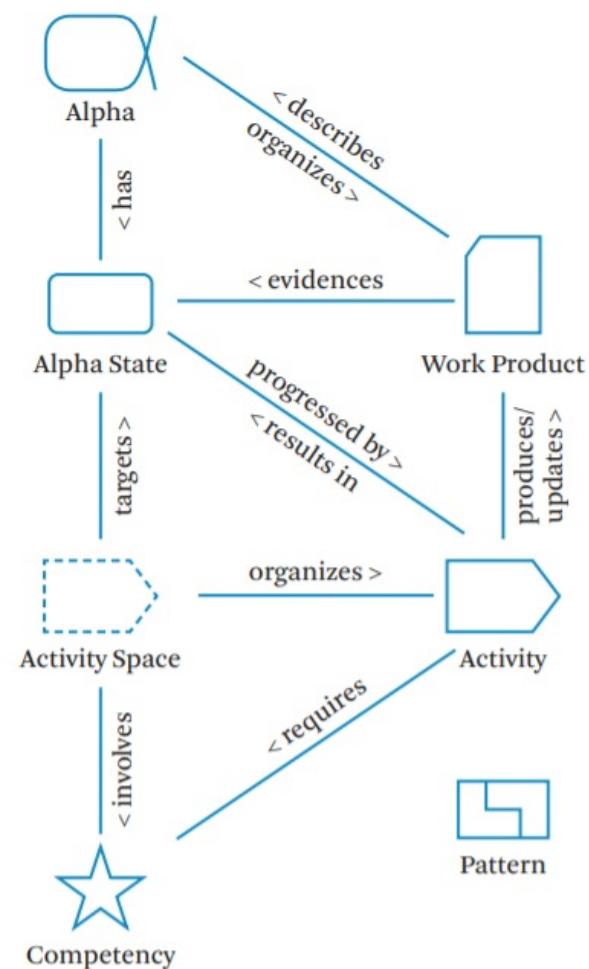


2.04

Customer

Solution

Endeavor



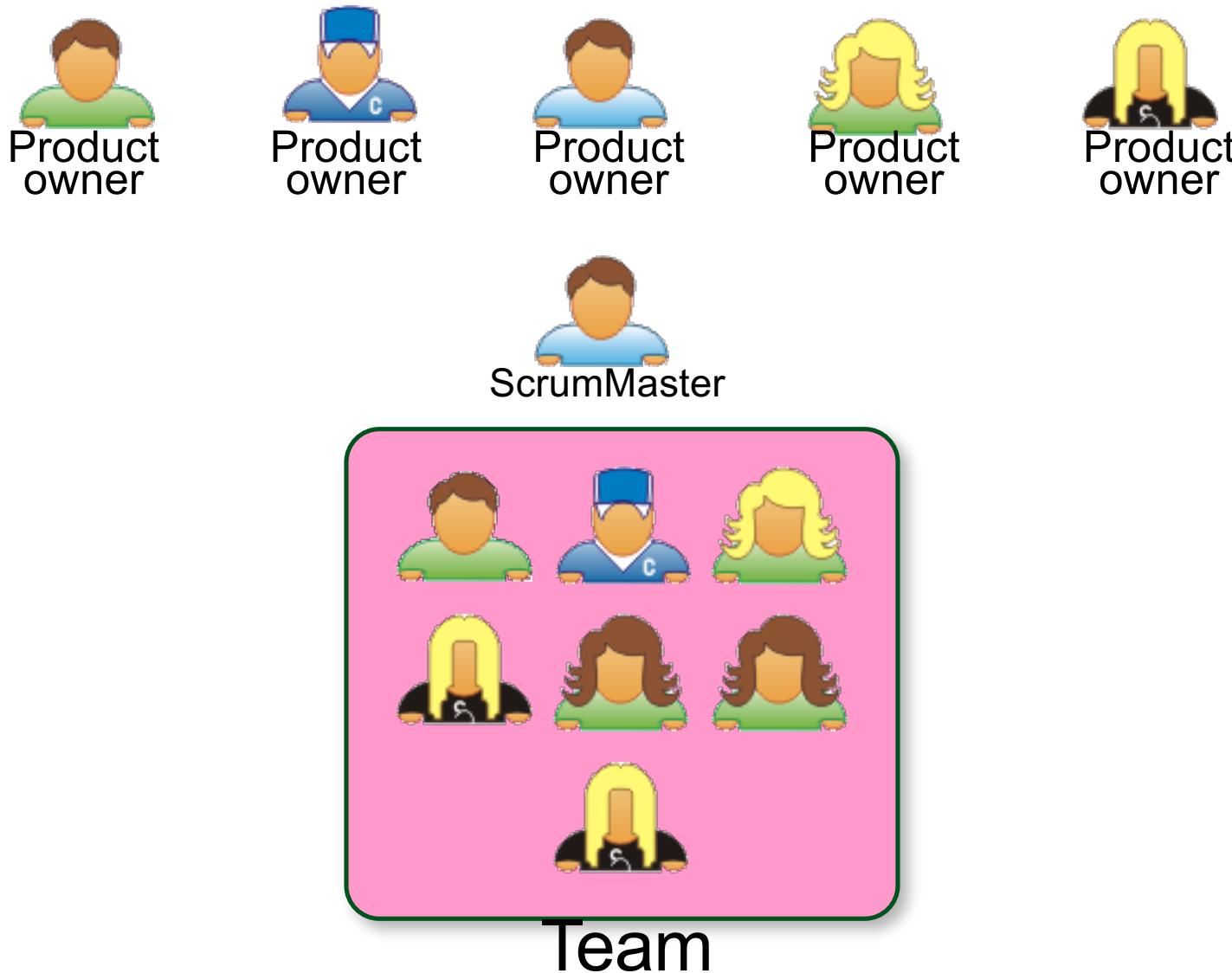
Legenda dei simboli Essence

Varianti di Scrum

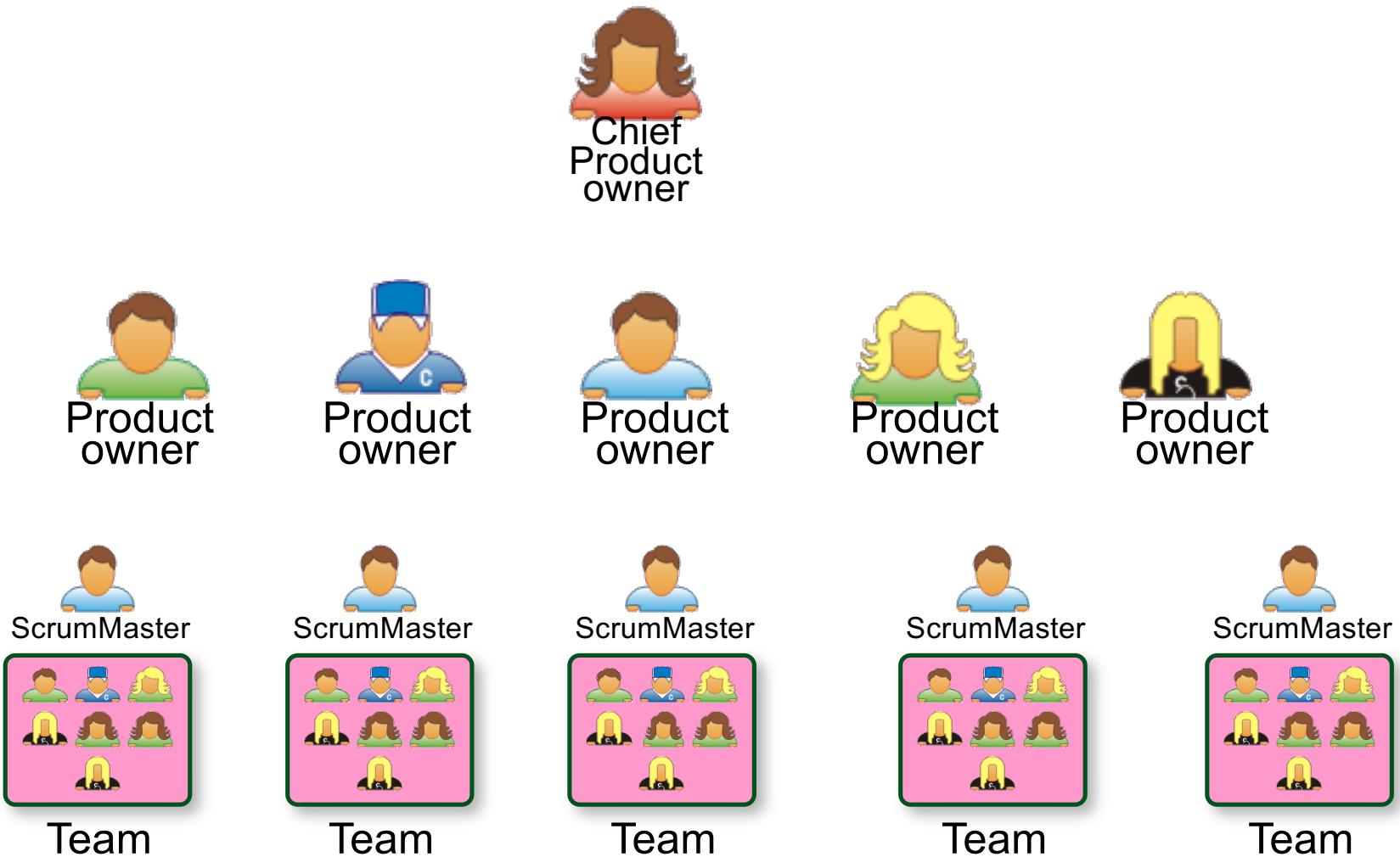
Scrum multipli

- Alcuni progetti Scrum hanno coinvolto oltre 500 persone
 - Si scala su grossi sistemi con scrum di scrum
- Fattori che guidano la strutturazione del processo con Scrum multipli
 - Tipo dell'applicazione
 - Durata del progetto
 - Dispersione dei team

Team unico, molteplici product owner



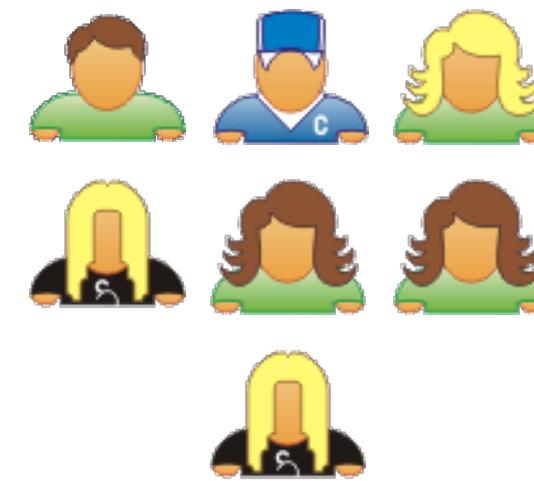
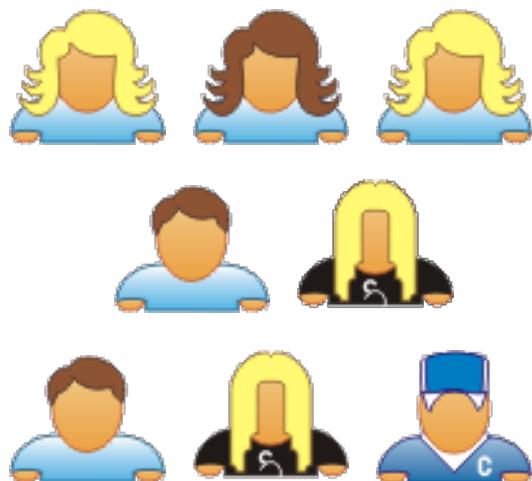
Molteplici team, prodotto unico



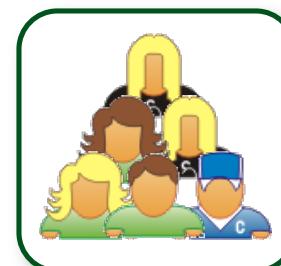
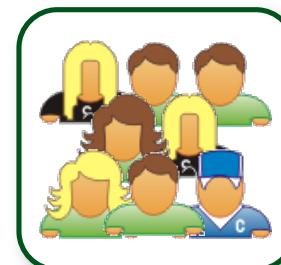
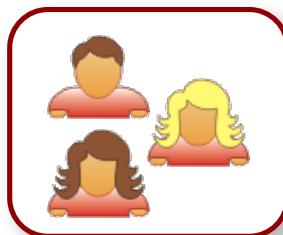
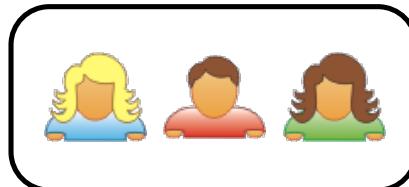
Scrum di scrum



Metascrum
degli ambasciatori



Scrum di scrum di scrum



Problemi tipici con Scrum

1. Ignoranza dei valori agili e di Scrum
2. Prodotto software non testato alla fine dello sprint
(cattiva definizione di “*Fatto*”)
3. Backlog non pronto all’inizio dello sprint (cattiva definizione di *Ready*)
4. Mancanza di facilitazione (o cattiva facilitazione)
5. Mancanza di supporto da parte dei manager
6. Mancanza di supporto da parte degli stakeholder
7. Gestione caotica degli scrum di scrums

XP o Scrum?

XP	Scrum
Orientato alla qualità (test driven)	Orientato al project management
Iterazione: 1-2 settimane	Sprint: 2-4 settimane
Requisiti sempre modificabili	Req modificabili alla fine dello sprint
Il cliente ordina le storie	Il team ordina le storie
Coaching informale	Scrum master certificato
Buone pratiche tipiche di XP: TDD, Pair programming, planning game, refactoring	Buone pratiche tipiche di Scrum: Retrospettiva post-mortem, uso di strumenti di PM, planning poker

Nota bene: XP e Scrum possono coesistere

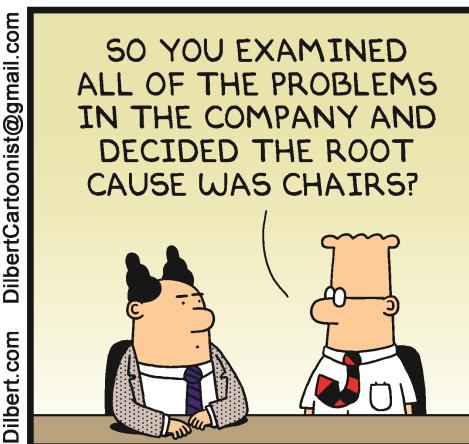
Riferimenti utili

- www.scrum.org
- www.scrumalliance.org
- scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-Italian.pdf

Una lista di libri su Scrum

- Larman: *Agile and Iterative Development: A Manager's Guide*
- Cohn: *Agile Estimating and Planning*
- Cohn: *User Stories Applied for Agile Software Development*
- Derby & Larsen: *Agile Retrospectives*
- Rubin: *Essential Scrum*

Domande?



Le forme dei requisiti



*Corso di Ingegneria del Software
Università di Bologna*

Obiettivi della lezione

- Cosa sono i requisiti di un software?
- La forma dei requisiti:
 - Frasi strutturate, scenari, casi d'uso, user stories
- Approfondimento sulle user story

Requisito:

definizione dal vocabolario italiano

- Ciascuna delle qualità necessarie e richieste per uno scopo determinato.
- esempio: *ha tutti i requisiti per diventare un bravo informatico*

I requisiti sono desideri!



Come si esprimono i desideri dal punto di vista dello sviluppo software?

Requisito funzionale:

«Il sistema permetterà di prenotare un taxi e di avere una stima del tempo di attesa»

Scenario:

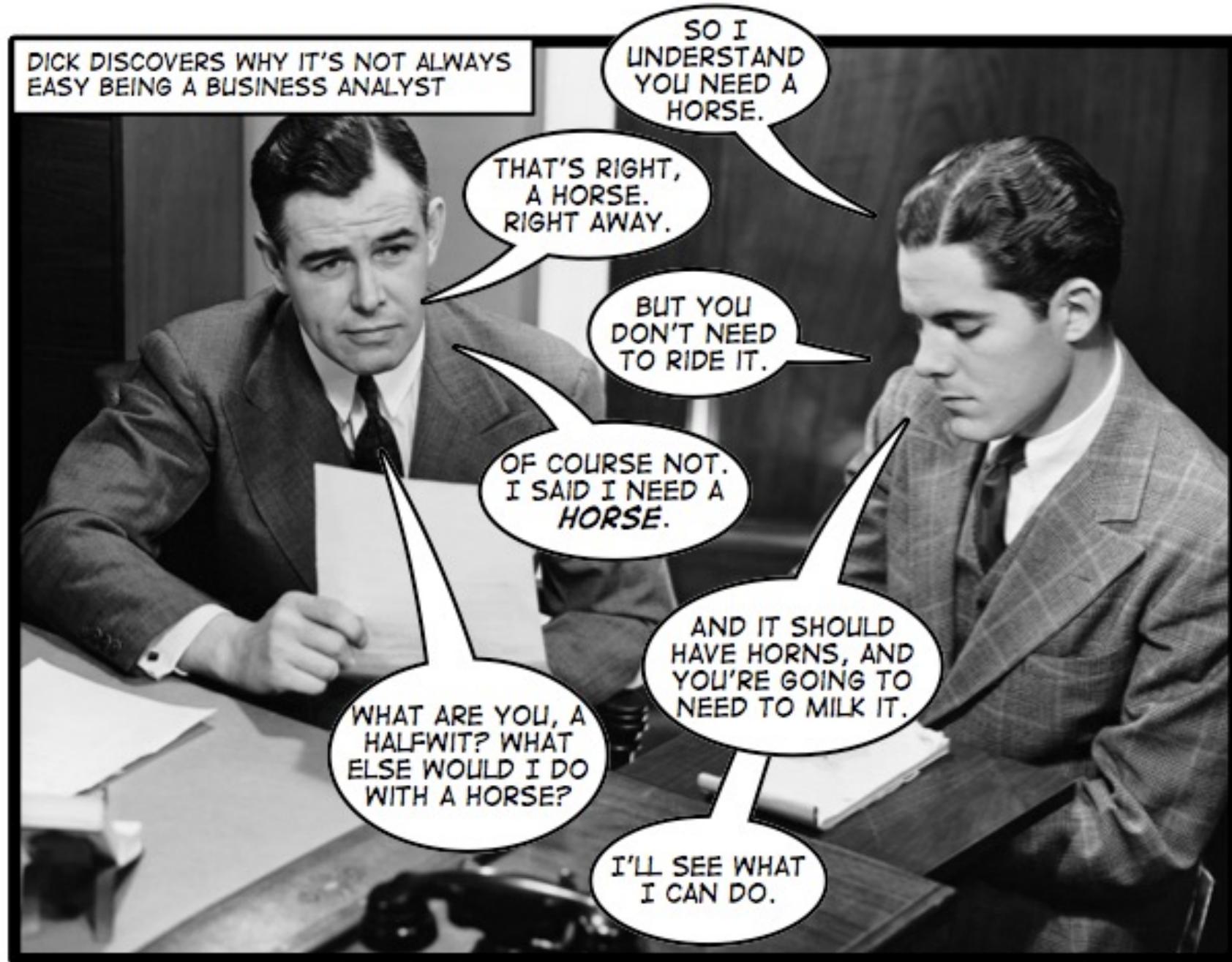
Cliente: “ho bisogno di un taxi in via Zamboni 33”

Sistema: “Fragola 33 arriverà in 3 minuti”

User story:

Paolo vuole poter chiamare un taxi da una app e ottenere risposta con stima del tempo di arrivo, per calcolare se arriverà in tempo al suo appuntamento

Ma cos'è un desiderio?



Legge di Humphrey

I requisiti di un nuovo prodotto software
non saranno chiari finché gli utenti non
iniziano a usarlo,
ovvero un problema non è ben compreso
finché non si sviluppa una soluzione

Gli utenti: “I'll Know It When I See It” (IKIWISI)

Perché esistono tanti forme per i requisiti?

La forma centrata sull'utente

La forma centrata sullo scenario d'uso

La forma contrattuale

La forma adeguata agli sviluppatori

Analisi dei requisiti

(Wikipedia)

L'**analisi dei requisiti** è un'attività preliminare allo sviluppo di un prodotto software, e serve a definire le funzionalità da offrire, ovvero i *requisiti che devono essere soddisfatti* dal software sviluppato

L'analisi dei requisiti è una fase presente in tutti i modelli di ciclo di vita del software, pur con diverse enfasi e diverse connotazioni

Nel nostro corrisponde alla definizione e modifica del backlog di prodotto

Esempio

- Una biblioteca gestisce prestiti di 100.000 volumi a 5.000 iscritti.
- La biblioteca è dotata di un sistema di catalogazione dei libri.
- I volumi sono catalogati con i metadati bibliografici usuali (autore, titolo, editore, anno, ecc.) e identificati mediante il proprio ISBN ed un contatore di copia.
- Ci sono due tipi d'utente: il bibliotecario e l'iscritto; il primo può aggiornare la base di dati, mentre il secondo può solo consultare i dati dei libri. A tutti gli utenti sarà fornita un'interfaccia Web standard utilizzabile anche da casa.
- Un iscritto chiede alla biblioteca il prestito di uno o più volumi alla volta mediante un Web browser; la biblioteca invia al cliente la lista dei volumi disponibili.
- I libri sono prestati agli iscritti della biblioteca e gli iscritti sono identificati sia da un codice numerico, che dal cognome, nome e data di nascita.
- Il bibliotecario accede mediante password alle operazioni d'aggiornamento, mentre l'iscritto accede liberamente alle operazioni di consultazione
- L'applicazione da progettare deve consentire l'inserimento dei dati delle nuove acquisizioni, l'iscrizione di nuovi utenti, la registrazione dei prestiti, il rientro del libro, il controllo del prestito e la consultazione dei libri disponibili mediante i metadati bibliografici.

Discussione

- Questo testo cosa descrive?
- Sapreste scrivere un software in base ad esso, senza ulteriori interazioni col cliente?



La stesura dei requisiti

- Stabilire **cosa** richiede il cliente ad un sistema software (scopo)
- **senza** definire **come** il sistema verrà costruito (funzioni)

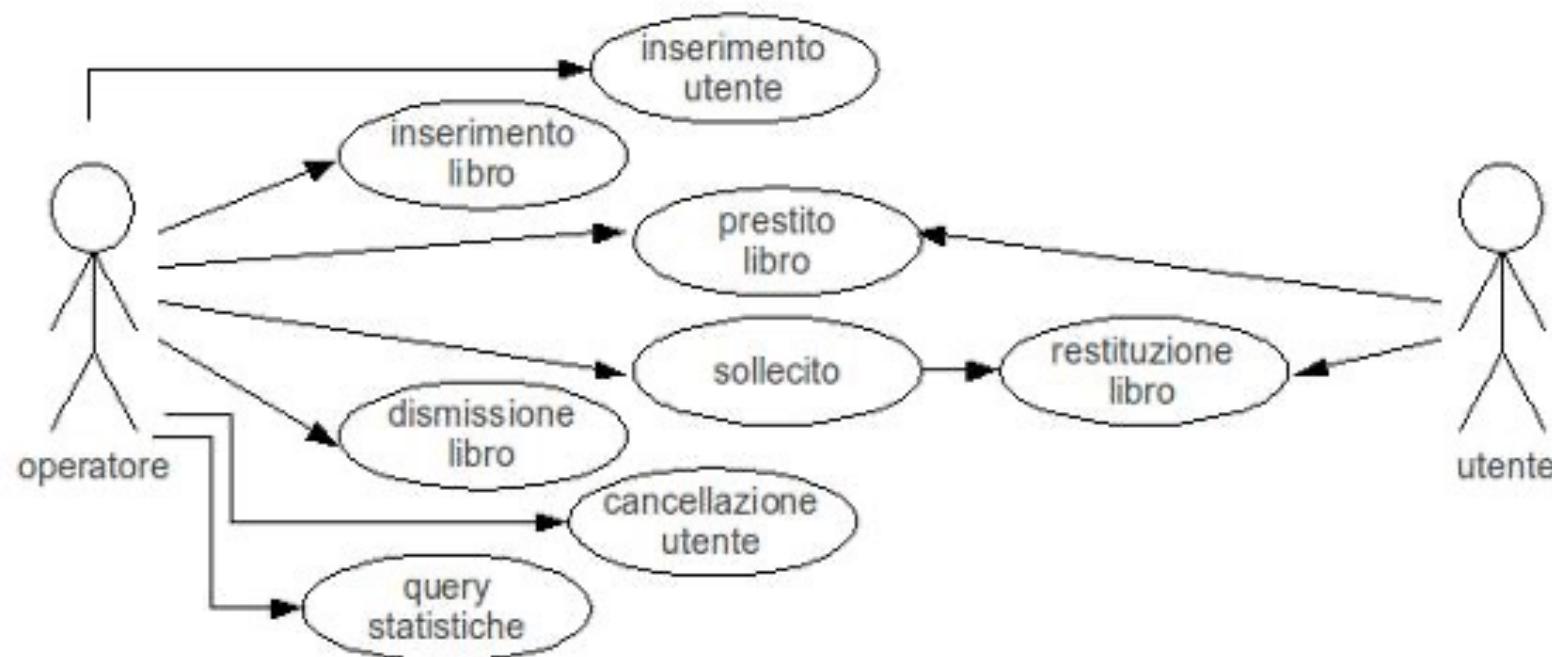
Requisiti per la biblioteca

(scritti in formato funzionale)

1. Il sistema gestirà due tipi di utenti: l'operatore bibliotecario e l'utente «normale»
2. Il sistema permetterà all'utente di ottenere un libro in prestito
3. Il sistema permetterà all'utente di restituire un libro
4. Il sistema permetterà al bibliotecario di inserire un nuovo libro nel catalogo
5. Il sistema permetterà al bibliotecario di dismettere un libro dal catalogo
6. Il sistema permetterà al bibliotecario di inserire un nuovo utente registrato
7. Il sistema permetterà al bibliotecario di cancellare un utente
8. Il sistema permetterà al bibliotecario di approvare un prestito di un libro
9. Il sistema permetterà al bibliotecario di sollecitare la restituzione di un libro
10. Il sistema permetterà al bibliotecario di elaborare statistiche sui prestiti
11. eccetera
12. ... (continua) ...

Requisiti per la biblioteca

(disegnati graficamente come **casi d'uso**)



Esempio di scenario per caso d'uso «prestito»

Titolo del Caso d'Uso: Prestito di un Libro

Attori Principali: Utente

- **Scenario Principale**

1. L'utente avvia l'applicazione della biblioteca.
2. L'utente effettua l'accesso al suo account, fornendo le credenziali di accesso.
3. L'utente cerca il libro desiderato utilizzando la funzione di ricerca all'interno dell'app.
4. L'utente seleziona il libro che desidera dalla lista dei risultati della ricerca.
5. L'applicazione verifica la disponibilità del libro. Se il libro è disponibile, procede col prestito; altrimenti, mostra un messaggio di avviso.
6. L'utente conferma la richiesta di prestito.
7. L'applicazione registra il prestito del libro a nome dell'utente.
8. L'applicazione fornisce una conferma del prestito, includendo la data di scadenza prestito
9. L'utente può ora visualizzare il libro nel proprio account come libro in prestito.

- **Scenario Alternativo**

- 6a. Se il libro non è disponibile, l'applicazione mostra un messaggio di errore e l'utente può decidere se cercare un altro libro o uscire dall'app.

Requisiti

- I requisiti sono **descrizioni** di ciò che occorre e andrebbe realizzato
- Sono descrizioni di 1) **cosa** il sistema dovrebbe fare, oppure 2) di una **proprietà** o attributo del sistema
- Possono anche essere 3) un **vincolo** sul processo di sviluppo del sistema

Esempio:

- **Cosa** dovrebbe fare il sistema: permettermi di comprare un biglietto del treno
- **Attributo** del sistema: l'acquisto dovrebbe richiedere non più di un minuto
- **Vincolo**: l'acquisto deve avvenire su un telefono Android

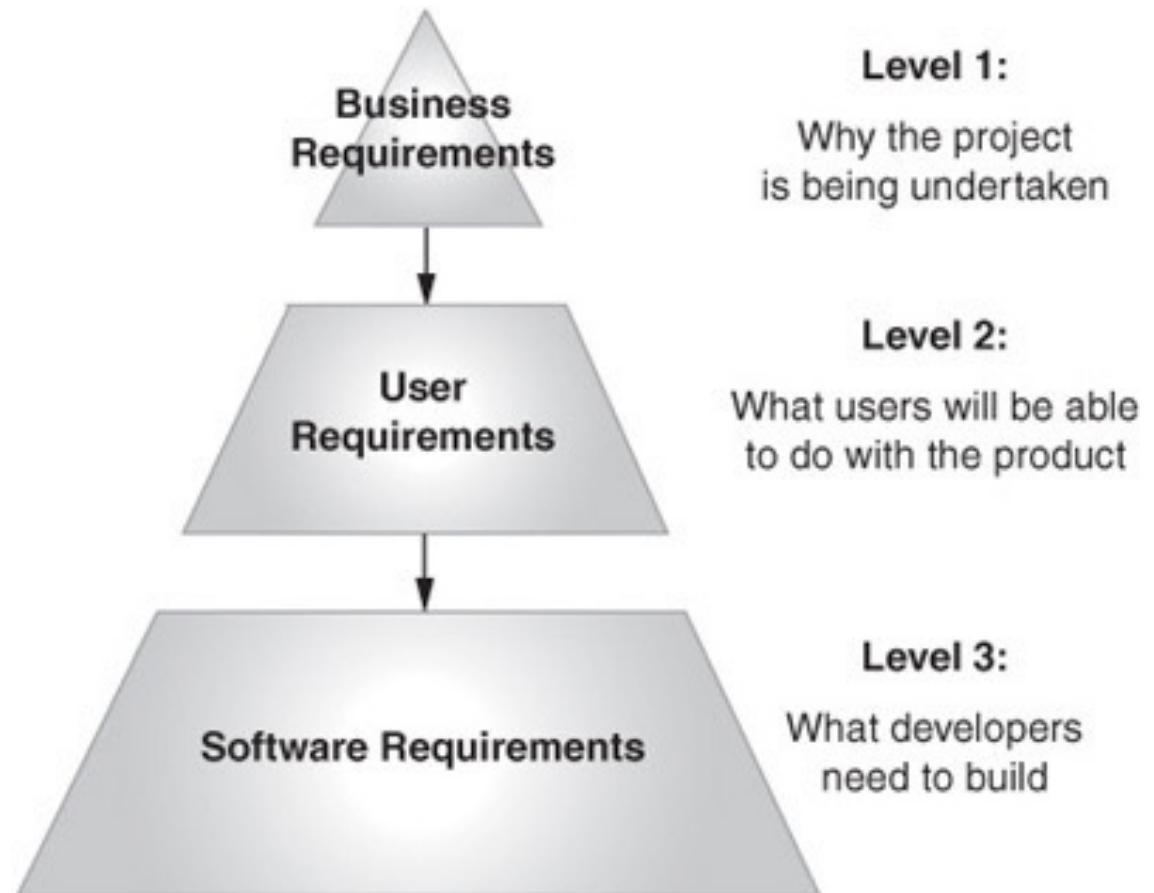
Requisito

Il requisito dice <chi> vuole
<cosa> e <perché>

Un requisito dice **cosa** un
sistema dovrebbe fare, ma
non **come**

I requisiti possono essere:

- di business
- di utente
- del software



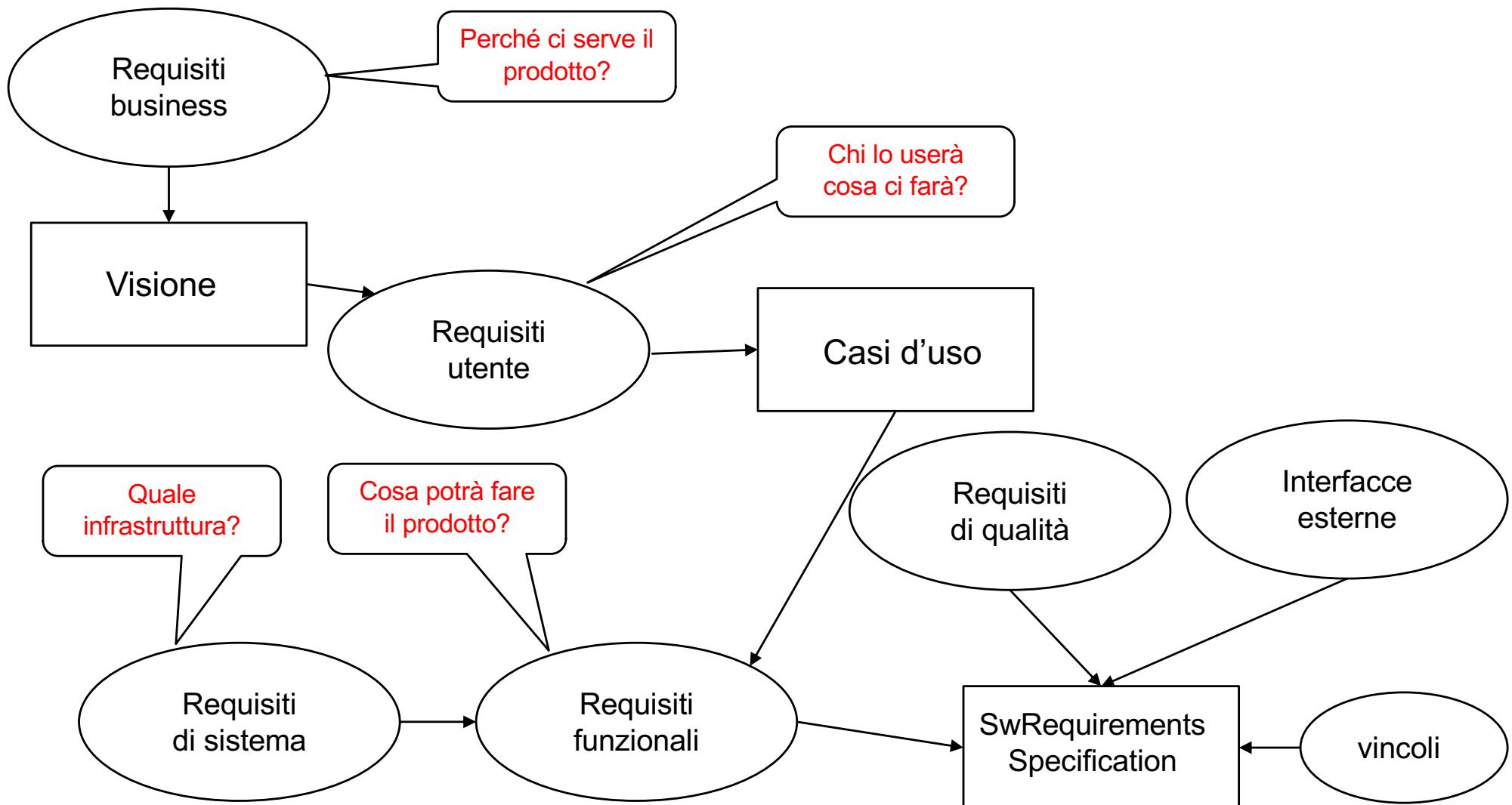
Esempi

Requisito di business: *il sistema biglietteria deve gestire almeno 1M di biglietti al giorno*

Requisito utente: *come viaggiatore voglio comprare un biglietto da Bologna a Milano per far viaggiare mia mamma*

Requisito software: *il record Biglietto conterrà il campo Persona di tipo vettore di stringhe*

Ci sono molti tipi di requisiti e molti modi di specificarli



Discussione

Con quale formato possiamo scrivere i requisiti funzionali di un prodotto digitale?

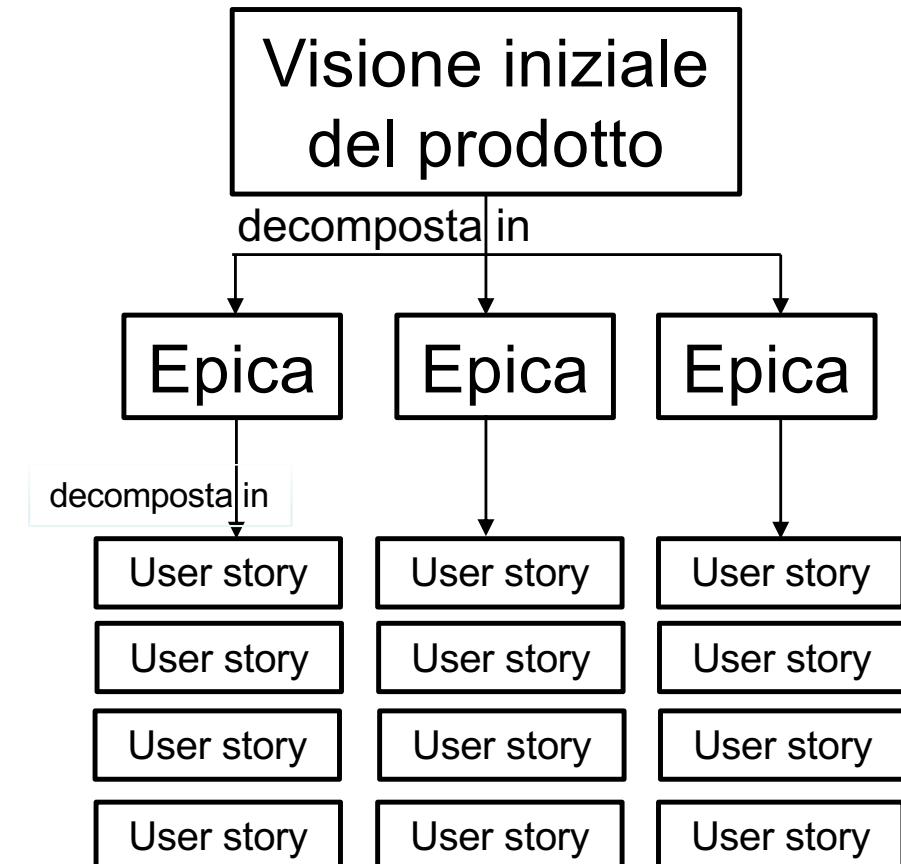


La forma del requisito

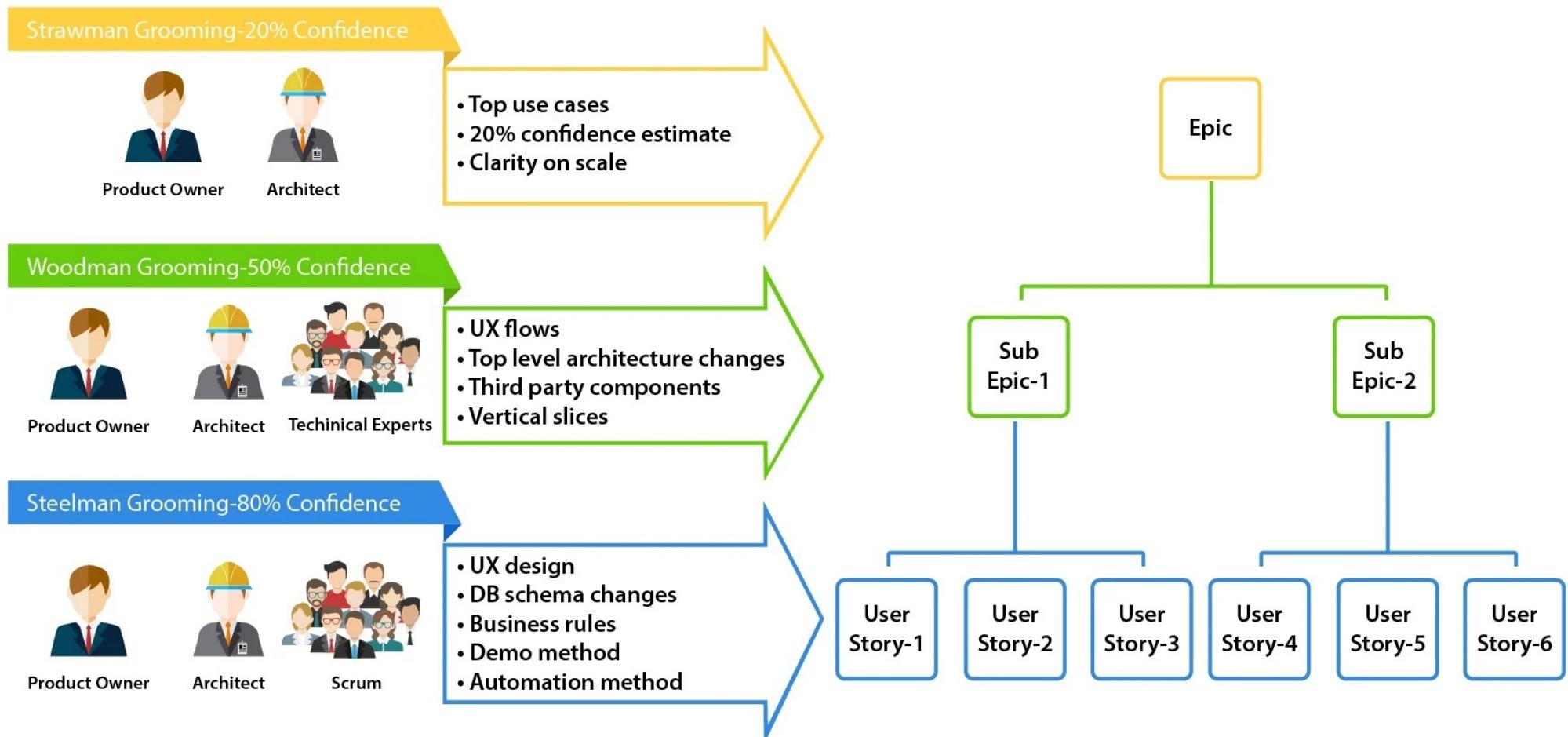
- **Epica**: «*La biblioteca ha bisogno di una app per prestiti e restituzioni dei libri*»
- **User story**: «*Come iscritto voglio consultare il catalogo per prendere in prestito un volume*»
- **Caso d'uso**: «*Un iscritto interroga la funzione Catalogo per cercare un volume, il Catalogo chiede all'iscritto i dati necessari alla ricerca, l'iscritto inserisce i dati, il Catalogo risponde...*»
- **Funzionale**: «*L'applicazione gestirà i prestiti dei volumi agli iscritti*»

Epiche

- L'epica rappresenta un insieme di funzionalità ad alto livello richieste in una visione di prodotto
- L'epica aiuta a identificare i requisiti principali di un prodotto e fornisce una visione d'insieme di ciò che deve essere sviluppato.
- In genere, un'epica è troppo grande per essere implementata in una singola iterazione di sviluppo e deve essere decomposta in più user story
- la user story rappresenta una singola funzionalità implementabile in una sola iterazione



Raffinare il backlog



I requisiti con le user stories

La forma agile dei requisiti: le user stories

Nei modelli di sviluppo agili i requisiti si scrivono mediante le **user stories**, che hanno questa forma:

Come <*tipo di utente*> (*ruolo*)

Voglio <*fare qualcosa*> (*funzione*)

Per <*conseguire un obiettivo*> (*scopo*)

Esempi:

Come *utente della biblioteca*

Voglio *consultare il catalogo*

Per *prendere a prestito un volume*

Come *bibliotecario*

Voglio *consultare il catalogo*

Per *aggiungere un nuovo libro*

1 Individuare gli utenti

- Una user story descrive come qualcuno usa un prodotto
- Se non sapete ancora chi saranno gli utenti oppure come useranno il prodotto **NON scrivete le user story**

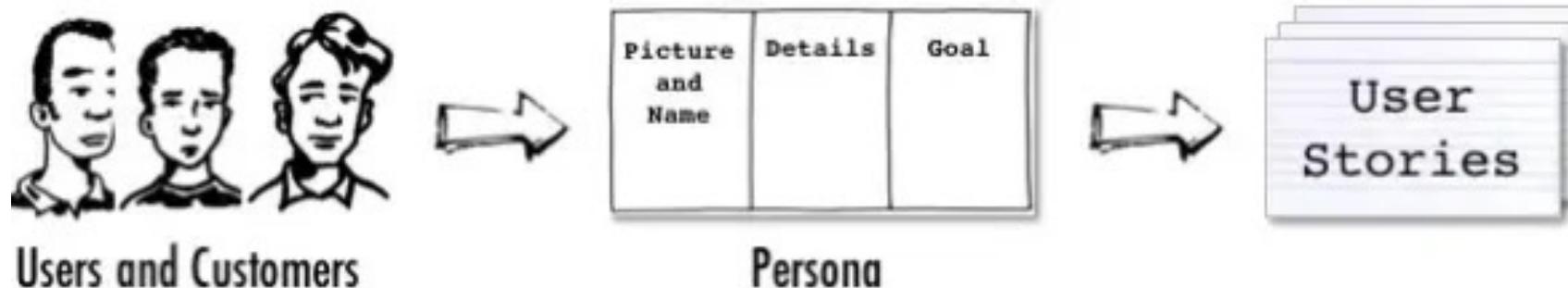


2 usare utenti ideali (*personae*)

Le *personae* sono personaggi immaginari che si basano sulla conoscenza diretta del target del prodotto.

Di solito sono costituiti da: un nome e da un'immagine; alcune caratteristiche e comportamenti rilevanti; e un obiettivo personale.

L'obiettivo è il beneficio che la persona vuole ottenere, o il problema che vuole risolvere usando il prodotto



Schema per definire una persona

ROMAN'S PERSONA TEMPLATE



PICTURE & NAME	DETAILS	GOAL
 What does the persona look like? What is its name? Choose a realistic and believable picture and name.	 What are the persona's relevant characteristics and behaviours? For instance, demographics, such as age, gender, occupation, and income; psychographics, including lifestyle, social class, and personality; and behavioural attributes like usage patterns, attitudes, and brand loyalty. Only list relevant details.	 What problem does the persona want to solve or which benefit does the character seek? Why would the persona want to use or buy the product?

3 creare le US collaborando coi devs

Le US sono uno strumento ed un prodotto di conversazioni non solo con gli utenti ma anche con i devs



<https://ronjeffries.com/xprog/articles/expcardconversationconfirmation/>

4 le US sono semplici e concise

Come *<persona>*
voglio *<cosa?>*
in modo che *<perché>*

Nota bene: Questo
schema è utile ma non
obbligatorio

Nella prossima slide
vedremo una forma
ammissibile anche se
diversa

Forma situazionale di user story

Quando _____ voglio poter_____ cosicché_____
Situazione. *Motivazione.* *Risultato atteso*

Quando ho bisogno di un libro voglio poterlo prenotare per averlo al più presto

Quando cambio telefono voglio usare il vecchio per configurare il nuovo

Quando prendo un treno voglio poter comprare il biglietto per partire presto

Requirements

What the software system must do to address the opportunity and satisfy the stakeholders.

Conceived

Bounded

Coherent

Acceptable

Addressed

Fulfilled



Generated by UI Practice Workbench™

1.1.1

Requirements

Conceived

- Stakeholders agree system is to be produced
- Users identified
- Funding stakeholders identified
- Opportunity clear

1 / 6



Generated by UI Practice Workbench™

2018.09

Requirements

Bounded

- Development stakeholders identified
- System purpose agreed
- System success clear
- Shared solution understanding exists
- Requirement's format agreed
- Requirements management in place
- Prioritization scheme clear
- Constraints identified & considered
- Assumptions clear

2 / 6



Generated by UI Practice Workbench™

2018.09

Requirements

Coherent

- Requirements shared
- Requirements' origin clear
- Rationale clear
- Conflicts addressed
- Essential characteristics clear
- Key usage scenarios explained
- Priorities clear
- Impact understood
- Team knows & agrees on what to deliver

3 / 6



Generated by UI Practice Workbench™

2018.09

Requirements

Acceptable

- Acceptable solution described
- Change under control
- Value to be realized clear
- Clear how opportunity addressed
- Testable

4 / 6



Generated by UI Practice Workbench™

2018.09

Requirements

Addressed

- Enough addressed to be acceptable
- Requirements and system match
- Value realized clear
- System worth making operational

5 / 6



Generated by UI Practice Workbench™

2018.09

Requirements

Fulfilled

- Stakeholders accept requirements
- No hindering requirements
- Requirements fully satisfied

6 / 6



Generated by UI Practice Workbench™

2018.09

Conclusioni

- Le user story sono una delle modalità preferite per scrivere i requisiti di un prodotto digitale
- La gestione del prodotto arriva a definire le user story dopo un'analisi di mercato che include la caratterizzazione di *personae* ideali utenti del prodotto

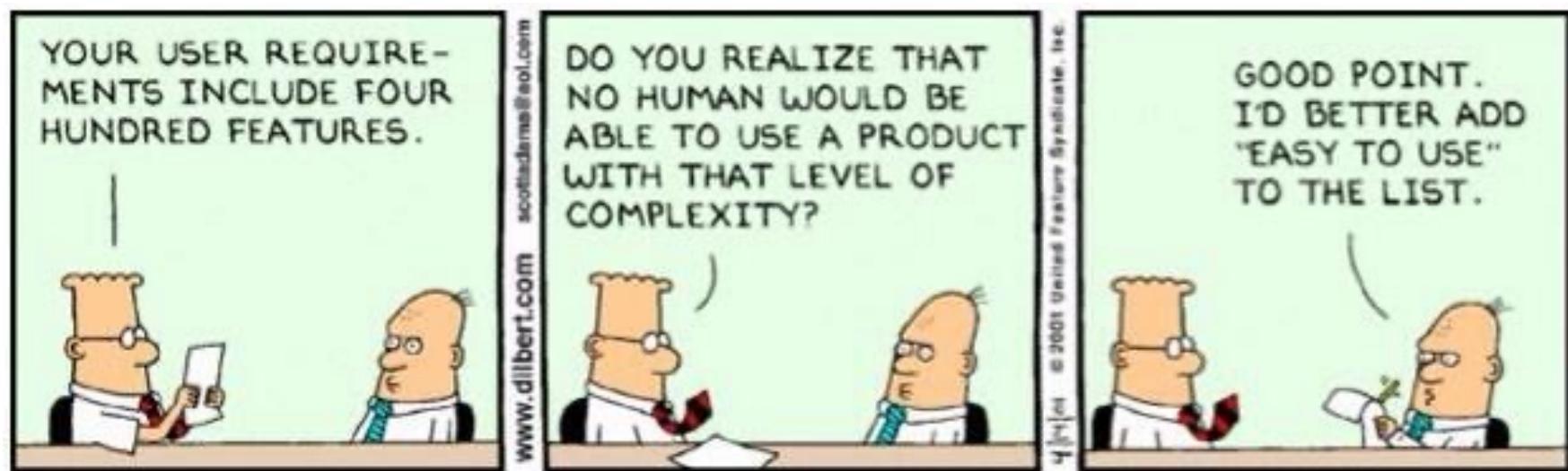
Domande di autovalutazione

- Cosa asserisce la legge di Humphrey?
- Cos'è un requisito utente?
- Qual è la forma standard delle user story?
- Cos'è un'epica?
- Quali sono gli stati di un requisito nel modello Essence?

Riferimenti

- Cohn, *User stories applied*, Addison-Wesley, 2004
- Pichler, *Agile product management with Scrum*, Addison-Wesley, 2010
- Sedano, Ralph, Perarire, The Product Backlog, Proc. 41° ICSE, 2019

Domande?



Il Product Owner

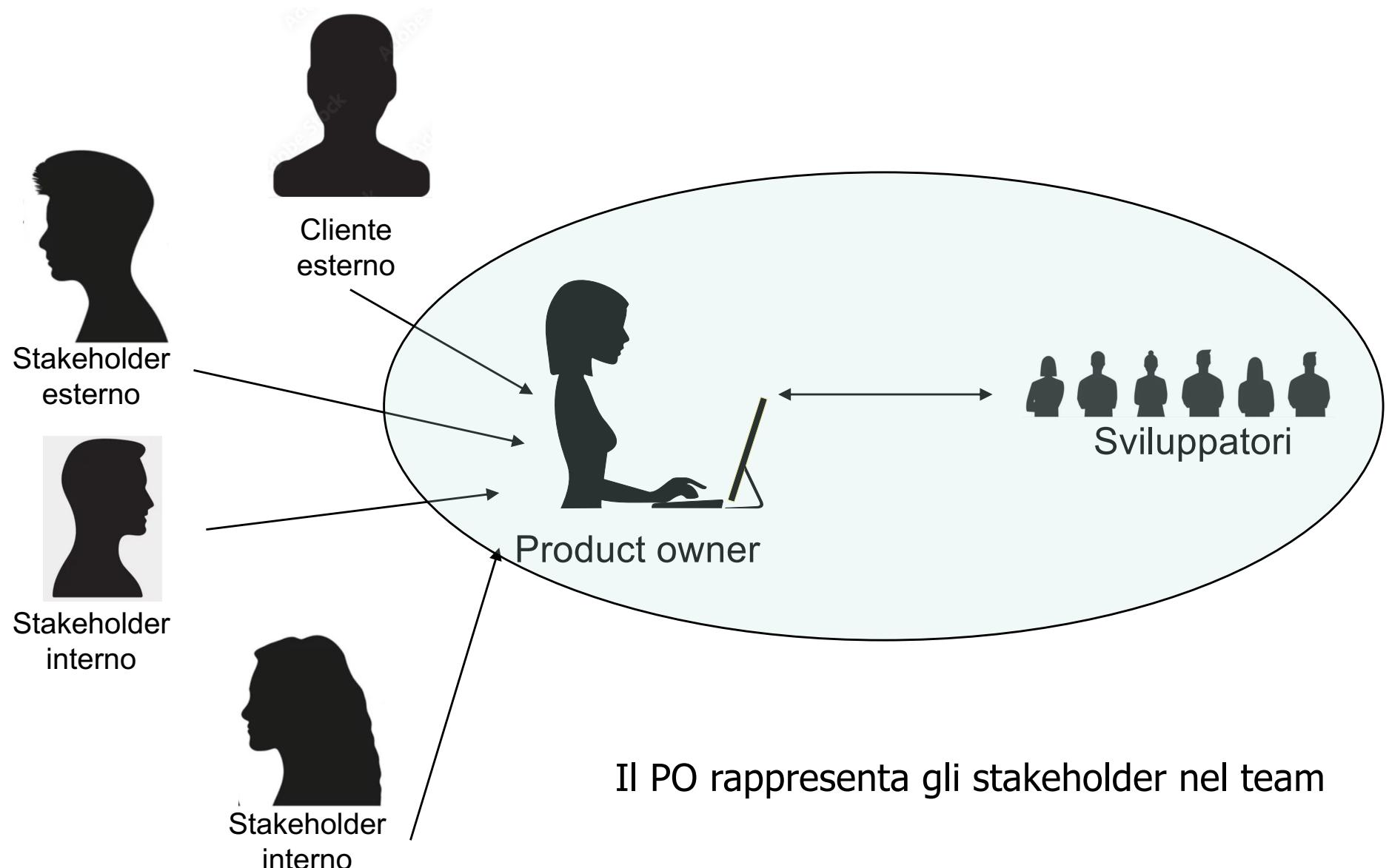


*Corso di Ingegneria del software
Università di Bologna*

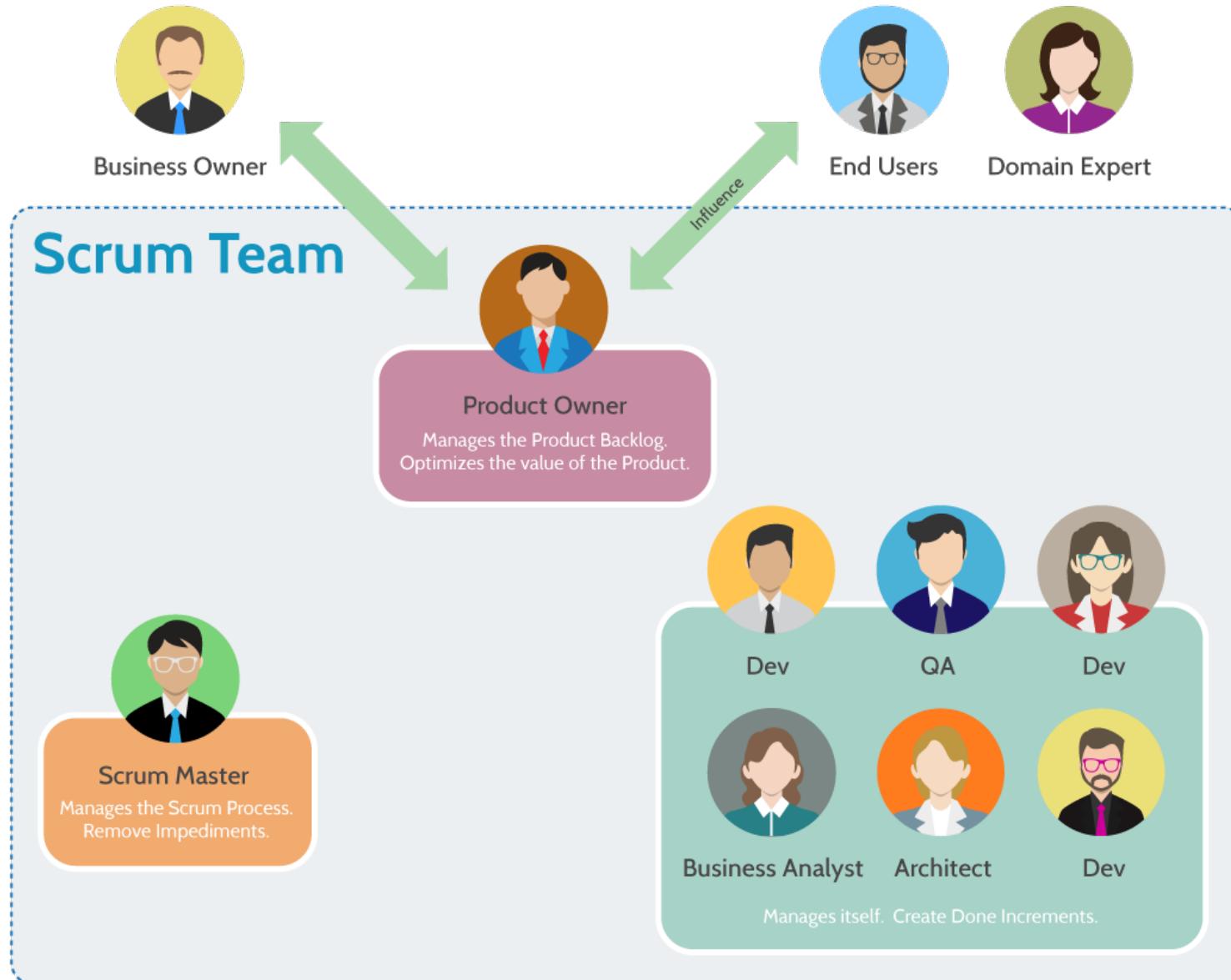
Agenda

- Il Product Owner: chi è e cosa fa
- Compiti del PO in Scrum
- Definire le priorità
 - Gioco: Desert survival
- Carte Essence del Product Ownership

Il PO, gli stakeholder e il team



Il PO è un membro del team



Le competenze del PO

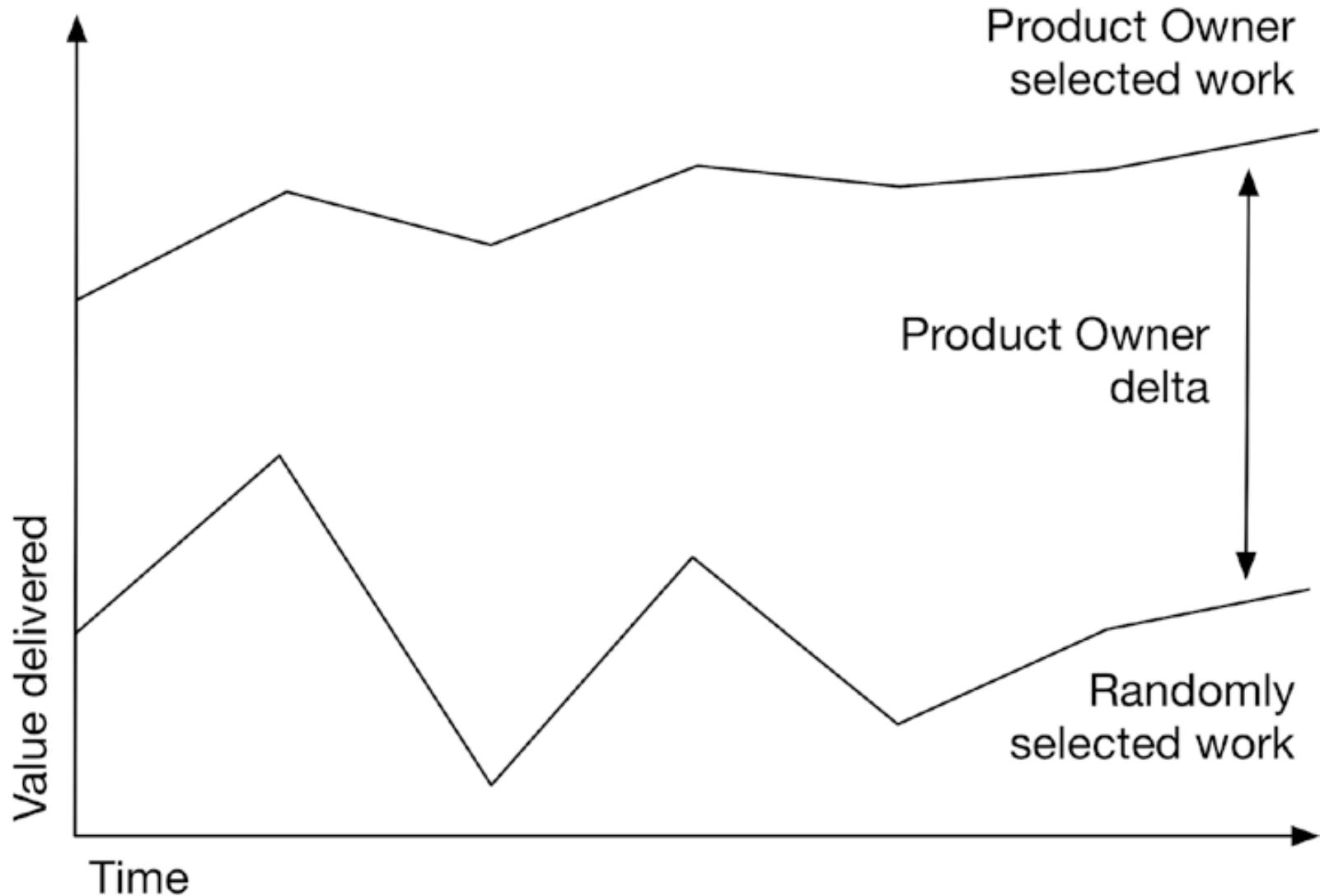
Le principali competenze del PO sono:

- Vedere il prodotto dal punto di vista del cliente e comprenderne le esigenze.
- Stabilire le priorità e decidere quali funzionalità creare e quali ritardare o eliminare (questo si chiama: *massimizzare il valore dei rilasci del prodotto*)
- Il PO rappresenta il cliente nel team, e riporta verso il cliente le domande del team cui non sa rispondere
- Mediare tra gli stakeholder e i membri del team.
- Avere flessibilità e adattarsi al cambiamento del mercato e del prodotto
- Raccogliere, analizzare e valutare i dati sul prodotto che viene usato, per migliorarlo alle iterazioni successive

Il PO massimizza il valore

- Il PO è responsabile di *massimizzare il valore* del prodotto risultante dal lavoro svolto dal Team
- Questa “massimizzazione” si intende dal punto di vista degli stakeholder, in particolare spesso si cerca di migliorare l’attrattiva del prodotto per gli utenti
- Come questo venga fatto può dipendere molto dall’organizzazione, dal Team e dagli stakeholder individuali coinvolti nel progetto di costruzione del prodotto

Massimizzare il valore



Il PO è responsabile del backlog

- Il PO è il responsabile della gestione del prodotto, incluso:
 - definire e comunicare al team il Product Goal;
 - creare gli elementi del Product Backlog e assicurarsi che siano visibili e comprensibili a tutti
 - ordinare (mettere in priorità) gli elementi del Product Backlog;
 - definire lo sprint goal
 - aiutare il team a definire lo Sprint Backlog
 - Concordare col team la Definition of Done
 - decidere i rilasci
- Il PO può fare tutte le attività sopra indicate oppure può delegarne qualcuna ad altri. Tuttavia, anche se delega il PO rimane il responsabile

Il PO nella guida Scrum

- Affinché il PO possa agire con successo all'interno dell'organizzazione, tutti devono rispettare le sue decisioni.
- Queste decisioni sono visibili nei contenuti e nell'ordine delle priorità del Product Backlog, e attraverso l'ispezione dell'Incremento durante la Sprint Review.
- Il PO è una singola persona e non un Gruppo.
- Il PO può esprimere nel Product Backlog i bisogni di diversi stakeholder
- Coloro che volessero modificare il Product Backlog possono farlo provando a convincere il PO.

Compiti del Product Owner

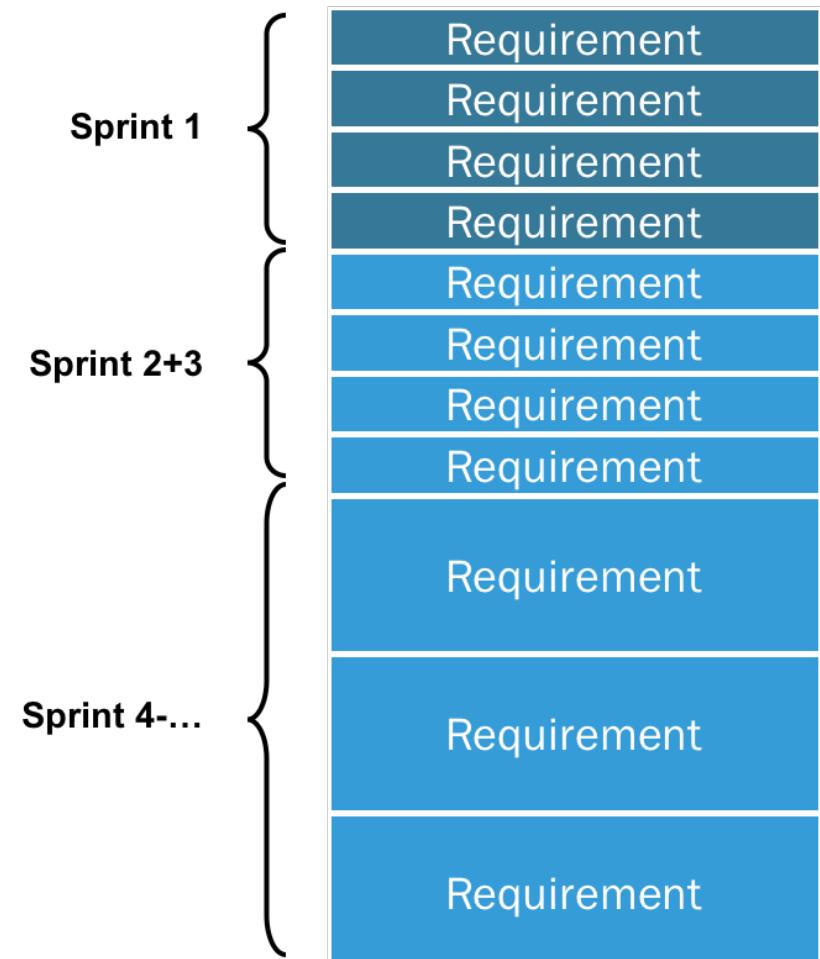
- Crea la visione del prodotto: un solo PO per prodotto
- Rappresenta gli stakeholder nel team
- Crea il backlog di prodotto
- Massimizza il valore di ciò che viene prodotto, sprint dopo sprint
- Definisce le priorità funzionali e la roadmap di prodotto
- Per ogni sprint definisce lo sprint goal
- Costruisce con l'aiuto del team le “Definition of Ready” e “Definition of Done”
- Rivede il prodotto assieme al team (nel meeting detto: *demo review*) e accetta o respinge il lavoro svolto

Il backlog di prodotto è una lista di storie ordinate dal PO e stimate dal team

ToDo List			
Story	Estimation	Priority	
As a user I want to be able to reset my password	1		1
As a user I want to edit items	3		2
As a user I want to export data	2		3
As an administrator I want to define KPI's for my sales team	4		4
As a user I want to view my data on mobile	5		5
As an administrator I want to send alerts when new leads come in	2		6
As a user I want to create a report of my data	5		7
As a user I want to update my reminder settings when a date is added	3		8
As a user I want filtering enhancements	4		9
As an administrator I want to configure views of data	5		10
Total	34		

Il backlog di prodotto è ordinato

- L'ordinamento (priorità) delle user story è compito del PO
- La suddivisione di quali storie verranno realizzate nel prossimo sprint è compito del team, sulla base delle stime di sviluppo
- (le user story sono requisiti)



Esercizio: lost in desert!

- Sono circa le 10:00 di metà luglio e sei appena atterrato nel deserto vicino al confine tra Messico e Stati Uniti.
- L'aereo è completamente bruciato, rimane solo il telaio. Miracolosamente i 10 passeggeri sono illesi ma il pilota è morto.
- Il pilota prima di morire non è stato in grado di dire a nessuno quale sia la posizione attuale.
- Tuttavia, poco prima dell'incidente il pilota aveva detto che si è a circa 100 km fuori dalla rotta indicata nel piano di volo (cioè fuori strada rispetto alle ricerche di eventuali soccorritori).

Sopravvivere nel deserto

- Pochi istanti prima dello schianto, il pilota ha rilevato una posizione a circa 110 km a sud-est di un campo minerario. Il campo è dunque l'insediamento conosciuto più vicino, in una direzione nota
- L'area circostante la zona d'impatto è pianeggiante e, fatta eccezione per occasionali rovi e cactus, è piuttosto arida.
- Prima che l'aereo prendesse fuoco, il gruppo è riuscito a salvare i 10 tipi di oggetti descritti nella prossima slide.
- Il vostro compito è classificare questi 10 oggetti in base alla loro importanza per la sopravvivenza nel deserto.
- A coppie, classificate gli elementi iniziando da 1 per il più importante, fino a 10 per il meno importante. Preparatevi a giustificare le vostre decisioni!

I 10 oggetti salvati dall'incidente

Elemento	Priorità
Torcia	
Coltellino svizzero	
Impermeabile grande	
Bandage kit con garza	
Pistola carica calibro 45	
Paracadute bianco e rosso	
1 litro di acqua per persona	
Un paio di occhiali da sole per persona	
Un cappotto per persona	
Specchietto cosmetico	

La soluzione dell'esperto

1. **Lo specchio cosmetico** Riflette la luce solare ed è il modo più efficace per attirare l'attenzione sul gruppo. Un rapido salvataggio senza restare sotto il sole cocente è la massima aspirazione per la sopravvivenza del gruppo.
2. **Un soprabito per persona** Per tenersi al caldo durante la notte fredda.
3. **1 litro di acqua a persona** La disidratazione è peggio della mancanza di cibo.
4. **Torcia** Questa può essere usata per attirare l'attenzione sul gruppo durante la notte.
5. **Paracadute (rosso e bianco)** I sopravvissuti possono ripararsi sotto durante il giorno e usarlo per attirare l'attenzione sul gruppo dall'alto.
6. **Coltellino** Può aprire i cactus per cercare acqua; serve anche per tagliare tessuti
7. **Impermeabile in plastica (taglia grande)** Se allargato su una piccola conca, si formano goccioline d'acqua dopo la notte fredda, raccoglie acqua da bere!
8. **Pistola calibro .45 (carica)** può essere usata per attirare l'attenzione sul gruppo e spaventare gli animali indesiderati!.
9. **2 paia di occhiali da sole a persona** Aiutano a proteggere gli occhi dei sopravvissuti dall'intensa luce solare.
10. **Kit Bendaggio con garza** Utile, ma nessuno è rimasto ferito nell'incidente!

Punteggi

- 3 punti se classificato esattamente
- 2 punti se classificato distanza uno
- 1 punto se distanza due
- 0 punti per tutti gli altri posti

Come il PO organizza lo sprint planning

Preparare una sintesi dei bisogni degli stakeholder

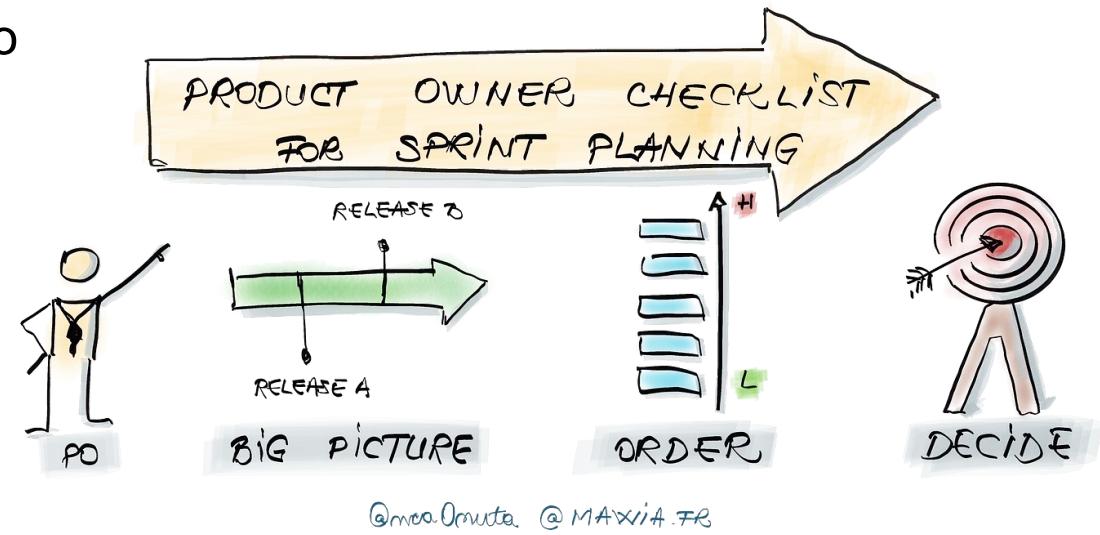
Selezionare e ordinare le User Stories proposte al prossimo sprint

Rivedere le US in modo che riportino tutte le informazioni necessarie: descrizione, criteri di accettazione, legame alla funzione da realizzare, dipendenze da altre US

Rivedere la Definition of Ready e assicurarsi che le US scelte per il prossimo sprint siano pronte

Preparare uno sprint goal e un'agenda per la sprint review

Aiuta il team a definire i task associati a ciascuna US



La definizione di Pronto (definition of Ready)

Una storia è Ready quando:

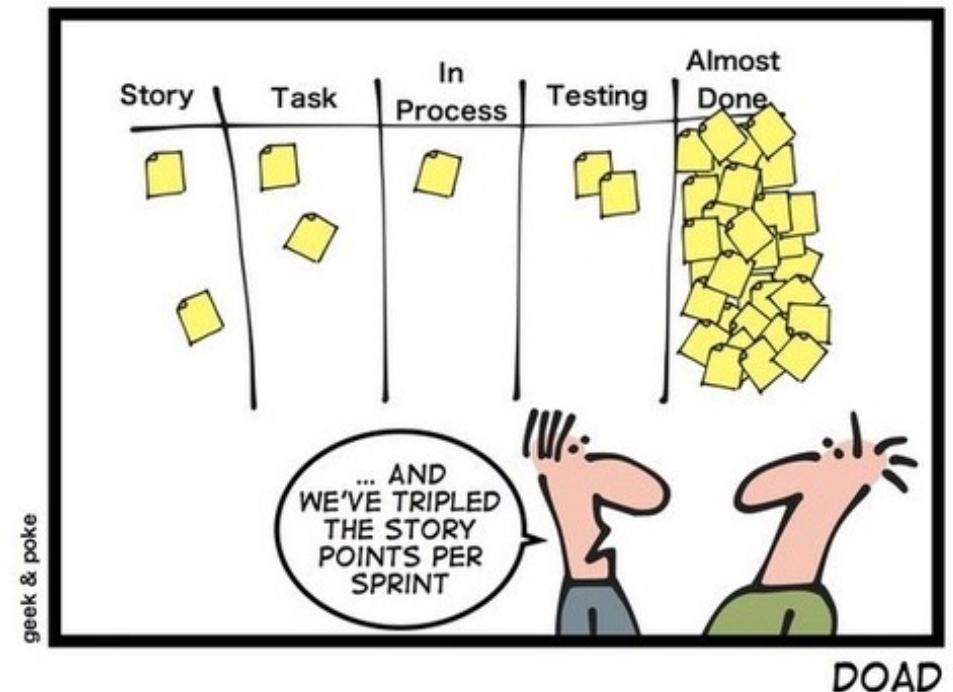
- Tutti i membri del team la capiscono
- È dettagliata abbastanza da poter essere poi testata
- Il PO deve comprenderne il valore per poterla ordinare in priorità con le altre
- Il Team dev'essere in grado di farne una stima in modo da completarla entro uno sprint
- La storia è indipendente dalle altre: il Team può lavorarla avendone il pieno controllo

La definizione di Fatto (Definition of Done)

La Definition of Done descrive lo stato dell'Increment quando questo soddisfa le metriche di qualità richieste per il prodotto.

La Definition of Done crea trasparenza fornendo a tutti una comprensione condivisa di quale lavoro è stato completato come parte dell'Increment.

Se un elemento non soddisfa la Definition of Done, non può essere rilasciato e nemmeno presentato durante la Sprint Review: ritorna nel Product Backlog



Esempio di definizione di “Fatto”

La lavorazione di una funzione che realizza una user story non è finita a meno che non soddisfi il PO: la definizione di “Fatto” è concordata dal Team col PO all'inizio del processo

Una possibile definizione di Fatto:

- i. Codifica della funzionalità richiesta: completata
- ii. Test di unità: scritti ed effettuati
- iii. Test di integrazione: superato
- iv. Test prestazionale: superato
- v. Documentazione (minimale): scritta
- vi. Approvata: dal PO

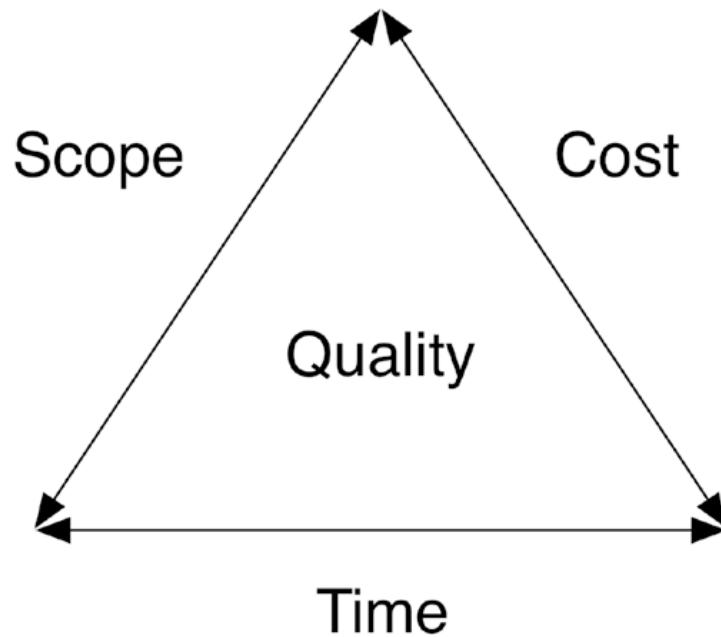
Tipi di Product Owner

- Amministratore del backlog
- Esperto del dominio
- Business analyst
- Gemello del Product Manager

Ogni organizzazione è diversa

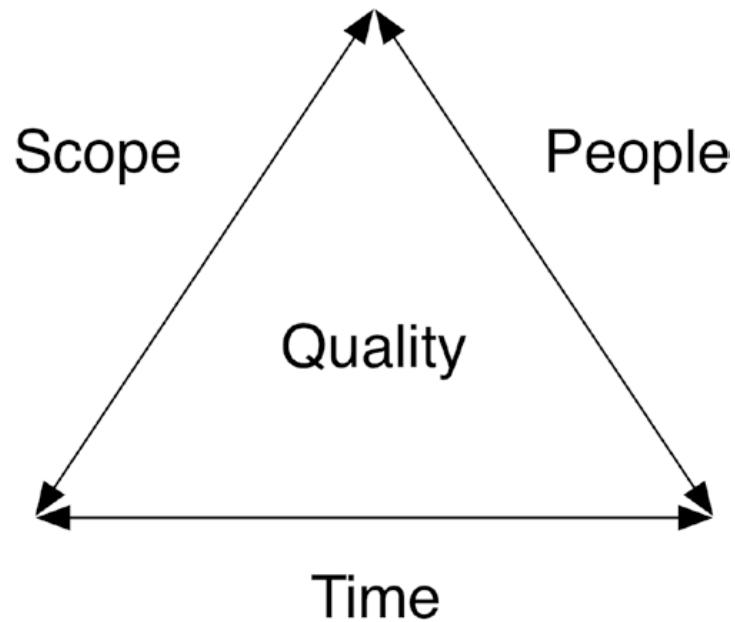
Il PO non lavora isolato, ma nel team

Il triangolo di ferro



- Lo *scope* è l'insieme delle funzioni richieste al prodotto (in forma di requisiti o di codice)
- Obiettivo del PO e del team è di costruire il prodotto restando nei vincoli di budget, soldi e tempo

Triangolo di ferro rivisto

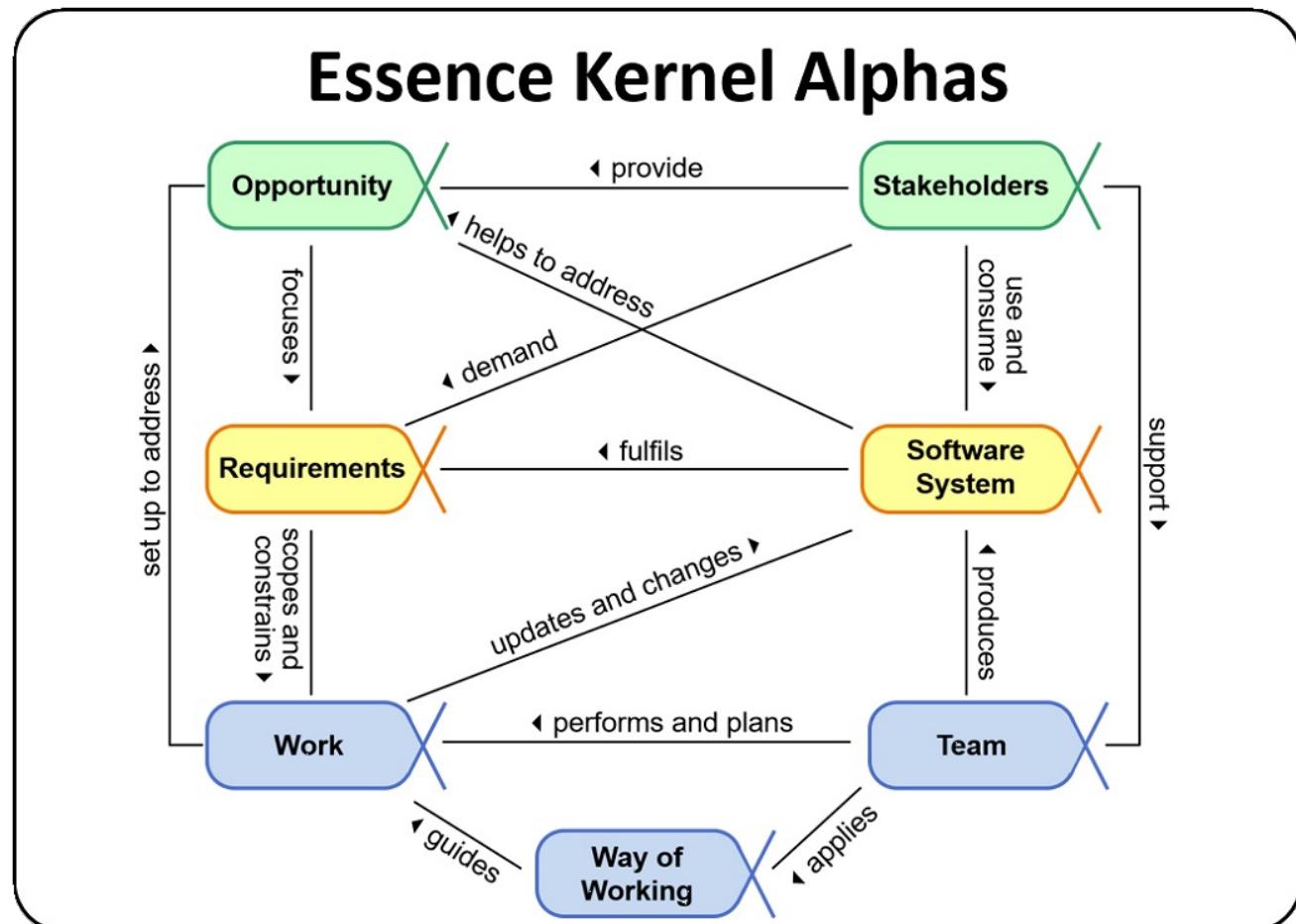


- Il costo del prodotto è una funzione dell'effort totale speso dalle persone coinvolte nella produzione
- $\text{Cost} = \text{People} \times \text{Time}$
- Obiettivo del PO è ottenere la migliore soluzione che ci si può permettere costruire col budget disponibile

Il PO in Essence

In questa figura

- Dov'è il cliente?
- Dov'è il PO?
- Dov'è Scrum?
- Dov'è il backlog?
- Dov'è il prodotto?





Stakeholders

The people, groups, or organizations who affect or are affected by a software system.

Recognized

Represented

Involved

In Agreement

Satisfied for Deployment

Satisfied in Use



Opportunity

The set of circumstances that makes it appropriate to develop or change a software system.

Identified

Solution Needed

Value Established

Viable

Addressed

Benefit Accrued





Business Case

The business case establishes the viability of producing a product to exploit an opportunity. It calculates the anticipated business value and compares this with the estimated solution cost.

Opportunity Understood

Value Established

Solution Options Costed

Full ROI Analysis Documented

Describes:



Business Case

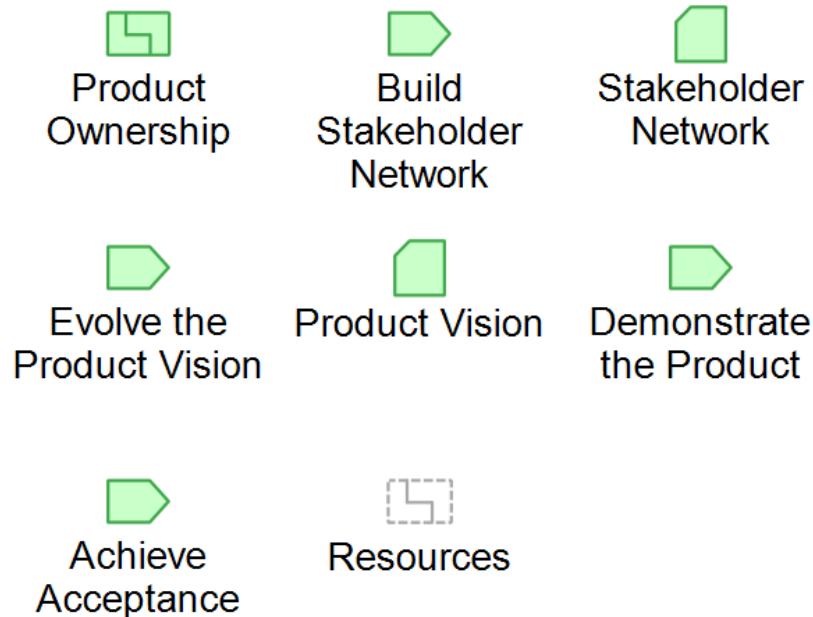
Opportunity Understood

- The underlying problems to be addressed have been identified and agreed
- The underlying opportunities to be exploited have been outlined and agreed.



Product Ownership Essentials

Own, evolve and communicate the vision, and guide the evolution of the product to achieve the vision.



Product Ownership

A single point of ownership for a product that provides rapid, empowered decisions and



dispute arbitration regarding what should be built into the product.

Having one person play this role (the “Product Owner”) provides the responsiveness needed for agile delivery, but this person must be able to represent all stakeholders and will need support from others in the team.

Owns: Product Vision

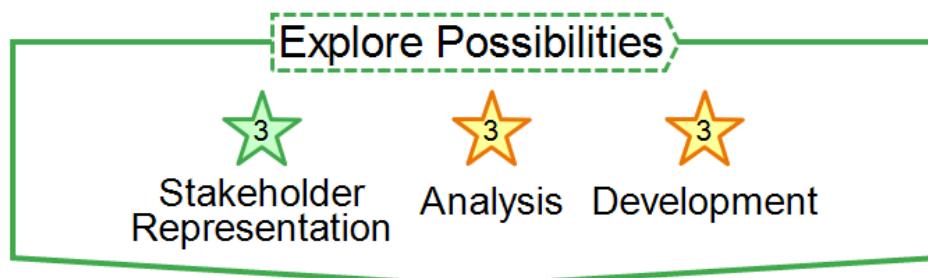
Represents: Stakeholders

Ref: Product Ownership



Evolve the Product Vision

Agree and communicate the goals and return-on-investment case for the product to drive and inform ongoing decisions about the product.



Opportunity: Value Established

Product Vision: Value Release Strategy Outlined



Product Vision

Communicates what is ultimately wanted or needed from the product, as well as how value will be progressively realized.



Need Identified

Solution Envisaged

Value Release Strategy Outlined

ROI Projected

Describes: Opportunity

Ref: Product Vision





Demonstrate the Product

Show the evolving product to stakeholders and elicit feedback as frequently as possible to converge on an optimal solution.

Stakeholders: Involved

Opportunity: Value Established

Understand Stakeholder Needs



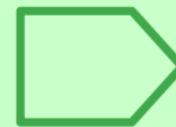
Stakeholder Representation



Testing

Stakeholders: In Agreement (contributes to)

Opportunity: Viable (contributes to)



Achieve Acceptance

The product is accepted for release. Progressively accepting the product enables frequent releases to be made to maximize return-on-investment.

Stakeholders: In Agreement

Opportunity: Viable

Ensure Stakeholder Satisfaction



Stakeholder Representation



Testing

Stakeholders: Satisfied for Deployment

Opportunity: Addressed





Requirements

What the software system must do to address the opportunity and satisfy the stakeholders.

Conceived

Bounded

Coherent

Acceptable

Addressed

Fulfilled



Requirements

Conceived

- Stakeholders agree system is to be produced
- Users identified
- Funding stakeholders identified
- Opportunity clear

1 / 6



Requirements

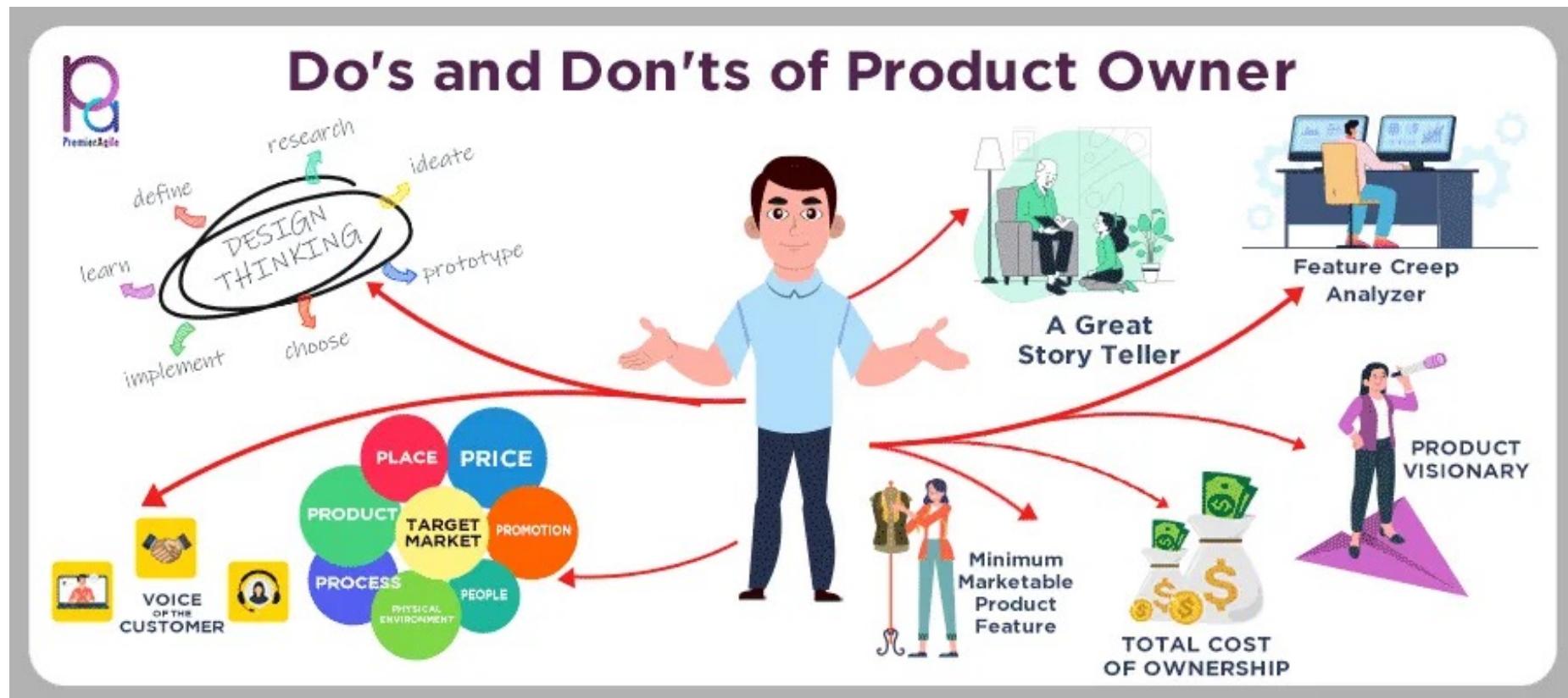
Bounded

- Development stakeholders identified
- System purpose agreed
- System success clear
- Shared solution understanding exists
- Requirement's format agreed
- Requirements management in place
- Prioritization scheme clear
- Constraints identified & considered
- Assumptions clear

2 / 6



Conclusioni



<https://premieragile.com/dos-and-donts-of-product-owner/>

	Product Manager	Product Owner	Program Manager	Project Manager
Owns	Product	Feature(s)	Service or Operation	Project
Planning	Product Roadmap	Product Backlog	Release Plan (Gantt Chart)	Project Plan (Gantt Chart)
Focus	Product-Market Fit	Requirements Definition	Efficient Delivery Pipeline	Coordination
Process	Lean	Agile	Agile	Waterfall

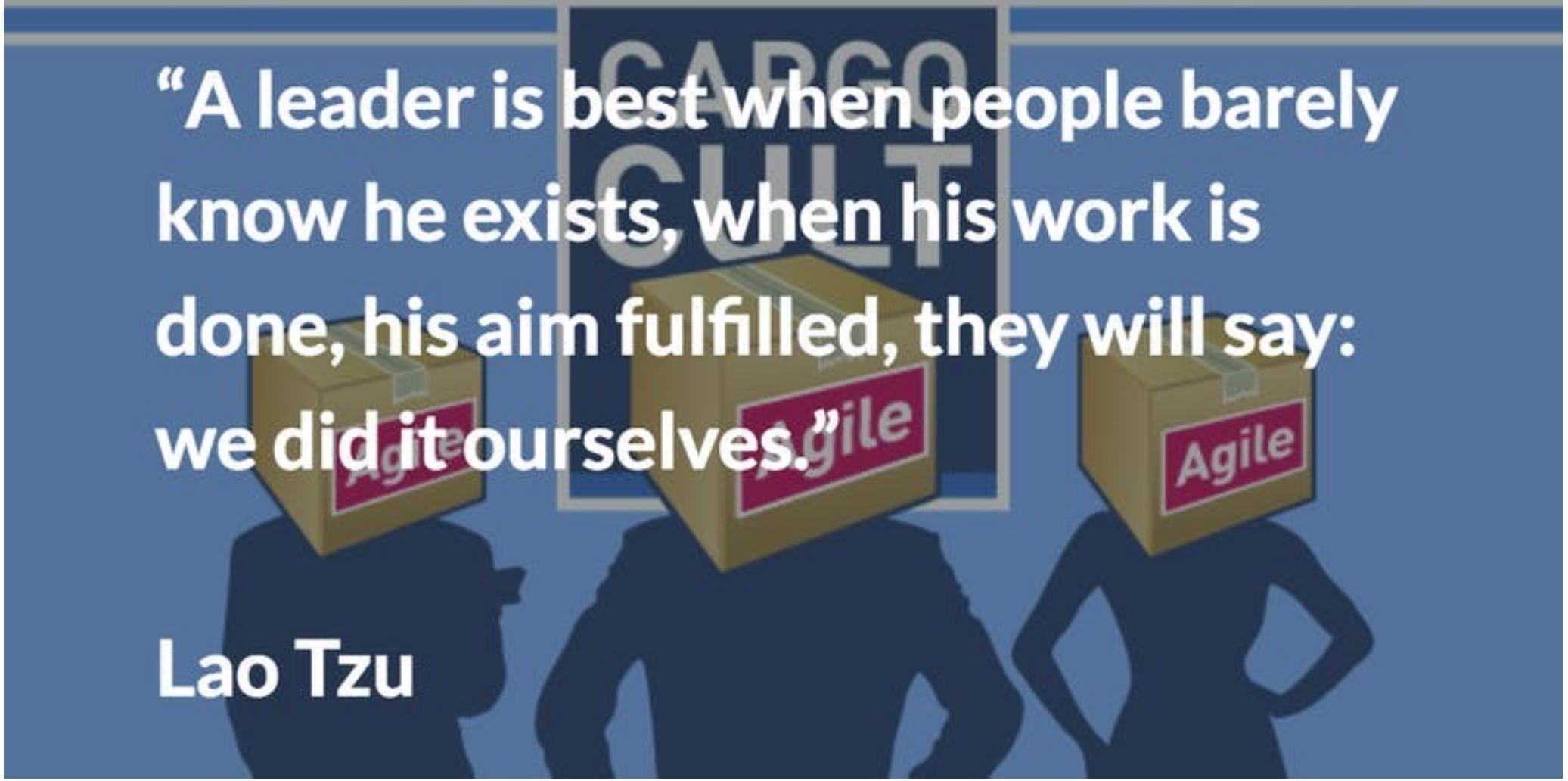
©Cabage 2019

<https://nealcabbage.com/product-vs-project-vs-program-management/>

Riferimenti

- Schwaber e Sutherland, La guida Scrum, 2020
- Kelly, *The art of Agile product ownership*, Apress, 2019
- Burri, *Educare un Product Owner*, tesi di laurea, Univ di Bologna, 2023
- Essence <https://practicelibrary.ivarjacobson.com/start>
(richiede registrazione registrazione gratuita)

Domande?



“A leader is best when people barely know he exists, when his work is done, his aim fulfilled, they will say: we did it ourselves.”

Lao Tzu

Il backlog di prodotto



*Corso di Ingegneria del software
Università di Bologna*

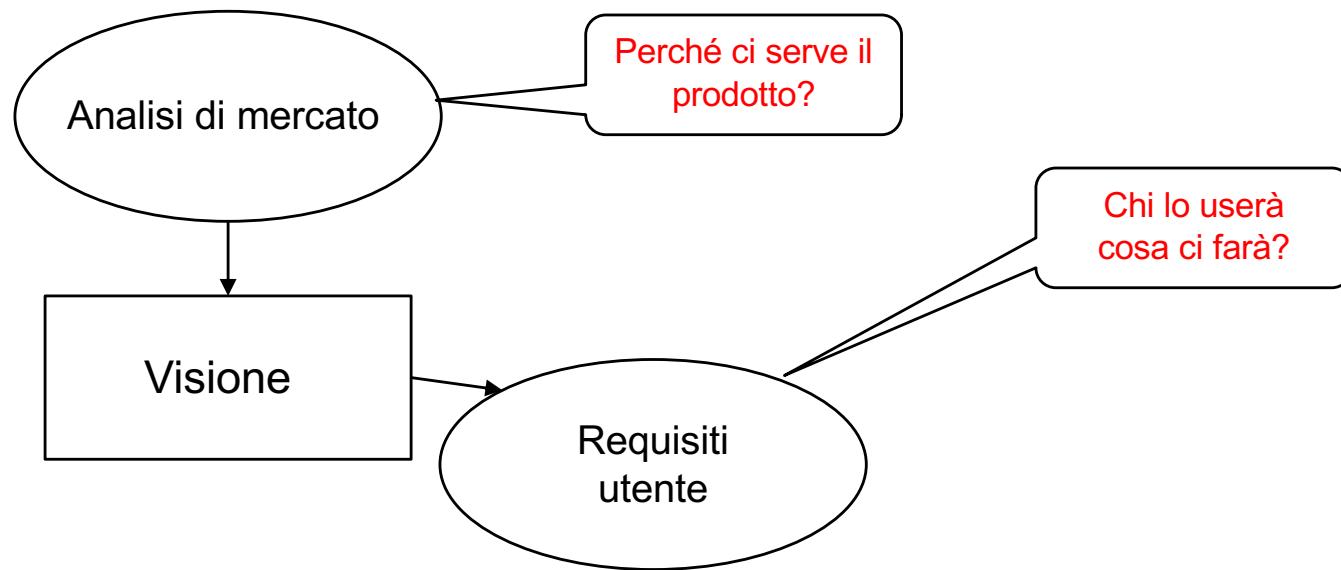
Obiettivi della lezione

- Cosa sono i requisiti di un prodotto?
- Analisi e classificazione degli elementi del product backlog
- «grooming» del product backlog
- Le stime: poker
- Gioco: PO value game
- Carte Essence

Come nasce un prodotto digitale

1. Ricerca di mercato: la prima fase prevede una serie di ricerche di mercato per comprendere le esigenze, i trend del mercato, la concorrenza, le opportunità e le sfide del settore.
2. Ideazione del prodotto: sulla base delle informazioni raccolte nella fase di ricerca di mercato, si inizia a ideare il prodotto digitale, definendone le caratteristiche principali, il target di riferimento e il valore unico che può offrire: si crive il documento «**Visione del prodotto**»
3. Specifica del prodotto: la fase di specifica del prodotto prevede la definizione di un piano dettagliato che specifica la funzionalità del prodotto, il design dell'interfaccia utente e l'esperienza utente: sono i **requisiti** del prodotto

Come nasce un prodotto digitale



Esempio 1

- Il mercato chiede una nuova app per gestire guardaroba personali
 - Consiglio di acquisti
 - Consiglio di abbinamenti
 - Ricerca di prezzi vantaggiosi
 - Statistiche sui vestiti che uso di più o di meno

Esempio2

- Guardando trasmissioni TV come L'Eredità oppure Pechino Express oppure VivaRai2, voglio:
 - raccogliere in tempo reale i tweet di altri spettatori,
 - partecipare con i miei tweet
 - mandare schermate di visual analytics sui tweet raccolti

La comunicazione e gli stakeholders

- La definizione dei requisiti di un prodotto digitale è un problema di comunicazione
- Quelli che vogliono il nuovo prodotto devono comunicare con quelli che lo costruiranno
- **Se una delle due parti prevale sull'altra, il progetto fallirà**
 - Chi vuole il prodotto avrà vincoli di budget e di tempo, e chiederà funzionalità senza tener conto dei limiti degli sviluppatori
 - Chi sviluppa usa un linguaggio tecnico di difficile comprensione per gli stakeholders, e se non ascoltano perderanno l'opportunità di imparare ciò che è necessario

Requisiti

- Quando qualcuno chiede a qualcun altro di creare un **prodotto** software occorre stabilire i *requisiti* di ciò che verrà costruito
- I requisiti saranno la base del progetto, della codifica, del testing/verifica e della validazione

Esempio di backlog: app per cercare lavoro

ID	User story	User task	activity
0	As a job seeker, I want to search jobs with basic criteria, so that I can find a suitable job.	Browse jobs	Find job
1	As a job seeker, I want to select desired working location, so that I can save transportation costs.	Browse jobs	Find job
2	As a job seeker, I want to upload resume in PDF, so that others can view my resume online.	Post resume	Find job
3	As a job seeker, I want to upload resume in Word, so that others can give me some advice and help me to amend it.	Post resume	Find job

Esercizio

- Scrivete 4 US per una to-do app
- è un'app che consente di pianificare e organizzare le attività, gli eventi, le idee e altro ancora
- Es.: Trello, MS To Do

Per una to-do app, scrivi 4 user story (principiante vs professionista)

- 1. Come utente, voglio poter aggiungere compiti alla mia lista To-Do, in modo da poter tenerli a mente e non dimenticarli.
- 2. Come utente, voglio essere in grado di ordinare i miei compiti in base alla loro importanza e scadenza, così da poter gestire meglio le mie attività.
- 3. Come utente, voglio ricevere notifiche quando scade un compito, così da poter ricordare di svolgerlo in tempo.
- 4. Come utente, voglio condividere la mia lista To-Do con altre persone, per condividerla.

- 1. Come utente, voglio essere in grado di creare e salvare nuovi compiti nella mia lista To-Do, in modo da mantenere un elenco aggiornato delle cose da fare.
- 2. Come utente, voglio ordinare e filtrare la mia lista To-Do per priorità, scadenze e altre categorie rilevanti, per organizzare i miei compiti in modo efficiente e gestirli in base alla mia disponibilità di tempo.
- 3. Come utente, voglio impostare promemoria e notifiche per i miei compiti, in modo che possa essere avvertito in tempo utile delle scadenze importanti e delle attività da svolgere.
- 4. Come utente, voglio condividere i miei compiti con altri utenti o gruppi di utenti, per collaborare su progetti o assegnare attività a specifici membri del team.

Cosa c'è nel backlog?

Customer Side

Quello che crea
valore

Quello che devi
scoprire per
creare valore

Tech Side

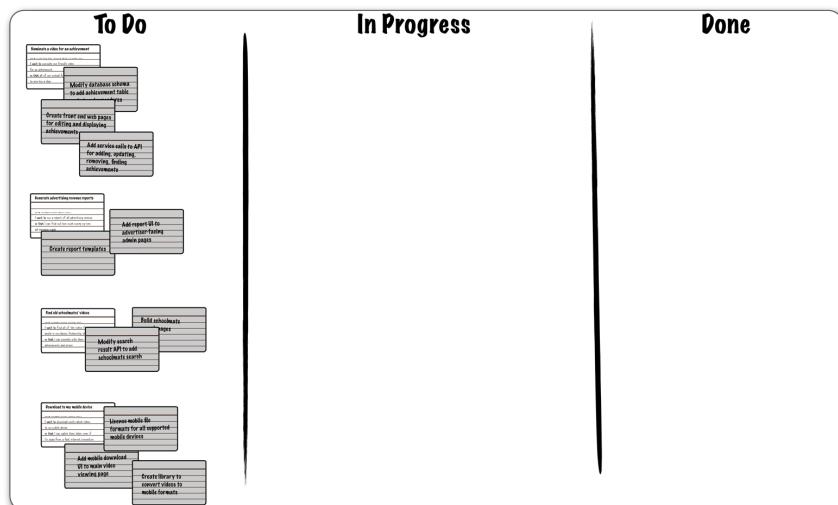


Quello che
impedisce la
creazione di valore

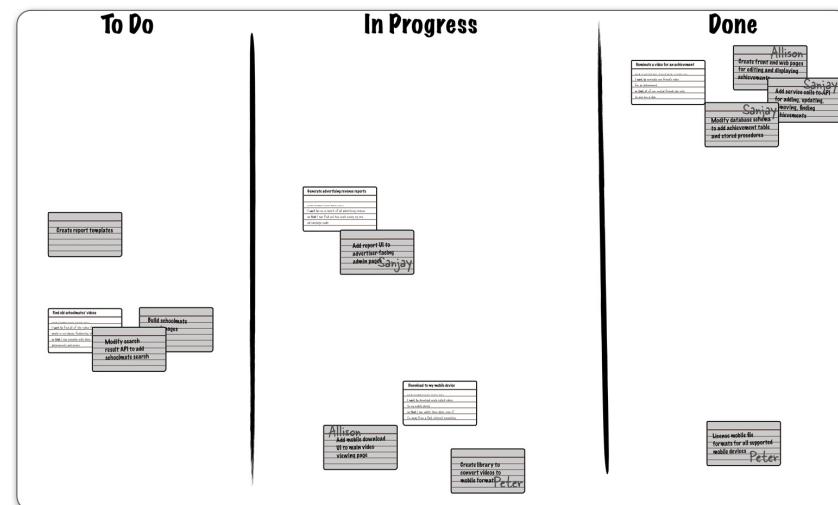
Quello che devi fare
affinché il valore sia
generato (in modo più
“semplice”).

Evoluzione del backlog

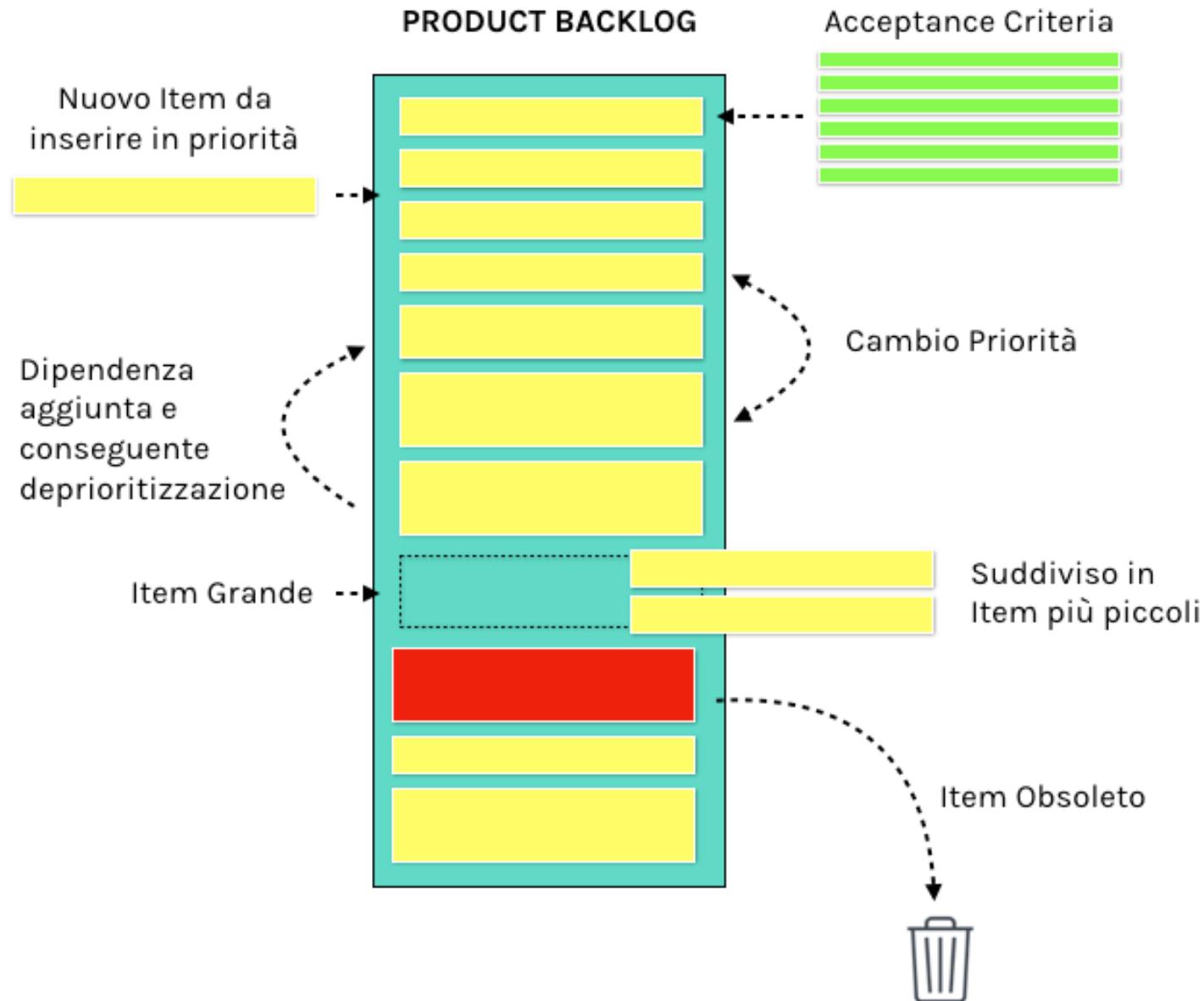
Non esiste una forma standard di rappresentazione del backlog (es. alcuni usano Excel, altri post-it sul muro)



Stato iniziale



Stato durante uno sprint



Esempio di backlog

A	B	C	D	E	F	G	H
#ID	Item	Priorità	Categoria	Dependencies	SP	Team	Acceptance Criteria / Note
231	Come Autore, Voglio selezionare i permessi per i singoli utenti, perché voglio avere il controllo.	1	User Story		20	Batman	<ul style="list-style-type: none"> - Esistono 3 tipi di permessi: View, Edit, Comment. - View può solo vedere la presentazione. - Edit può modificare. - Comment può aggiungere dei commenti in delle sezioni specifiche. - Ogni utente può avere solo un permesso. - Quando revoco un permesso comment o edit le modifiche effettuate restano valide - Quando un utente con un permesso "rimosso" si logga nuovamente riceverà il messaggio di errore "non puoi più apportare modifiche al documento, rivolgiti all'amministratore per qualsiasi dubbio" <p>E' una Epica > Splittare e nuova estimation prima di prossima Sprint</p>
112	Come Autore, Voglio inserire l'indirizzo email dei miei contatti, Per poter condividere la presentazione.	2	User Story	#231	8	Catwoman	<ul style="list-style-type: none"> - Se l'indirizzo è sbagliato inviare notifica via mail a chi ha inviato l'invito - Inserire Max 5 contatti
156	Come Autore, Voglio poter condividere la presentazione anche se non ho l'indirizzo email, così da poterla condividere più velocemente	3	User Story	#231	13	Batman	
97	Date e orari sbagliati sulla gestione delle versioni modificate.	2	Bug		4	Hulk	Quando apro la cronologia delle modifiche Date e Orari non sono ordinati i sono sbagliati
29	Intervistare 6 Commerciali di SME per esplorare funzionalità Analytics (#7)	4	Discovery		13	Catwoman	
72	Come Autore voglio utilizzare le GIF per rendere la presentazione più efficace e divertente	5	Epic			Batman	
100	Creare prototipo navigabile (su Invision) Dashboard Analytics in base a esito interviste per testarlo su 25 utenti	2	Discovery	#29	13	Catwoman	Rispetto alle interviste sembra essere un'ottima Biz Opportunity. Gli utenti intervistati sarebbero disposti a pagare una extra fee per il servizio

Il metodo MOSCOW per mettere in priorità le US

MOSCOW : acronimo per Must/Should/Could/Won't
(DEVE / DOVREBBE / POTREBBE / NO)

- Must: funzioni che DEBBONO esserci nel prodotto
- Should: funzioni che DOVREBBERO esserci
- Could: funzioni che POTREBBERO esserci
- Wont: funzioni che NON INSERIREMO nella versione attuale

US di una to-do app

- Come utente, voglio aggiungere nuovi compiti alla mia lista delle cose da fare, in modo da poter tener traccia di ciò che devo fare.
- Come utente, voglio contrassegnare un compito come completato, in modo da tener traccia dei compiti che ho svolto.
- Come utente, vorrei organizzare i compiti in liste diverse, in modo da poter gestire meglio le diverse aree della mia vita.
- Come utente, vorrei poter impostare scadenze per i compiti, in modo da essere avvisato quando devo completarli.
- Come utente, vorrei condividere una lista delle cose da fare con altre persone, in modo da collaborare su attività condivise.
- Come utente, vorrei giocare all'interno dell'app

US di una to-do app

MUST HAVE

1. Come utente, voglio aggiungere nuovi compiti alla mia lista delle cose da fare, in modo da poter tener traccia di ciò che devo fare.
2. Come utente, voglio contrassegnare un compito come completato, in modo da tener traccia dei compiti che ho svolto.

SHOULD HAVE

3. Come utente, vorrei organizzare i compiti in liste diverse, in modo da poter gestire meglio le diverse aree della mia vita.
4. Come utente, vorrei poter impostare scadenze per i compiti, in modo da essere avvisato quando devo completarli.

COULD HAVE

5. Come utente, vorrei condividere una lista delle cose da fare con altre persone, in modo da collaborare su attività condivise.

WON'T HAVE

6. Come utente, vorrei giocare all'interno dell'app

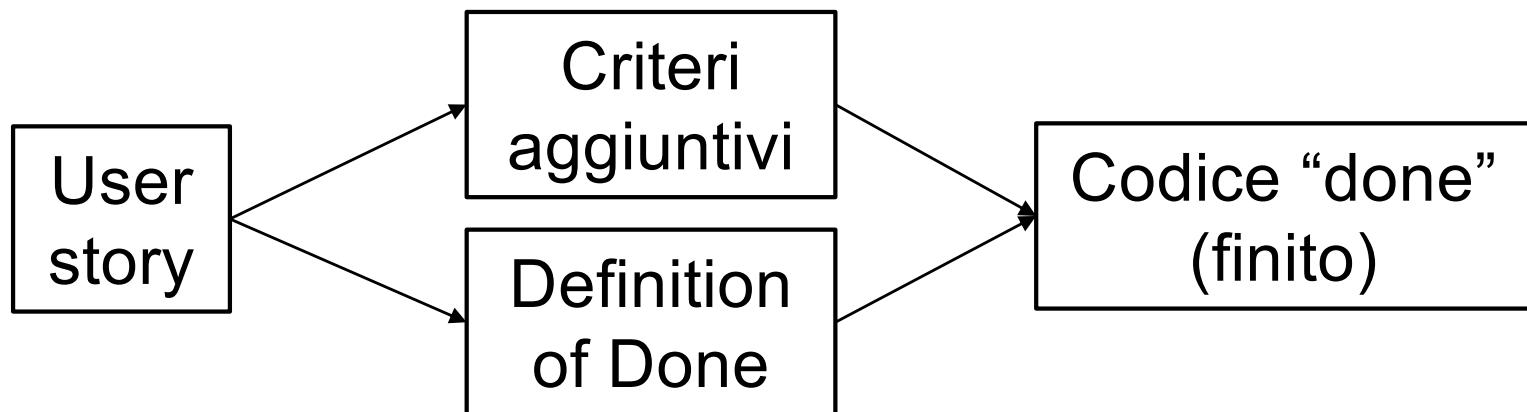
User story «completate»

Come <tipo_utente>

Voglio poter <fare qualcosa>

Perché/affinché <ottengo valore>

- Criteri aggiuntivi (es. l'utente deve riempire tutti i campi del modulo)
- DoD: validazione finale (es. superati tutti i test di unità)



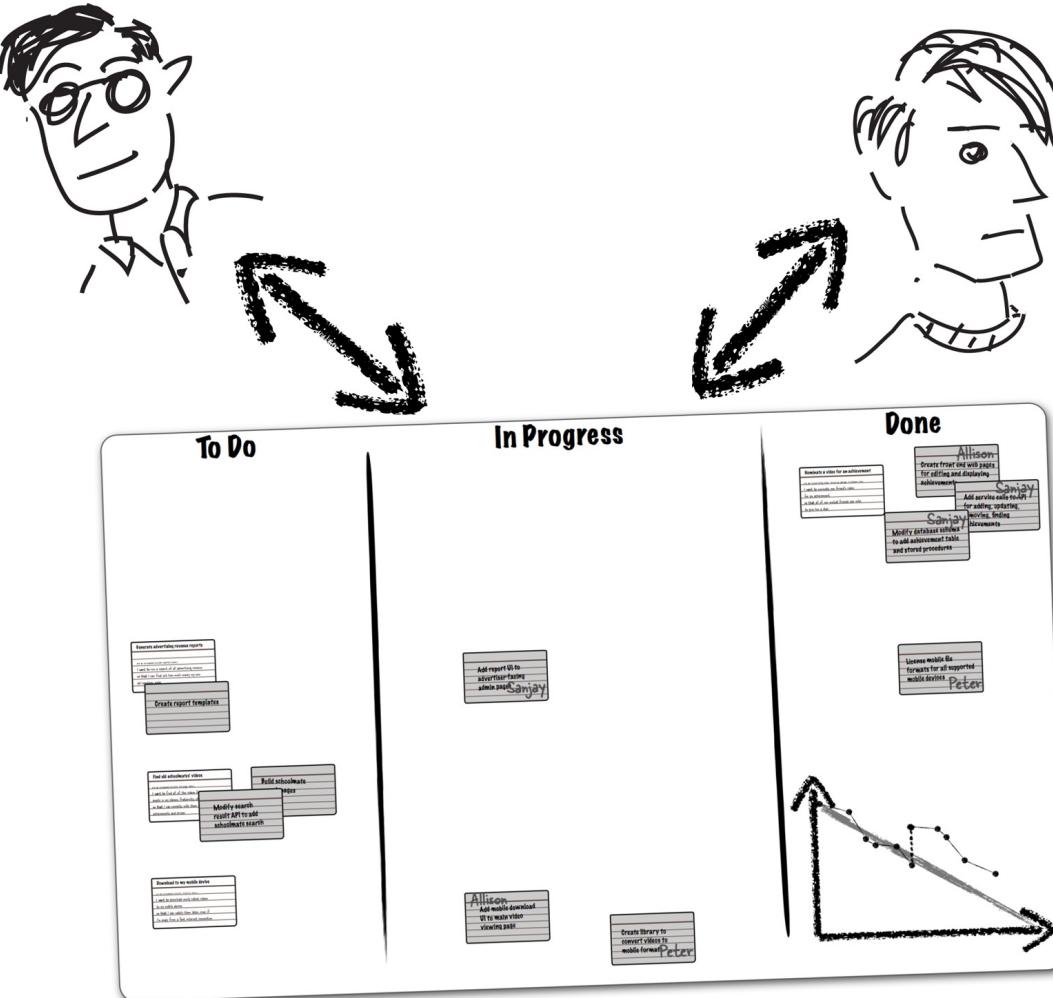
Aggiornare il backlog

- Il backlog contiene prima di tutto le US, ovvero il futuro del prodotto
- L'aggiornamento del backlog consiste nel modificarlo in modo da riflettere nuove informazioni ed esigenze che emergono nel tempo.
- Questo potrebbe comportare l'aggiunta di nuove US o la modifica delle priorità in base a nuove esigenze
- L'aggiornamento del backlog è quindi una pratica continua durante lo sviluppo

Aggiornamento del backlog

La divisione più semplice
del backlog è tra attività

- Da fare (il backlog)
- In corso
- Fatte



Stime

- Durante lo sprint planning occorre stimare lo sforzo necessario per le users' stories
- Tecnica del **planning poker**: i membri del team eseguono stime indipendenti su più turni; ad ogni turno si può cambiare la stima sulla base dei risultati del turno precedente, cercando il consenso (metodo Delphi)

Un mazzo per planning poker



Esempio

- User story: “qualcuno vuole gestire ordini d’acquisto usando un carrello elettronico”
- Team di sviluppo con 7 membri
- Primo turno di stime (valori history points):



Esempio (segue)



- I partecipanti P3 e P6 spiegano le loro stime
- Secondo turno:



Esempio (segue)

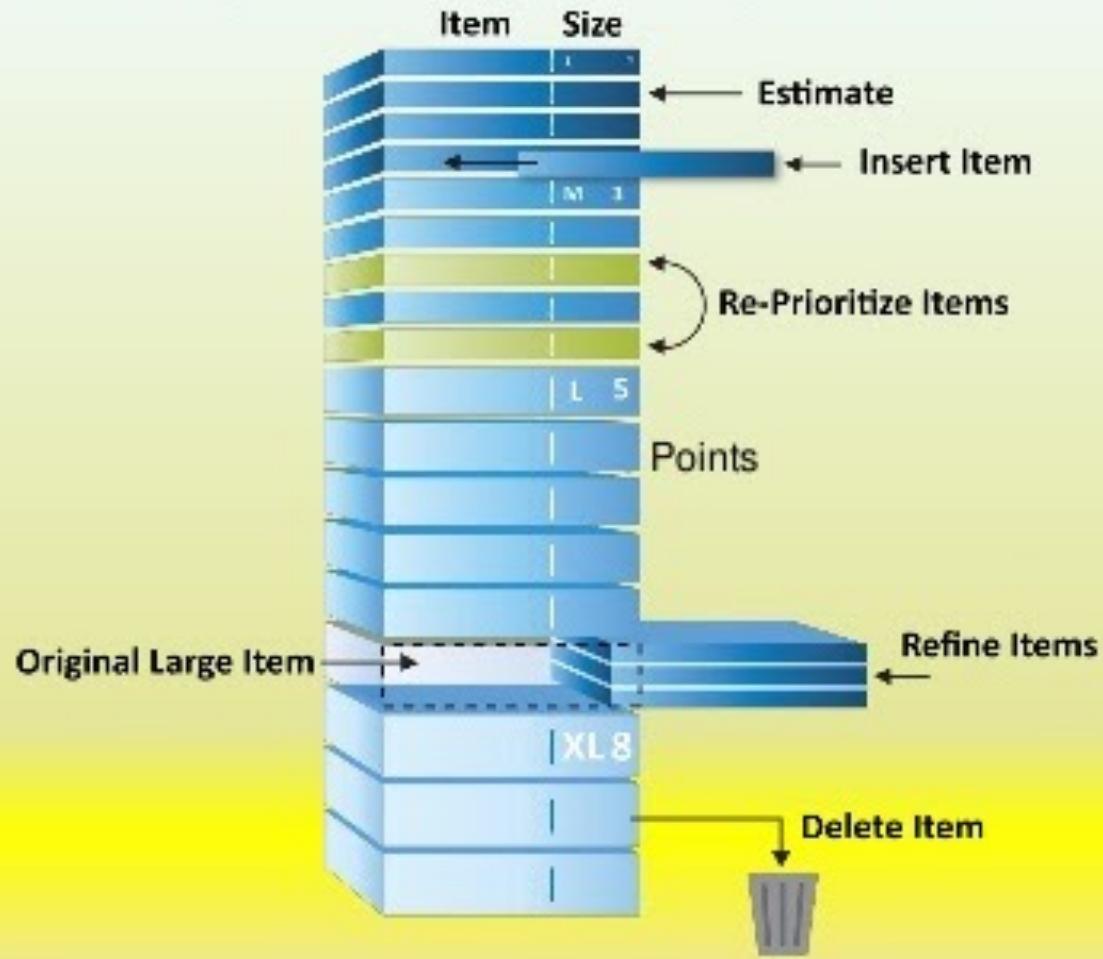


- Tutte le stime convergono eccetto la terza:
 - Altro turno, oppure
 - Usare 3 o 5 come stima

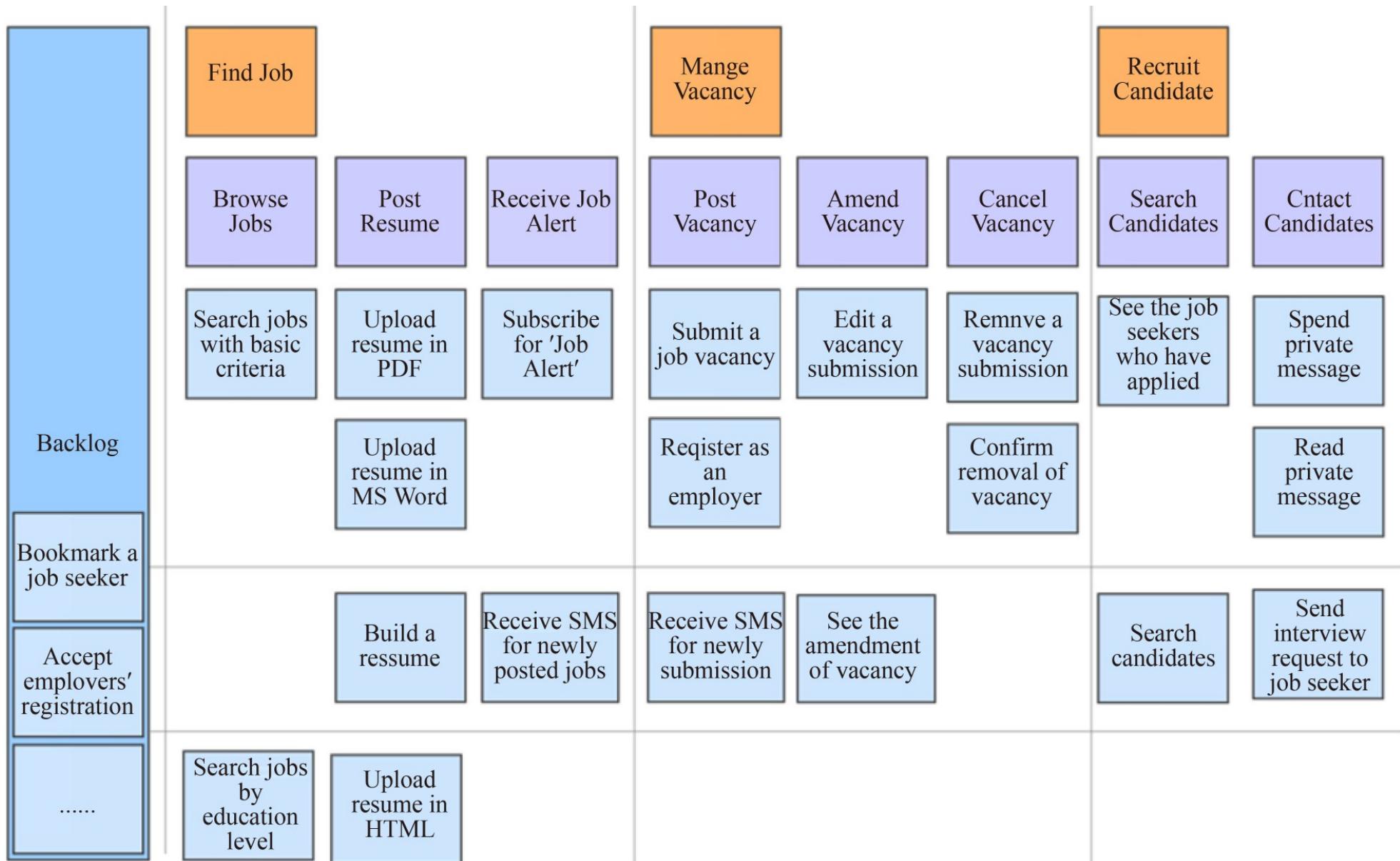
Backlog grooming

- Il *grooming* consiste nel riesaminare e nel riorganizzare il backlog per mantenere l'ordine e la priorità delle attività in modo da riflettere le esigenze degli stakeholder.
- Questa operazione può includere l'eliminazione di attività obsolete, la suddivisione di attività complesse in compiti più piccoli e la stima del tempo e delle risorse necessarie per ogni attività.
- Grooming viene eseguito a intervalli regolari (ad ogni inizio di sprint) per mantenere il backlog organizzato e gestibile

WHAT IS PRODUCT BACKLOG GROOMING?



User story mapping



Le pratiche correnti di creazione e analisi del product backlog

Name	Brief Description
Balanced teams	Balancing team composition with experts in business, product design, and software development.
Dual track agile	Organizing the work into two “tracks.” Track 1 typically includes research, negotiating with stakeholders, drawing user interface mockups, and writing user stories. Track 2 typically involves building, testing, architecting, refactoring, deploying, and maintaining the product.
Stakeholder mapping	Drawing a diagram of individuals who are interested in the success of the product.
Interviewing	Semi-structured discussions with stakeholders (e.g., users, product sponsor).
Persona modeling	Creating fictional users (character sketches) to reason about who will use product features.
Affinity mapping	Organizing data from user interviews or ideation sessions to generate insights.
Design studio	Converging on a product concept by iterating between generating and discussing design ideas.
Sketching / mockups	Drawing informal models of graphical user interfaces.
Usability testing / validation testing	Reviewing mockups with users.
Writing user stories	Writing brief, informal descriptions of some aspects of a software system.
Story showcase	Building a shared team understanding of upcoming user stories.
Backlog grooming	Refining and resequencing user stories.
Accepting stories	Evaluating delivered work.

Fonte: [Sedano, The Product Backlog, 2019](#)

Gioco: PO value game

- **PO value game** è un gioco che simula le scelte che deve effettuare un PO
- Vedere il sito del [gioco](#)

Product backlog

Product Backlog

An ordered list of things to build into the product to enhance its value.

Items Gathered

Items Prioritized

Cost-Benefit Quantified

Describes: Requirements

Ref: Agile Backlog and Product Backlog in Scrum

PB

IVAR JACOBSON INTERNATIONAL
Generated by UI Practice Workbench™

2018.09

Product Backlog

Items Gathered

- There is a list of things of value to build into the product
- The list is visible to the team and stakeholders
- The list is understandable by the team and the stakeholders

1 / 3

PB

IVAR JACOBSON INTERNATIONAL
Generated by UI Practice Workbench™

2018.09

Product Backlog

Items Prioritized

- The list of items is sorted in priority order from highest to lowest
- The priority ordering is communicated to the stakeholders and the team
- The stakeholders and the team accept the priorities as reasonable

2 / 3

PB

IVAR JACOBSON INTERNATIONAL
Generated by UI Practice Workbench™

2018.09

Product Backlog

Cost-Benefit Quantified

- The value of each item has been quantified at least relative to the other items
- The size / cost to implement of each item has been quantified at least relative to the other items
- The priority ordering reflects the relative value-for-money that is anticipated to result from implementing each item

3 / 3

PB

IVAR JACOBSON INTERNATIONAL
Generated by UI Practice Workbench™

2018.09



Product Vision

Communicates what is ultimately wanted or needed from the product, as well as how value will be progressively realized.



Need Identified

Solution Envisaged

Value Release Strategy Outlined

ROI Projected

Describes: ☐ Opportunity

Ref: [5] Product Vision



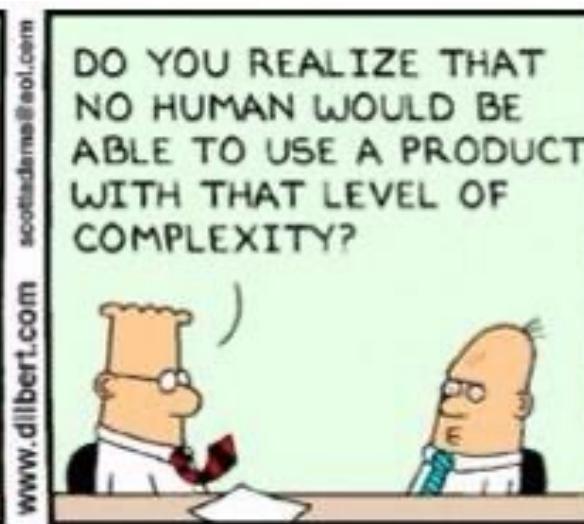
Riferimenti

- Stellman e Greene, *Learning Agile*, O'Reilly, 2015
- Sedano, Ralph, Perarire, The Product Backlog, Proc. 41° ICSE, 2019
- Cohn, *User stories applied*, Addison-Wesley 2004

Siti per planning poker

- <https://planningpokeronline.com>
- <https://www.planningpoker.com>

Domande?



Come scrivere le user stories



*Corso di Ingegneria del software
Università di Bologna*

Obiettivi della lezione

- Lavorare con le user story
 - Le user story sono frammenti di conversazioni
 - Le condizioni aggiuntive
 - L'indicazione dei test
 - Definition of Ready, Definition of Done
- Principi INVEST
- Metodo MoSCoW per mettere le US in priorità
- User story mapping
- Checklist carte Essence

Il PO elabora il product backlog

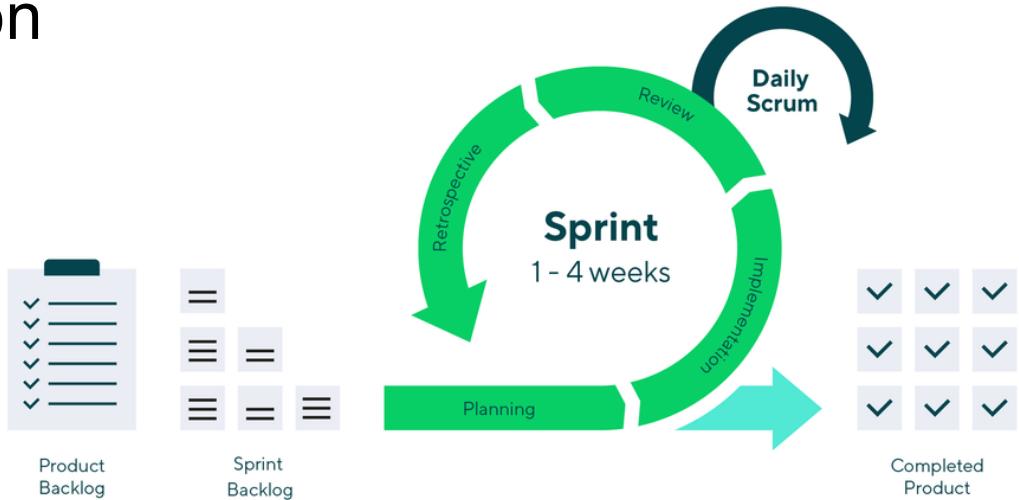


... e lo usa per collaborare col team

Quando?

Il PO elabora il backlog, con l'aiuto dei Devs:

- All'inizio dello sviluppo
- Durante lo sviluppo:
 - Durante lo sprint planning
 - Durante lo sprint review (demo)



A cosa servono le user story

- Il Product Owner scrive le storie nel backlog e collabora col team, per condividerne la comprensione e i criteri di accettazione
- Una storia è «Ready» se è sufficientemente piccola da poter essere realizzata in una o due settimane
- Il PO concorda col team la “Definition of Done”, cioè il criterio di accettazione di una storia lavorata «bene»
- Il team di sviluppo sceglie le storie da realizzare, definisce i test e prepara tutto il necessario perché il PO possa accettare il software risultante, così come richiesto dalla Definition of Done

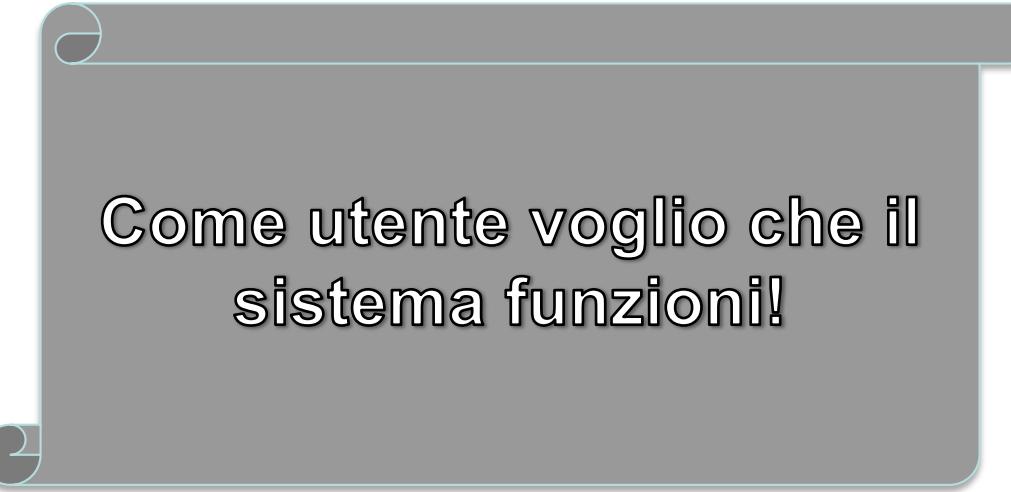
Come nascono le US

- Le US sono descrizioni in linguaggio naturale dei “*Requisiti funzionali*” di un prodotto o sistema software
- Le US sono semplici, non definitive e frutto di un dialogo a più riprese tra chi conosce l'esigenza e chi sa come risolverla con un prodotto software
- Le US devono riferirsi a specifici bisogni funzionali e non a enunciazioni di carattere generale (es.: «*voglio un prodotto che risolva i miei problemi di soldi*» **non va bene**）
- Le US possono essere raccolte o create mediante workshop dedicati, interrogando utenti e altri stakeholder

Le US sono frammenti di conversazioni

- Utente: come posso descrivere i miei bisogni/desideri?
- Stakeholder: come posso far sì che il prodotto abbia successo?
- PM: come traccio e pianifico questo compito?
- BA: quali sono i dettagli di questa feature?
- UX: quali sono i bisogni dell'utente?
- Developer: quali task devo eseguire oggi?
- QA: come posso validare questo task completato?

Livello di dettaglio delle U.S.



Come utente voglio che il
sistema funzioni!

- Storie di livello troppo generico tendono a perdere di significato e utilità.
- Storie con troppo contenuto “implicito” necessitano di spacchettamento e possono generare lavoro che richiede più “Sprint”

Attribuzione di storie ad un utente virtuale

Se molte storie si riferiscono ad un singolo profilo di utilizzatore può essere utile utilizzare una «*persona*», ovvero un simulacro di utente virtuale (una specie di «*avatar*») che aiuti nella creazione delle US

Esempio: in una app di gestione dell'albo comunale, analizzare le persone Sindaco, assessore, dirigente, impiegato, cittadino

Condizioni aggiuntive ad una US

Ogni storia dovrebbe essere accompagnata da una descrizione di condizioni aggiuntive e di come potrà essere considerata soddisfatta

Come maestro
elementare voglio
poter effettuare
l'appello sul
registro elettronico

Vedere l'elenco di
classe
Spuntare assenti e
presenti
Visualizzare
rapporto

Esempio con condizioni aggiuntive

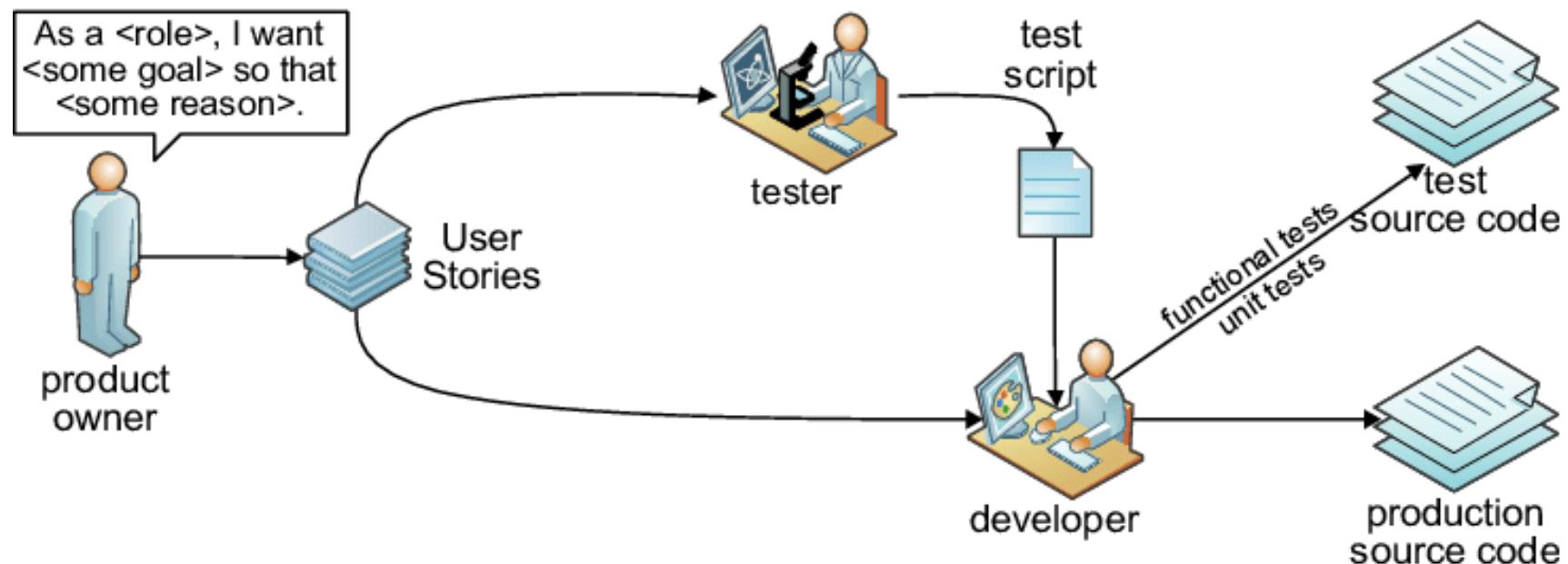
*Come utente della biblioteca
voglio consultare il catalogo
per prendere a prestito un volume*

Aggiungiamo dettagli sulle condizioni di svolgimento della US:

- Considerare se il volume desiderato è già in prestito
- Considerare se l'iscritto ha già in prestito troppi volumi
- Considerare se il volume desiderato è ordinato ma non arrivato
- Considerare se il volume desiderato andrebbe acquistato

Perché servono le condizioni aggiuntive

- Le US sono molto semplici, spesso occorrono molti dettagli in più
- Come si può testare una user story?
- Come gestire il tracciamento (cioè la traduzione) delle user story nel codice sorgente e nei test?



I principi INVEST

Le user story dovrebbero essere

- Independent
- Negotiable
- Valuable
- Estimable
- Small
- Testable

Indipendenti

- Le storie devono essere indipendenti o solo minimamente interdipendenti.

In qualità di contabile
devo poter modificare
la modalità di calcolo
della tabella
consuntiva

In qualità di contabile
devo poter modificare
la modalità di edit
della tabella
consuntiva

Negoziabili

**Nessuna storia è immodificabile
(non è incisa nel marmo)**

- Le storie sono sempre modificabili
- Modificarle implica un'assunzione di responsabilità da parte dell'utente
- La quantità di lavoro richiesta per le modifiche viene sottratta allo sviluppo di altre storie

(di) Valore

- Le storie devono avere un valore reale per l'Utente, per lo Sviluppatore o entrambi
- Conviene di solito includere gli aspetti non funzionali



E-stimabili



- Le storie devono consentire la stima del lavoro (costo) richiesto per completarle in codice
- La stima serve al Team per capire l'*effort* necessario e al PO per valutarne l'opportunità e la priorità
- Di solito le US più difficili e costose andrebbero fatte per prime
- La difficoltà di effettuare una stima può essere indice di una storia di livello troppo alto o di incapacità del team

Succinte (brevi)

- Le storie brevi sono d'immediata realizzazione in un singolo sprint: sono dunque da preferire
- Una storia non sintetizzabile in forma succinta non è necessariamente cattiva ma occorre limitarne il numero e sottoporle a revisione per una semplificazione quando possibile
- Le storie complesse vengono dette “Epiche”

Testabili

- Ogni storia va accompagnata con una serie di criteri di validazione di tipo binario (ok-nok)
- I criteri di test sono necessari per la Definizione di “Fatto” (*Definition-of-Done*), che conclude il ciclo di sviluppo relativo alla singola storia oppure delle storie di uno sprint

User story: esempio con test

- Come utente del portale per cercare lavoro voglio iscrivermi inserendo il mio CV e lo stipendio che desidero, per ricevere informazioni su ogni offerta di lavoro che soddisfi la mia richiesta
- Nota: mostrare stipendio, descrizione, e luogo
- Testare con descrizione CV vuota
- Testare con descrizione CV molto lunga
- Testare con stipendio mancante
- Testare con stipendio a sei cifre

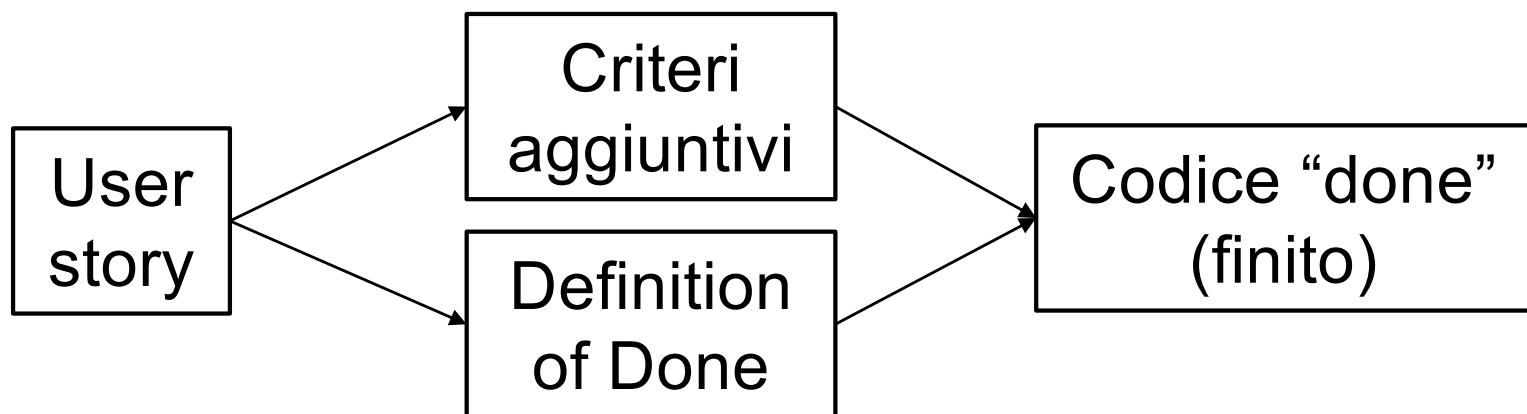
User story «completate»

Come <tipo_utente>

Voglio poter <fare qualcosa>

Perché/affinché <ottengo valore>

- Criteri aggiuntivi (es. l'utente deve riempire tutti i campi del modulo)
- DoD: validazione finale (es. superati tutti i test di unità)



Il metodo MoSCoW

Il metodo MoSCoW è una tecnica di analisi delle user story per scegliere le funzionalità più importanti del prodotto (cioè per metterle in priorità). MoSCoW sta per **Must Have**, **Should Have**, **Could Have** e **Won't Have**.

- **Must Have** (deve avere): sono le user story che rappresentano le funzionalità essenziali e non negoziabili del prodotto. Senza queste funzionalità, il prodotto non può essere considerato soddisfacente per gli utenti.
- **Should Have** (dovrebbe avere): sono le user story che rappresentano le funzionalità importanti ma non essenziali del prodotto. Sono importanti, ma possono essere rinviati a una successiva versione del prodotto se necessario.
- **Could Have** (potrebbe avere): sono le user story che rappresentano le funzionalità desiderabili ma non fondamentali del prodotto. Sono funzionalità che potrebbero migliorare l'esperienza utente, ma non sono essenziali per il prodotto.
- **Won't Have** (non deve avere): sono le user story che rappresentano le funzionalità che non sono richieste e che possono essere considerate per una futura versione del prodotto o scartate definitivamente.

Esempio (“to-do” app)

Must Have:

- Come utente, vorrei poter creare una nuova attività nella mia lista To-Do.
- Come utente, vorrei poter cancellare una attività dalla mia lista To-Do.

Queste US richiederebbero uno sforzo di sviluppo considerevole. La prima US richiede la sincronizzazione della lista To-Do con altri servizi di calendario o task manager, il che richiederebbe

Should Have:

- Come utente, vorrei poter aggiornare la data di una attività nella mia lista To-Do.
- Come utente, vorrei poter aggiungere note o commenti alle attività nella mia lista To-Do.

l'integrazione di API di terze parti. Questa funzionalità potrebbe essere implementata in una versione futura dell'applicazione, ma non è essenziale per soddisfare i requisiti principali dell'app To-Do.

La seconda US richiede l'implementazione di un sistema di gestione dei progetti completo. Anche in questo caso, questa funzionalità potrebbe essere utile per alcuni utenti, ma non è essenziale per la funzionalità principale dell'app To-Do.

Could Have:

- Come utente, vorrei poter creare sottoprogetti per aiutare a organizzare i miei compiti.
- Come utente, vorrei poter usare la mia lista To-Do per fornire maggiori dettagli.

gere note o commenti alle attività nella mia lista To-

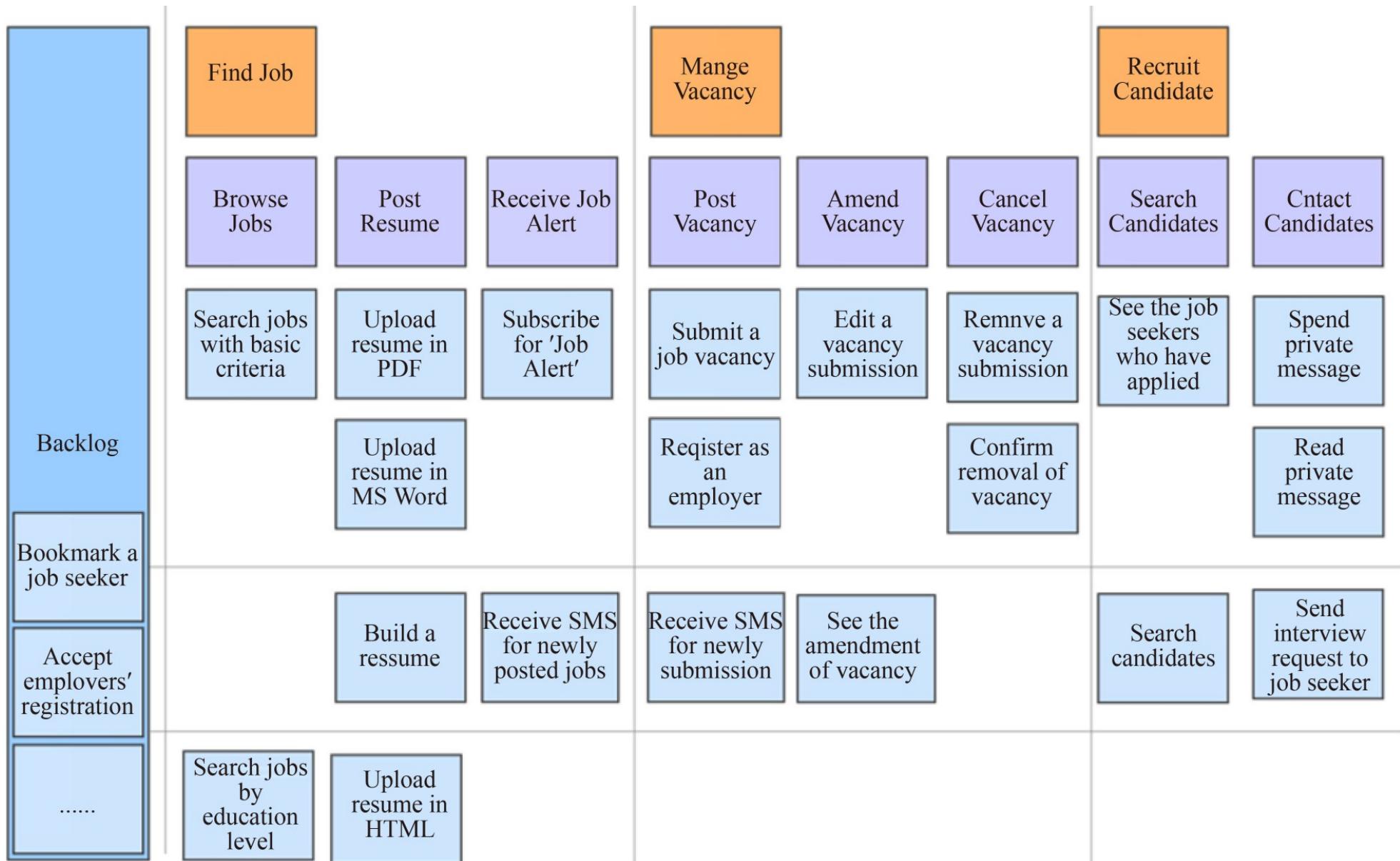
Won't Have:

- Come utente, vorrei poter sincronizzare la mia lista To-Do con altri servizi di calendario o task manager.
- Come utente, vorrei poter usare la mia lista To-Do per gestire progetti multipli con scadenze e task assegnati a diversi membri del team.

Esempio di backlog: app per cercare lavoro

ID	User story	User task	activity
0	As a job seeker, I want to search jobs with basic criteria, so that I can find a suitable job.	Browse jobs	Find job
1	As a job seeker, I want to select desired working location, so that I can save transportation costs.	Browse jobs	Find job
2	As a job seeker, I want to upload resume in PDF, so that others can view my resume online.	Post resume	Find job
3	As a job seeker, I want to upload resume in Word, so that others can give me some advice and help me to amend it.	Post resume	Find job

User story mapping



Story mapping

Prima di iniziare con la mappatura, occorre identificare chi è la persona o la **tipologia di utente** di riferimento.

La mappa è composta da due assi:

- **asse orizzontale** : questa dimensione rappresenta il “workflow”, ovvero l’interazione dell’utente con il nostro prodotto
- **asse verticale**: qui vengono mappate le attività (ma anche le componenti software, se necessario) con il corrispondente step del workflow. Ogni attività, corrisponde a una user story del nostro product backlog

Story mapping: esempio

Story Mapping: Booking online



Anna: voglio comprare un pacchetto dal sito di viaggi

Cerca pacchetto di viaggio

Visualizza pacchetto di viaggio

Seleziona pacchetto di viaggio

Sign up

Acquista pacchetto di viaggio

Ricevi informazioni di viaggio

Versione 20.01

Sfoglia catalogo

Visualizza dettagli pacchetto

Aggiungi al carrello

Login via email

Ordina

Invia biglietto via email

Ricerca semplice

Integrazione con sistema booking voli

Iscrizione alla newsletter

Paga con bonifico

Versione 20.02

Ricerca avanzata

Confronta due o più pacchetti

Salva tra i preferiti

Login via social media

Paga con carta di credito

Invia offerte correlate via email

Integrazione con software gestionale

Story mapping

- Oltre ad esprimere le attività, l'asse verticale riflette anche quelle che sono le priorità
- È possibile delimitare con **linee orizzontali** ciascun rilascio o incremento, in modo da dare visibilità sul “quando” determinate funzionalità saranno rese disponibili
- Ogni partecipante può suggerire azioni da includere nel workflow e il Product Owner potrà, in un secondo momento, riordinare tali azioni in base al valore di business e assegnarli a determinati rilasci
- L'evento di Story Mapping dovrebbe coinvolgere un gruppo diversificato di persone. Oltre agli sviluppatori, che hanno competenze tecniche, sarebbe utile includere anche persone esperte del dominio del prodotto
- Uno strumento utile per lo story mapping è [Miro](#)

Gioco: US mapping

- La vostra sveglia suona alle 8, dovete uscire entro 30 minuti
- Scrivete almeno 30 diverse azioni che fate per prepararvi e uscire
- Mettete a ciascuna azione l'etichetta temporale
- Al termine raggruppatele per tema

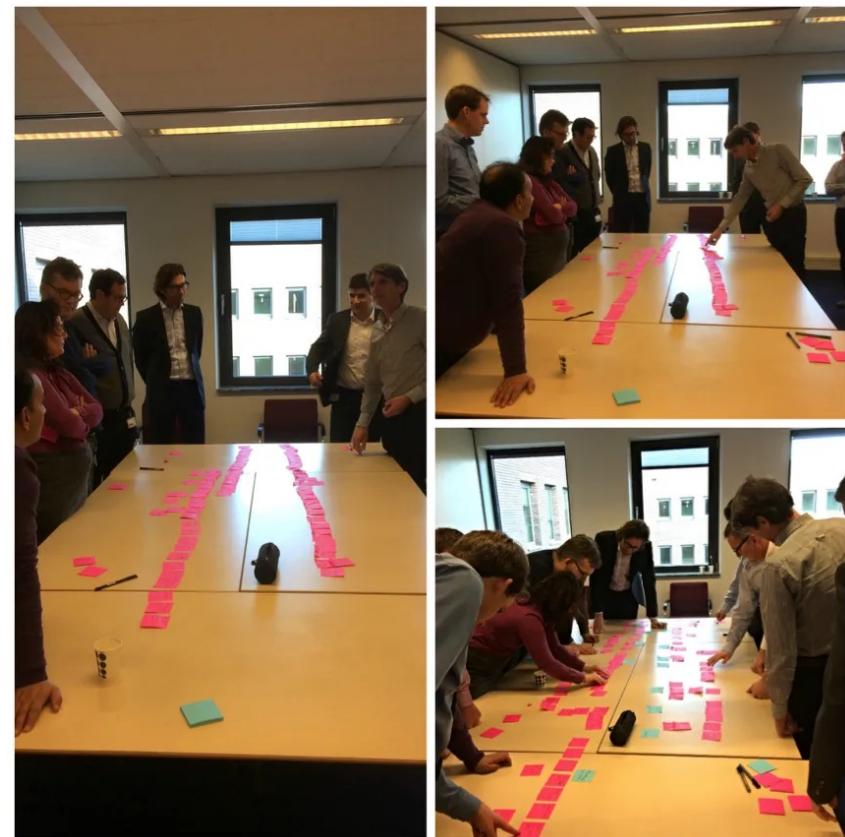
Esempi:

- Spegni la sveglia 8.00.01
- Esci dal letto 8.00.05
- Entra in doccia 8.00.30
- ...

Potete usare un prodotto specializzato per US mapping, per esempio: <https://www.avion.io>

Turni successivi

- 2 turno: avete solo 15 minuti
- 3 turno: avete solo 10 minuti
- 4 turno: avete solo 5 minuti

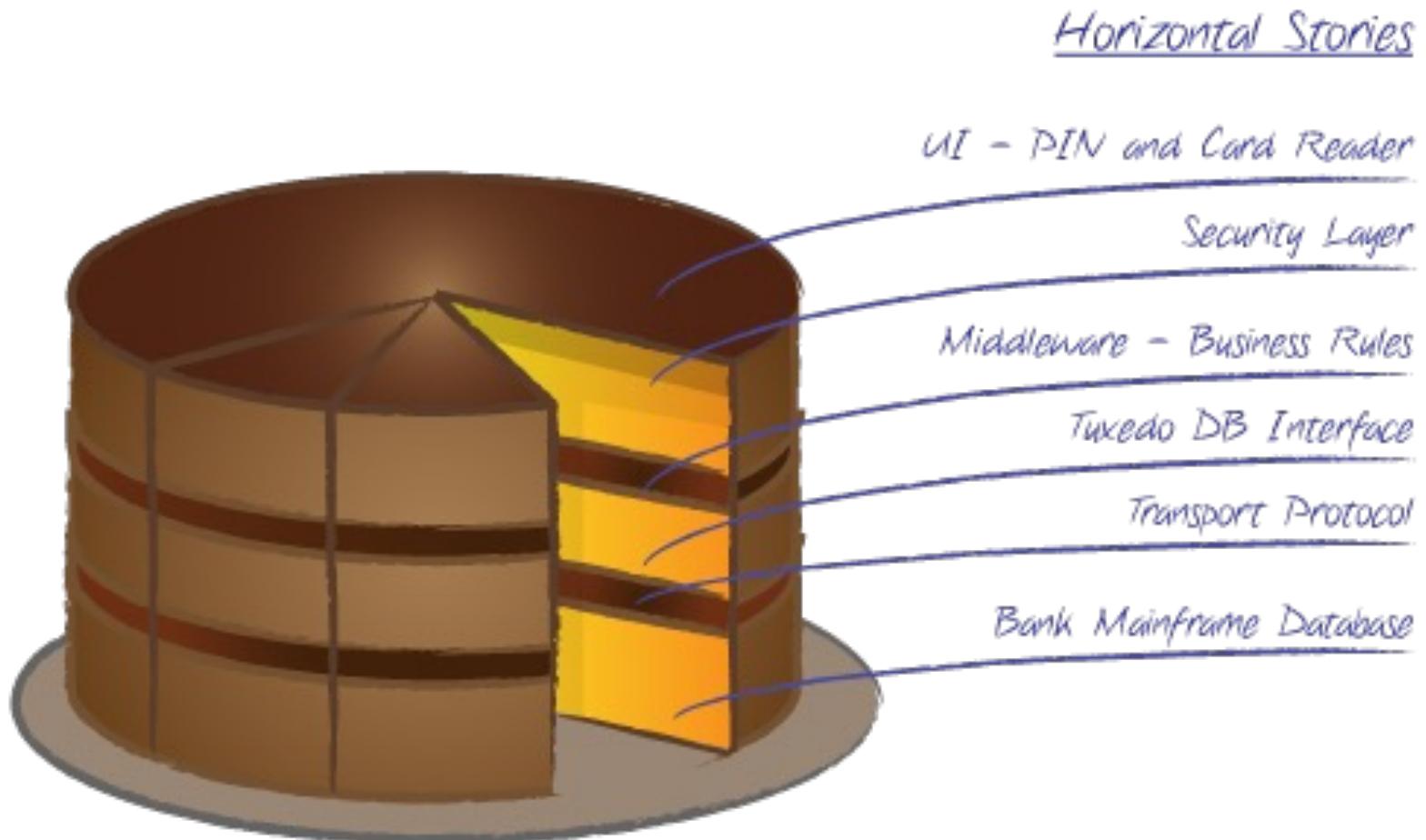


Non è facile scrivere user stories

- Spesso si partiziona un backlog di prodotto *orizzontalmente*, stratificando l'architettura funzionale da costruire
- Gli sviluppatori preferiscono lavorare così per ragioni di efficienza: gli strati inferiori sono concepiti per essere riusati dagli strati superiori
- Però se si lavora così è più difficile scrivere le user stories dal punto di vista degli utenti

Automated Teller Machine (ATM)

Horizontal and Vertical User Stories - Slicing the Cake



User stories “verticali”

- Permettono di confezionare e consegnare rapidamente singole funzionalità
- Feedback continuo dal cliente

Automated Teller Machine (ATM)

Horizontal and Vertical User Stories - Slicing the Cake

Vertical User Stories

Cash Withdrawal (90% usage)

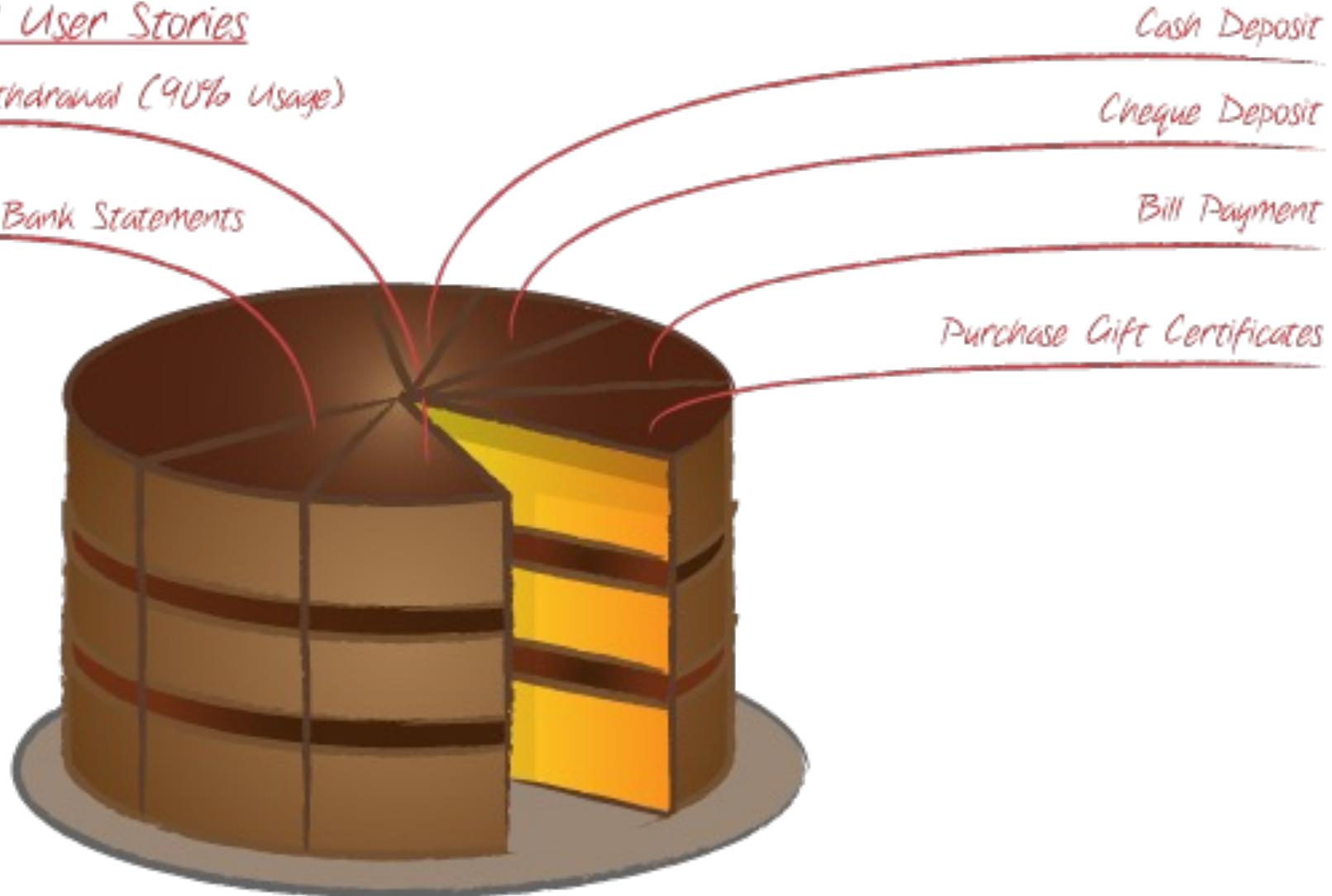
Printing Bank Statements

Cash Deposit

Cheque Deposit

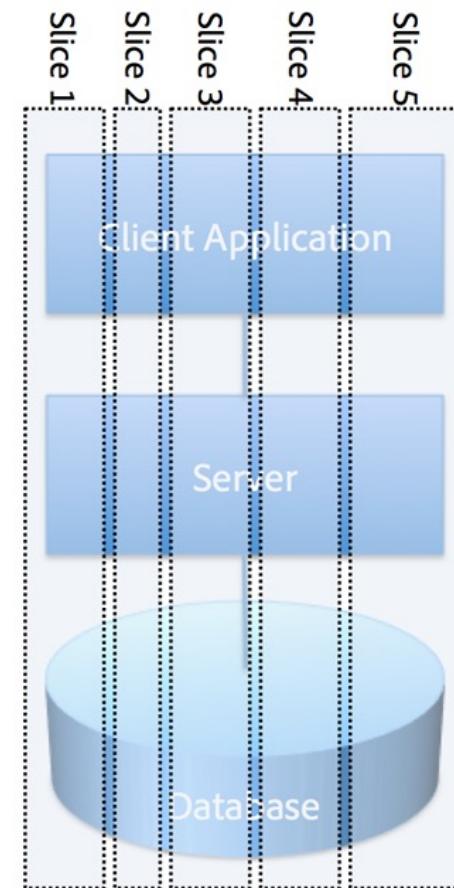
Bill Payment

Purchase Gift Certificates



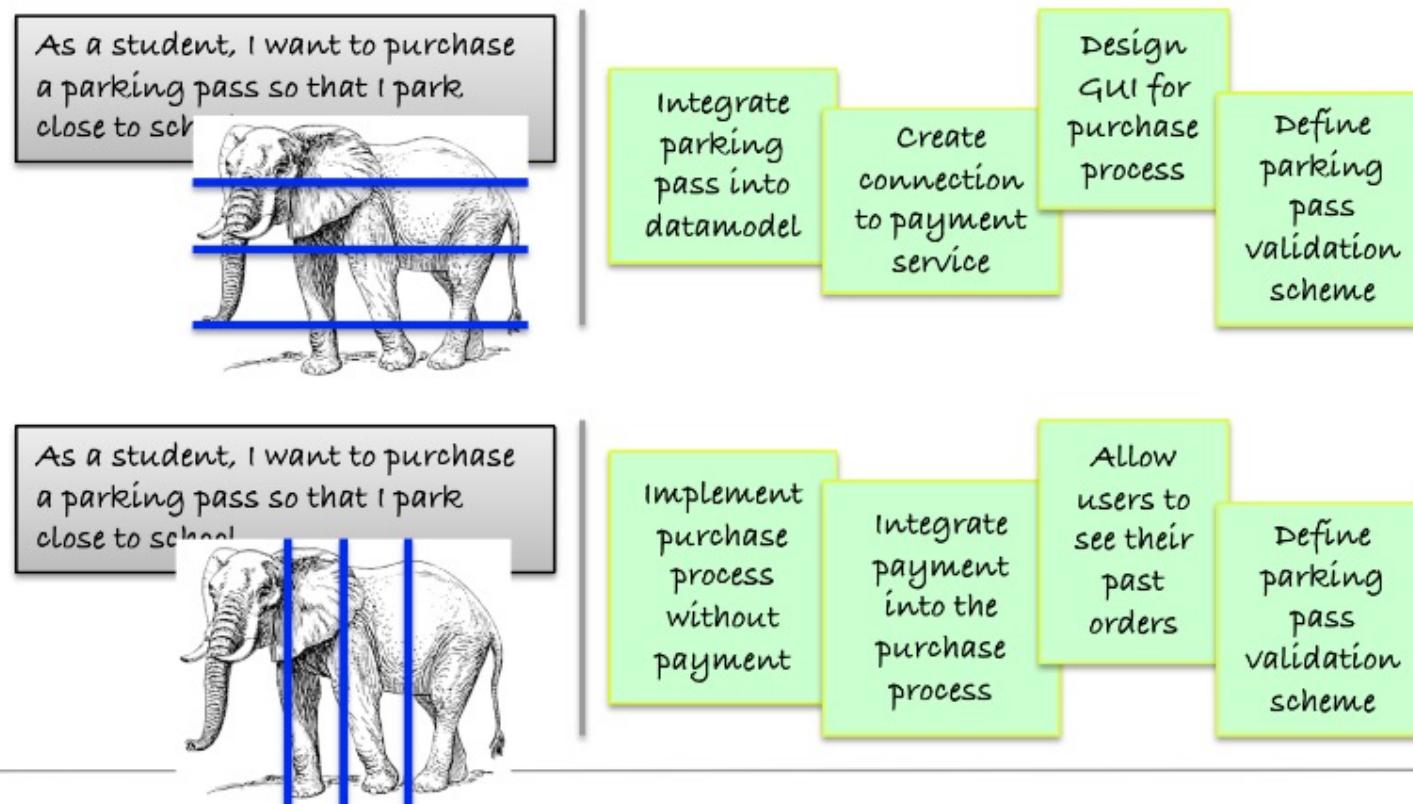
Vantaggi delle storie verticali

- **Ambiguità** La disambiguazione di termini vaghi può aiutare l'articolazione delle storie
- **Congiunzioni** Una funzione descritta mediante una congiunzione può facilmente essere decomposta
- **Accettazione** Il test di accettazione può indurre ad affettare una storia così da renderla più facilmente testabile
- **Segmentazione** Un compito complesso viene spezzato in più passi semplici



Storie verticali - orizzontali: esempio

Vertical Slicing/Laminating



https://www.linkedin.com/posts/nussi-einhorn_i-dont-like-user-stories-its-my-activity-6798607414863159296-vHKz

User story orizzontali

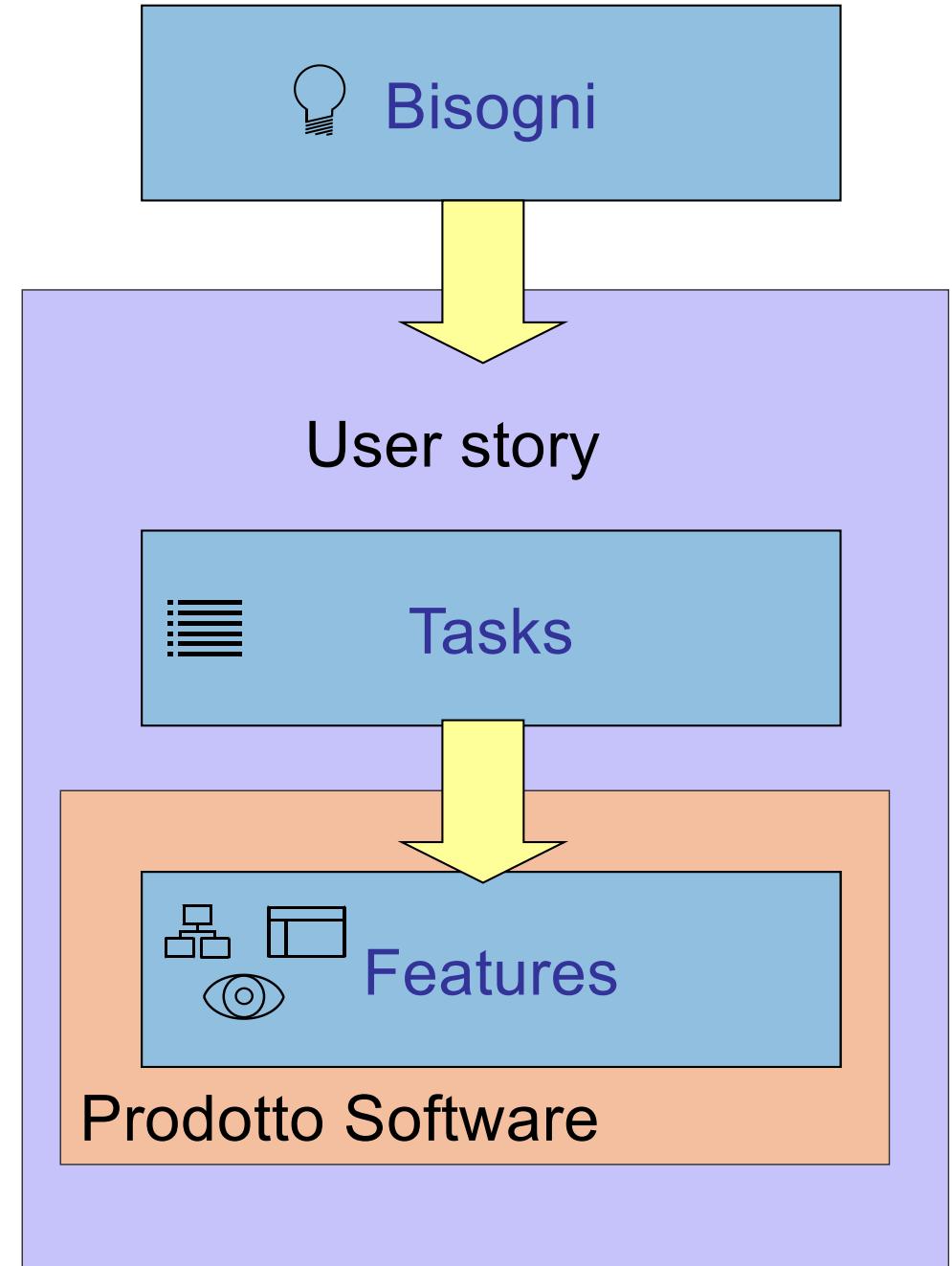
User Stories

- As a user of the app I would like the ability to be able to put in my username and password and login that way because it seems more secure.
- As a user I would like to have the ability to reset my password when I feel the need to.
- As a user I would like to be able to sign in on the app at any given time so I can access my data easily.
- As a user I would like to have indications of what is required to set a new password so I can know what password to choose.
- As a user I would like to be notified when a field has an invalid input so I can know what I need to fix.
- As a user I would like the ability to log out of the app so I can securely know my data is locked.
- As a user I would like to have 2-factor authentication with SMS so I can secure my account even more.
- As a user I would like the ability to login via my Facebook account so I don't have to create a new account for this app.
- As a user I would like the ability to add on Face ID so I can use my Apple device to authenticate my identity.

After

- Standard Login
- Reset Password
- Sign In - Authenticate user
- Password Requirement help
- In-line Feild validation
- Logout Function
- 2-factor Auth
- Facebook Login
- Face ID login

Dalle user stories al prodotto





Product Backlog Item

Something to build into the product to enhance its value.

Identified

Ready for Development

Done

Relates to: Requirements





INVEST

An acronym of the quality criteria for a Product Backlog Item to be ready for development:

- **Independent** – can be independently built / tested
- **Negotiable** – e.g. in its detail or even into or out of scope
- **Valuable**
- **Estimable** – can be sized
- **Small** – enough e.g. to be implemented rapidly, e.g. in a week or less
- **Testable**.

Relates To:

Ref:





Agree Definition of Done

Agree the quality criteria that will be used to determine whether any change to the product is fully and correctly implemented.

Understand the Requirements



Stakeholder Representation



Analysis



Development



Testing

Requirements: Coherent (contributes to)

Definition of Done: Completion Conditions Listed or beyond



Definition of Done

The quality criteria that will be used to determine whether the product is of acceptable / releasable quality.



Completion Conditions Listed

Quality Criteria and Evidence Described

Describes: Requirements



Definition of Done

Definition of Done

The quality criteria that will be used to determine whether the product is of acceptable / releasable quality.

Completion Conditions Listed

Quality Criteria and Evidence Described

Describes: Requirements

  AE - PB

 IVAR JACOBSON INTERNATIONAL
Generated by IJI Practice Workbench™

2018.09

Definition of Done

Completion Conditions Listed

- The deliverables that need to be supplied are listed, including supporting documentation etc.
- The quality criteria and levels that need to be achieved are described and justified
- It is clear who needs to accept which outputs for them to be accepted as done

1 / 2

 AE -  PB

 IVAR JACOBSON INTERNATIONAL
Generated by IJI Practice Workbench™

2018.09

Definition of Done

Quality Criteria and Evidence Described

- The types of quality assurance check / test that need to be completed are described
- It is clear how the test results will be evaluated, analysed and evidenced

2 / 2

 AE -  PB

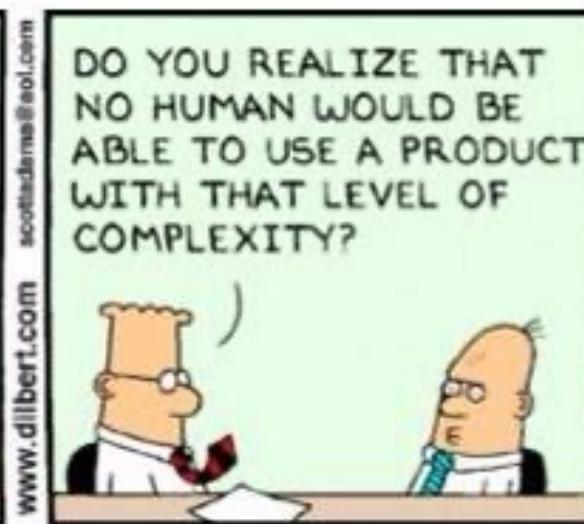
 IVAR JACOBSON INTERNATIONAL
Generated by IJI Practice Workbench™

2018.09

Riferimenti

- Cohn, *User stories applied*, Addison-Wesley, 2004
- Pichler, *Agile product management with Scrum*, Addison-Wesley, 2010

Domande?



Casi d'uso e scenari



*Corso di Ingegneria del su
Università di Bologna*

Definire i requisiti

Per avere una visione chiara e condivisa tra tutti gli stakeholder su ciò che deve essere creato, è importante definire i requisiti di un prodotto digitale. Le tecniche principali per raccogliere nuovi requisiti sono:

- **Interviste agli utenti**: per comprendere esigenze e aspettative rispetto ad un prodotto digitale esistente o nuovo.
- **Analisi dei concorrenti**: l'analisi delle funzionalità dei prodotti digitali simili presenti sul mercato per identificare le funzionalità importanti da includere nel proprio prodotto.
- **analisi dei requisiti**: studia le esigenze degli utenti, il contesto del prodotto e le funzionalità utili per soddisfare le esigenze.
- **Brainstorming**: generazione di idee da parte del team attraverso sessioni di brainstorming
- **Prototipazione**: creazione di prototipi del prodotto digitale per testare le funzionalità e raccogliere feedback dagli utenti.

Forme dei documenti dei requisiti

1. **Documento di specifica dei requisiti:** la forma tradizionale di specifica dei requisiti, in cui viene creato un documento strutturato secondo lo standard. Questo documento, detto Software Requirement Specification (SRS) include la descrizione del prodotto, i suoi obiettivi, la descrizione delle funzionalità richieste, i requisiti non funzionali e le condizioni di test.
2. **User Stories:** Le user story sono una forma agile di specifica dei requisiti. Una user story è una descrizione sintetica di una funzionalità del software, scritta dal punto di vista dell'utente. Può includere alcune condizioni di accettazione
3. **Casi d'uso:** combinazione di grafica+testo che descrive come il software richiesto interagisce con l'utente e con altri sistemi. Si integrano bene con le user story
4. **Prototipi e Mockup:** I prototipi e i mockup sono strumenti digitali che aiutano visualmente a descrivere le funzionalità del software. Un prototipo è una versione semplice del prodotto che fornisce un'idea di come funzionerà, mentre un mockup è un disegno che propone una finta interazione da realizzare.

In questa lezione ci occupiamo di questa tecnica

Obiettivi della lezione

- Cosa sono i casi d'uso? Cosa sono gli scenari?
- Come si integrano con le user story?
- La grafica dei casi d'uso
- Il testo di un caso d'uso: lo scenario
- Esempi
- Carte Essence dei casi d'uso

Modellare i casi d'uso (Use Case)

- Proposti da Ivar Jacobson nel 1992
- La ricerca dei casi d'uso (*use cases*) è
lo studio degli scenari operativi degli utenti di un prodotto
- Gli **scenari** sono i “modi” in cui il prodotto può essere utilizzato - cioè definiscono le operazioni o funzioni che il prodotto mette a disposizione dei suoi utilizzatori



Ivar Jacobson

User story vs caso d'uso

User story

Definisce chi, cosa e perché di una funzione del prodotto
Descrizione breve e semplice, spesso incompleta
Favorisce la conversazione e la creatività

Caso d'uso

Definisce chi vuole cosa dal prodotto – in qualche contesto, detto **scenario**
Descrizione dettagliata completa e concreta

US e casi d'uso

esempio di tre US per una To-Do App:

1. Come utente, voglio poter creare una lista di cose da fare in modo che io possa organizzare il mio lavoro.
2. Come utente, voglio poter impostare una data di scadenza per le mie attività in modo che io non dimentichi di completarle in tempo.
3. Come utente, voglio poter segnare le attività come completate in modo che possa tener traccia di ciò che ho fatto e di ciò che devo ancora fare.

Ecco alcuni casi d'uso che descrivono il comportamento della To-DO App:

UC1: Creare una lista di attività

- - L'utente seleziona l'opzione "Aggiungi attività" dal menu principale
- - Il sistema apre una finestra di dialogo in cui l'utente può inserire il titolo dell'attività e una descrizione opzionale
- - L'utente conferma l'aggiunta dell'attività e il sistema la aggiunge alla lista

UC2: Impostare una data di scadenza

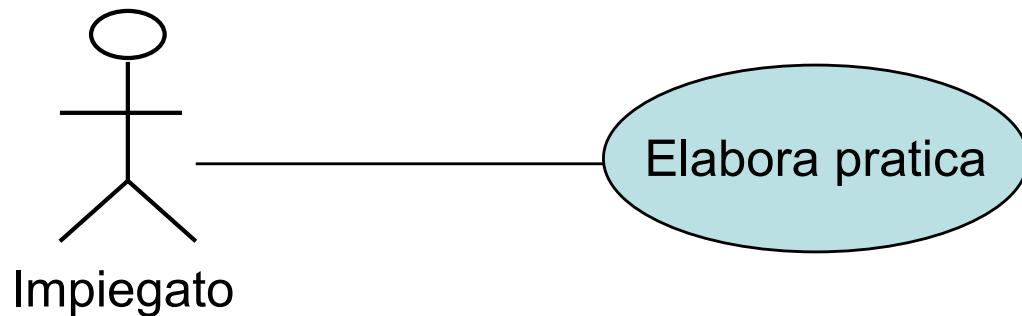
- - L'utente seleziona un'attività dalla lista
- - Il sistema mostra i dettagli dell'attività, tra cui la data di scadenza (se presente)
- - L'utente seleziona l'opzione "Modifica scadenza"
- - Il sistema apre una finestra di dialogo in cui l'utente può impostare una data di scadenza per l'attività
- - L'utente conferma la modifica e il sistema aggiorna la data di scadenza dell'attività

UC3: Segnare un'attività come completata

- - L'utente seleziona un'attività dalla lista
- - Il sistema mostra i dettagli dell'attività, tra cui lo stato (completata o da completare)
- - L'utente seleziona l'opzione "Segna come completata"
- - Il sistema aggiorna lo stato dell'attività e la rimuove dalla lista delle attività da completare.

Caso d'uso: rappresentazione grafica

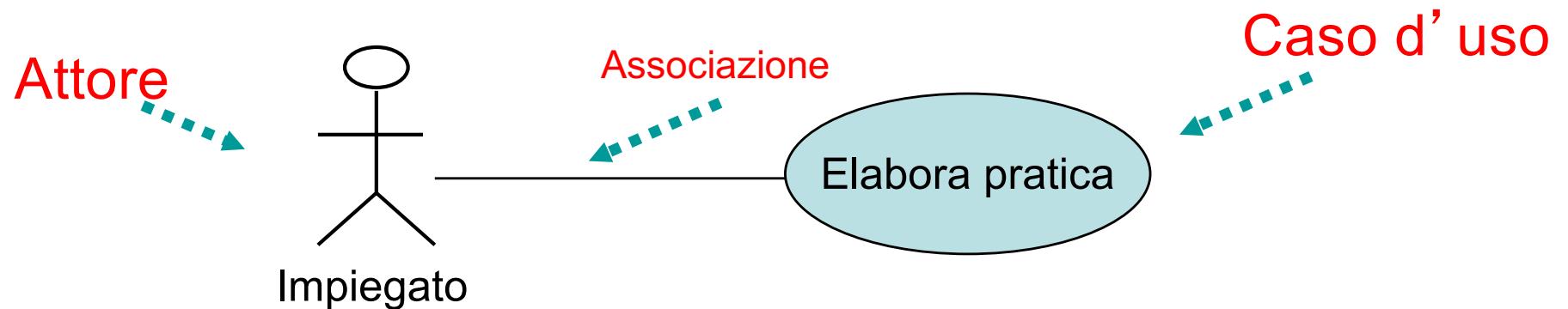
Un caso d'uso si rappresenta graficamente con un **attore** che usa una **funzione del sistema**



Associazione tra attore e caso d'uso

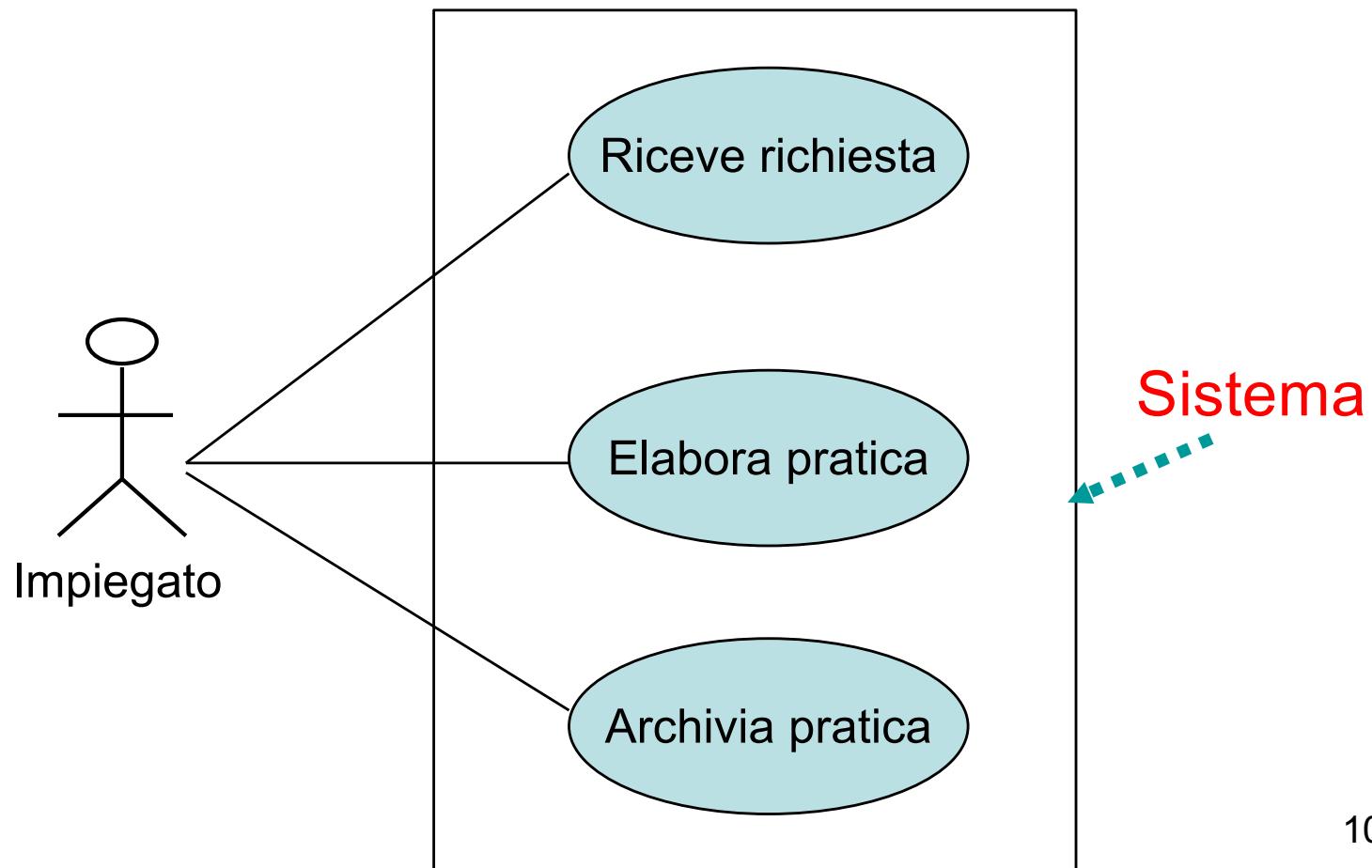
Il caso d'uso si rappresenta con un **attore** che usa una **funzione del sistema**

Diremo che all'attore «è associato» quel particolare caso d'uso

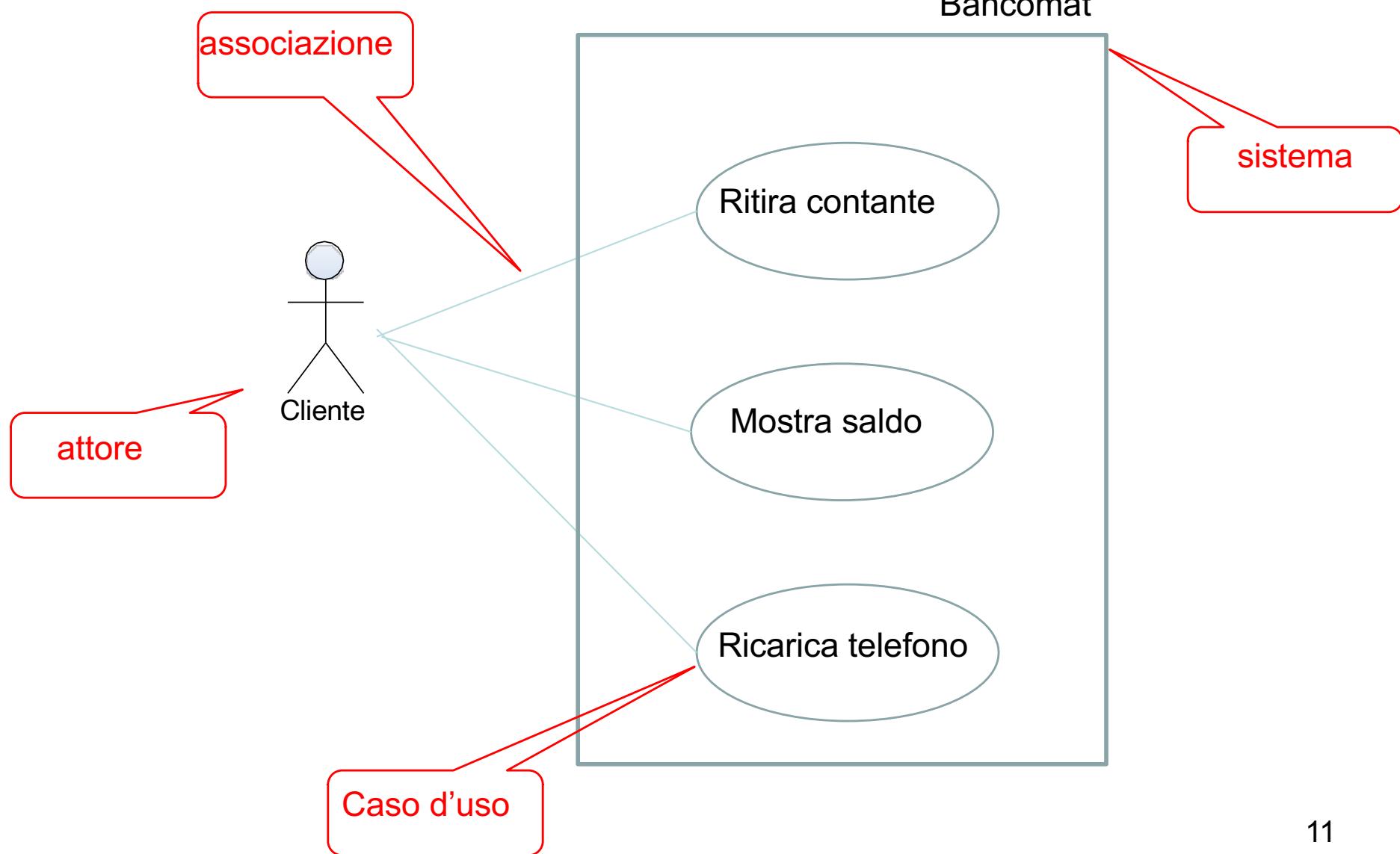


I casi d'uso rappresentano funzionalità richieste ad un sistema

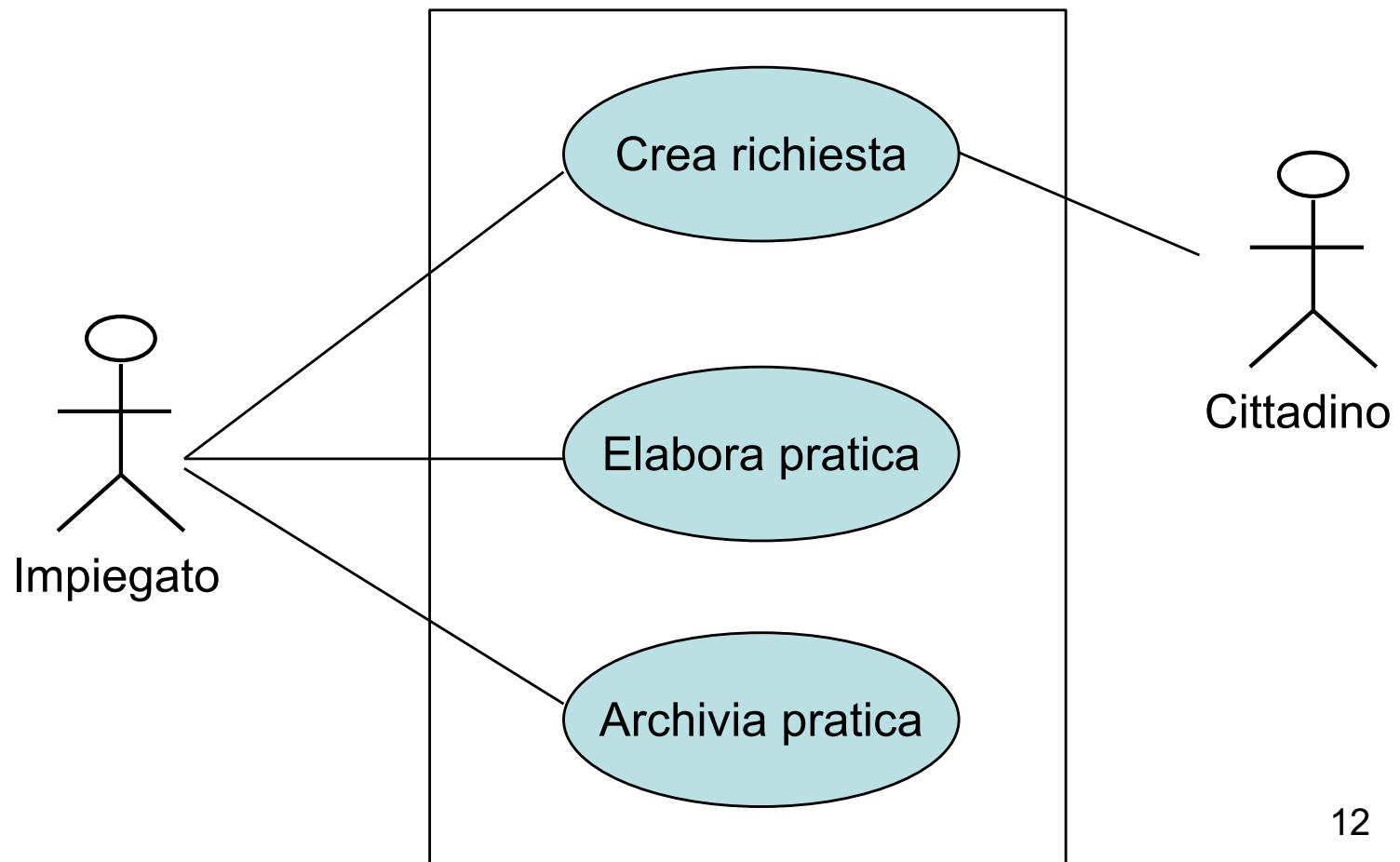
(per noi: sistema è sinonimo di prodotto)



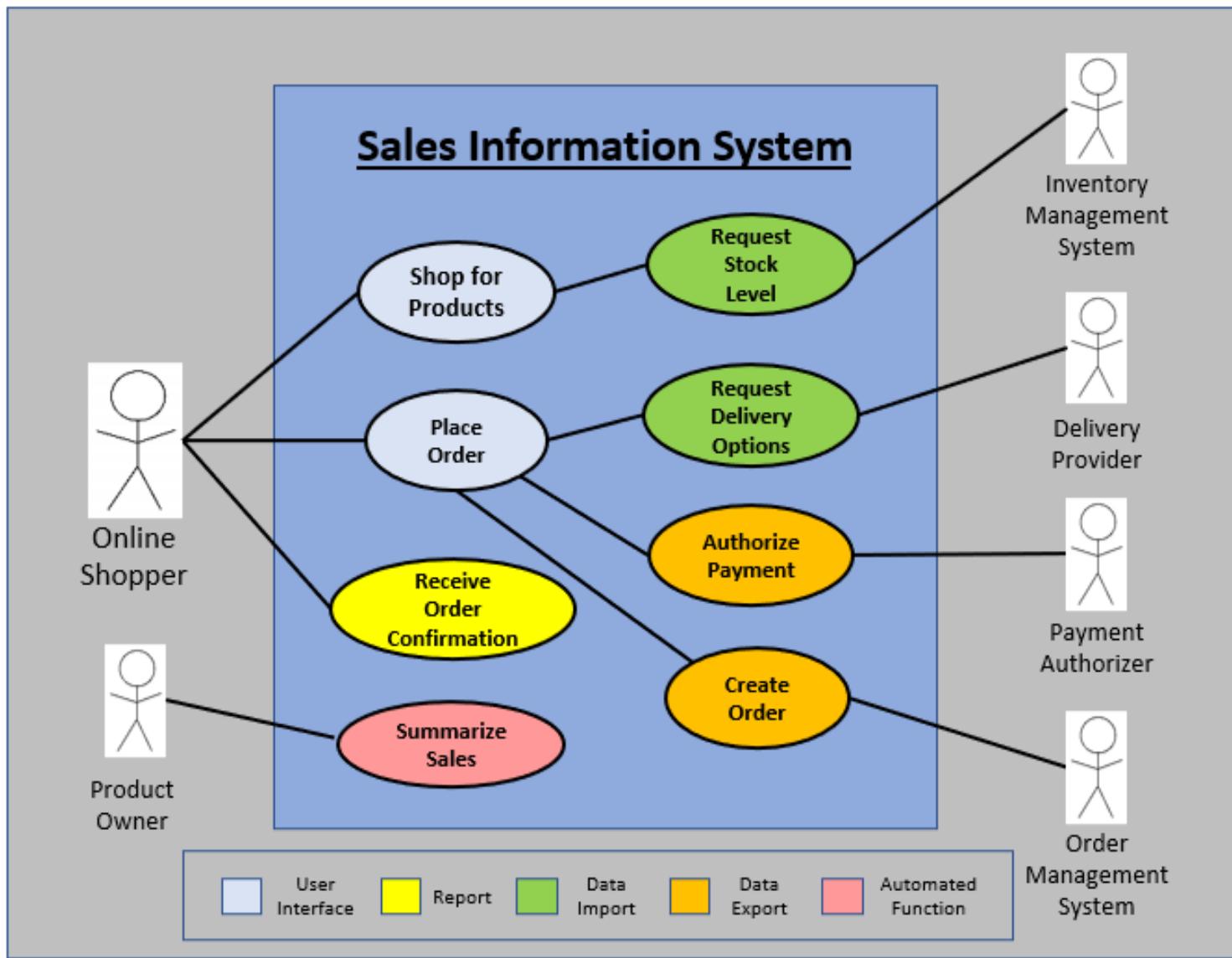
Un cliente usa un bancomat



Un caso d'uso può avere più attori



Un esempio colorato



Lo scenario delle interazioni

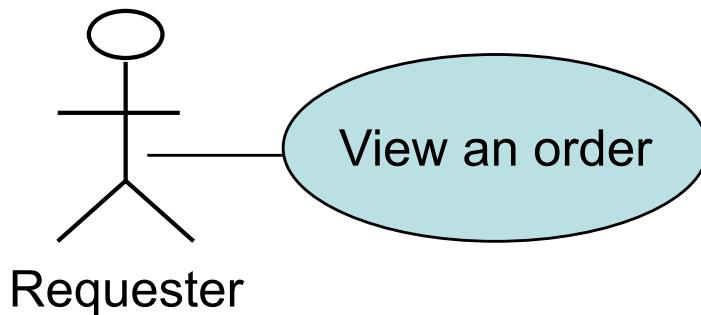
La grafica dei casi d'uso è molto essenziale, povera di dettagli
Un caso d'uso si dettaglia sotto forma di **scenario di interazione** - cioè un **dialogo** - tra un utente e il sistema

Esempio:

- il cittadino richiede l'elenco dei servizi
 - il sistema risponde con elenco dei servizi disponibili
- il cittadino sceglie dall'elenco il servizio che desidera
 - il sistema risponde col costo totale del servizio selezionato
- il cittadino accetta e conferma la richiesta di servizio
 - il sistema comunica all'amministrazione la richiesta

La «narrazione» del dialogo si focalizza sull'interazione,
non sulle attività interne al sistema

Scenario associato ad un caso d'uso



Use Case:
View an order.
Actor: Requester Frequency: 5/user/day
Preconditions: system contains orders;
user's identity is verified
Postconditions: order has been shown

Actor Actions

user enters order number he wants to view

user enters order number, but it doesn't exist

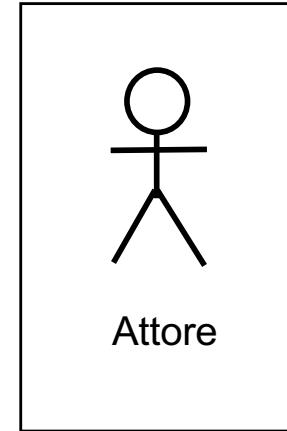
etc. for all normal and exception pathways

System Responses

display order details

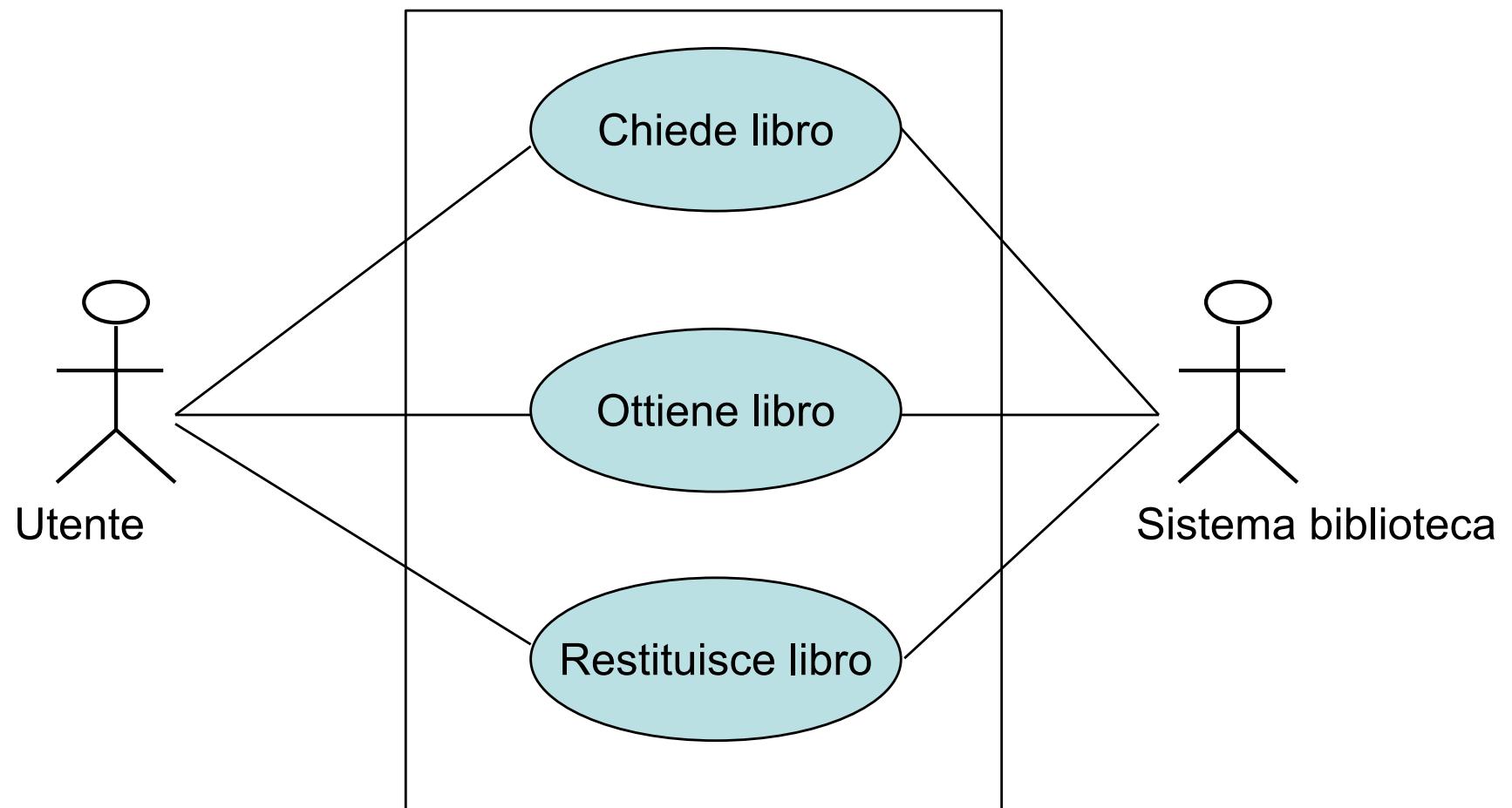
error message:
order number not found

Attore



- **Ruolo** che qualcuno o qualcosa ricopre rispetto al sistema
- Gli attori sono esterni al sistema
- Gli attori eseguono casi d'uso
 - Prima si cercano gli attori, poi i loro casi d'uso
- Gli attori “ottengono valore” dal caso d'uso o vi partecipano
- Gli attori possono NON essere persone (es.: altri sistemi, dispositivi hardware)!

Un attore può essere un altro sistema



Come trovare gli attori

- Chi o cosa è interessato al sistema?
- Chi o cosa modificherà i dati del sistema?
- Chi o cosa vuole informazioni dal sistema?
- Chi o cosa si interfacerà col sistema?

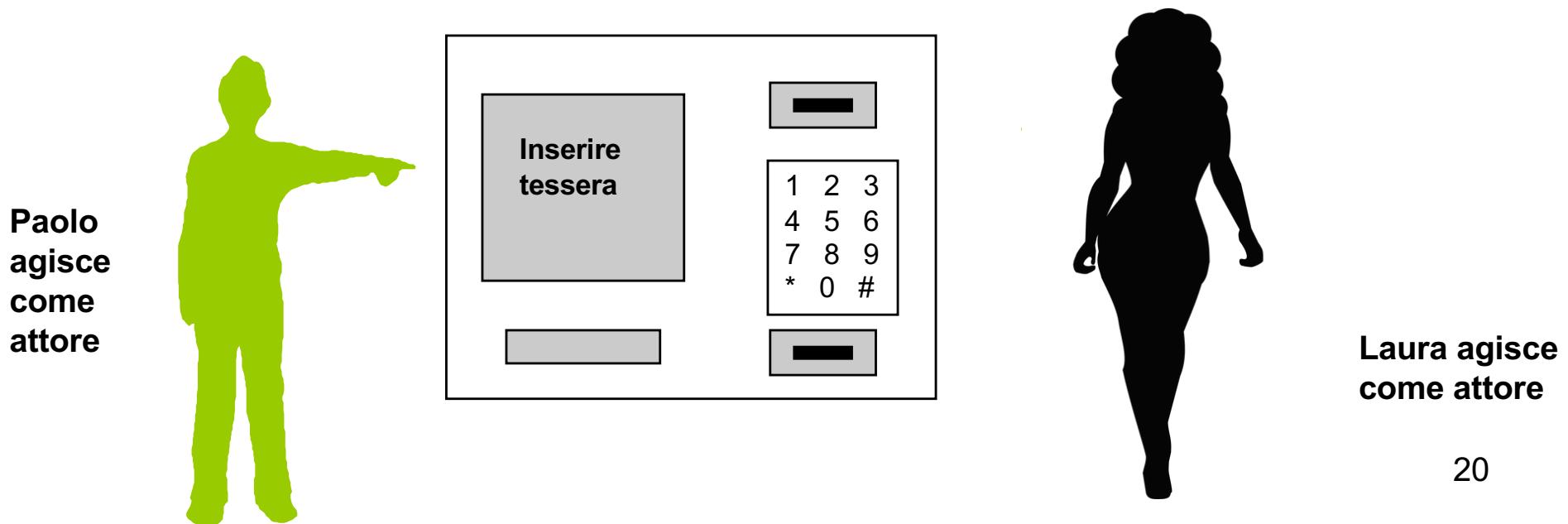
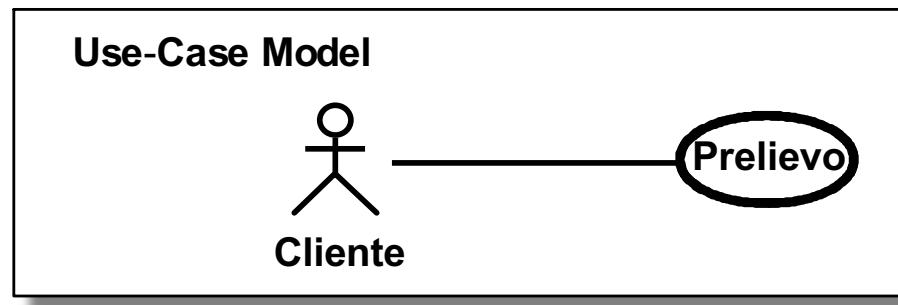
Ricerca degli attori: domande utili

- Chi è interessato a questo requisito?
- Chi utilizzerà questa funzione?
- Quali attori sono in relazione con gli Use Cases?
- Un attore ha più ruoli? Lo stesso ruolo è assegnato a più attori?

- In quale parte dell'organizzazione è utilizzato il sistema?
- Chi fornirà, utilizzerà ed eliminerà informazioni dal sistema?
- Chi supporterà e manterrà il sistema?
- Il sistema utilizzerà una risorsa esterna?

Cosa rappresenta un caso d'uso?

Un caso d'uso è una descrizione di come una persona che utilizza effettivamente quel sistema raggiungerà un obiettivo



Caso d'uso

- Rappresenta un **requisito funzionale**
- Esplicita cosa ci si aspetta da un sistema (“what?”)
- Nasconde il comportamento del sistema (“how?”)
- È una sequenza di **azioni** (con varianti) che producono un risultato osservabile da un **attore**

Si usa per

- Descrivere requisiti d’utente (analisi iniziale)
- Controllare se il sistema funziona bene (testing)

Caso d'uso

- Ogni sequenza di azioni di un attore sul sistema (sequenza detta *scenario*) rappresenta l'interazione di una entità esterna al sistema (*attore*) con il sistema stesso o sue componenti
- Il caso d'uso ha un *flusso principale* e zero o più varianti *alternative (opzionali o anomale)*

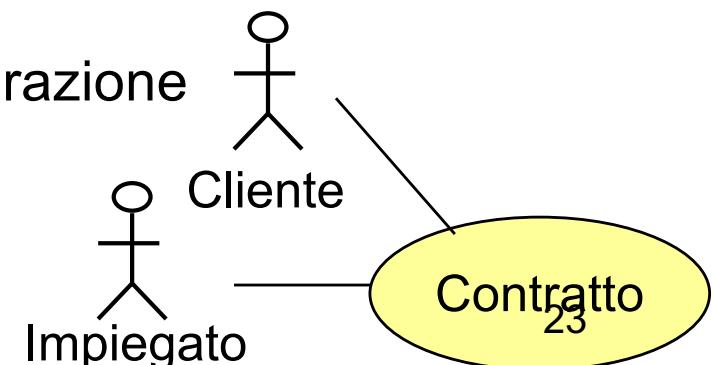
Scenario - esempio

Nome: Contratto di acquisto di azioni

Precondizione: l'impiegato è connesso

Flusso principale degli eventi:

1. Inserire nome cliente e numero conto
2. Controllare la loro validità
3. Inserire il numero di azioni da comprare e ID azienda quotata
4. Determinare il prezzo corrente delle azioni
5. Controllare il saldo del cliente
 - 5.1 Anomalia: saldo insufficiente
6. Mandare l'ordine alla Borsa
7. Memorizzare numero di conferma dell'operazione



Descrizione dello scenario

- E' una descrizione generica e sequenziale del flusso di eventi di un caso d'uso
 - Descrivere la precondizione (stato iniziale del sistema)
 - Elencare la sequenza di passi
- Include le interazioni con gli attori e descrive quali entità vengono scambiate
- La descrizione dev'essere chiara, precisa e breve

Ricerca dei casi d'uso: domande utili

- Quali sono i compiti di questo attore?
- L'attore gestirà le informazioni del sistema?
- Quali Casi d'Uso creeranno, modificheranno, leggeranno questa informazione?
- L'attore deve informare il sistema di cambiamenti improvvisi?
- L'attore dev'essere informato di certe situazioni?
- Il sistema supporta il business con un comportamento corretto?
- Quali Casi d'Uso supportano e manutengono il sistema?
- I requisiti funzionali sono tutti coperti dai Casi d'Uso che abbiamo trovato?

Fonti di informazione per i casi d'uso

- Documenti di specifica del sistema
- Bibliografia del dominio del sistema
- Interviste con gli esperti del dominio
- Conoscenza personale del dominio
- Sistemi simili già esistenti

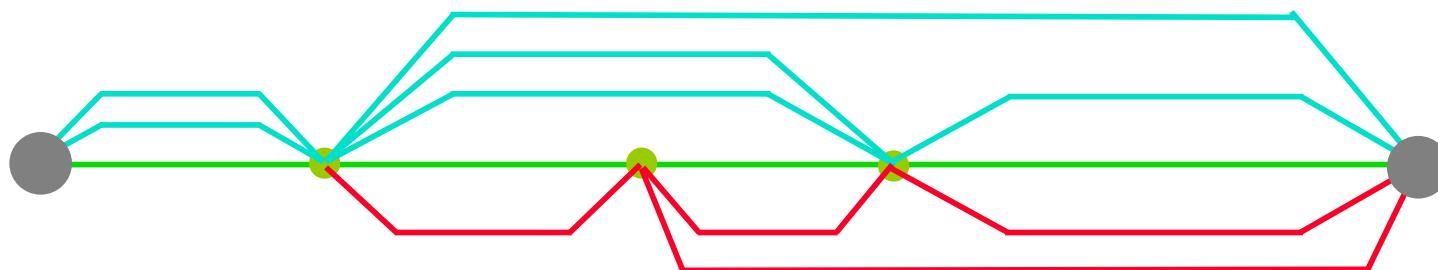
Documentazione sui casi d'uso

- I casi d'uso sono documentati da
 - Una breve descrizione
 - Lo scopo degli use case in poche linee
 - Flusso dettagliato degli eventi
 - Descrizione dei flussi primari ed alternativi degli eventi che seguono lo start-up dello use case
 - La documentazione del caso d'uso riporta il dialogo tra l'attore e il sistema
- Tutti i documenti sono scritti in modo comprensibile per il cliente (che li valida)

Flusso degli eventi

Ogni caso d'uso

- Ha una sequenza principale di azioni
- Può avere alcune sequenze alternative opzionali
- Ha alcune sequenze di azioni speciali per la gestione di situazioni erronee o anomale



Un caso d'uso ha uno scenario base più alcuni scenari alternativi

Scenario base: è quello che prevede il successo del caso d'uso,
ed uno svolgimento lineare

Scenari alternativi: possono essere di successo o fallimento, con
complicazioni varie

- non è necessario (e sarebbe molto costoso) analizzare in dettaglio tutti i possibili scenari di un caso d'uso
- è invece necessario individuare le singole possibili varianti che possono portare al fallimento del caso d'uso, o che comportano trattamenti particolari

Flusso degli eventi

- Descrive solo gli eventi relativi al caso d'uso
- Evita l'uso di termini vaghi come "per esempio", "ecc." o "informazione"
- Il flusso degli eventi dovrebbe descrivere
 - Come e quando il caso d'uso inizia e finisce
 - Quando il caso d'uso interagisce con gli attori
 - Quali informazioni sono scambiate tra un attore e il caso d'uso
 - Si tralasciano i dettagli dell'interfaccia utente
 - Il flusso di base degli eventi
 - Ogni flusso alternativo degli eventi

Come scrivere un caso d'uso

- Assegnare un nome al caso d' uso
- Descrivere l'attore primario ed eventuali comprimari
- Descrivere la condizione iniziale del sistema
- Descrivere il flusso principale degli eventi
- Descrivere la condizione d'uscita
- Descrivere possibili anomalie ed eccezioni
- Descrivere i requisiti di qualità (non funzionali)

Esempio: apertura conto corrente

- 1 il cliente si presenta in banca per aprire un nuovo c/c
- 2 l'addetto riceve il cliente e fornisce spiegazioni
- 3 se il cliente accetta fornisce i propri dati
- 4 l'addetto verifica se il cliente è censito in anagrafica
- 5 l'addetto crea il nuovo conto corrente
- 6 l'addetto segnala il numero di conto al cliente

Varianti:

- 3 (a) se il cliente non accetta il caso d'uso termina
- 3 (b) se il conto va intestato a più persone vanno forniti i dati di tutte
- 4 (a) se il cliente (o uno dei diversi intestatari) non è censito l'addetto provvede a registrarlo, richiede al cliente la firma dello specimen e ne effettua la memorizzazione via scanner

Esempio: apertura conto corrente

- ulteriore dettaglio

(5) l'addetto crea il nuovo conto corrente

- 1 l'addetto richiede al sistema la transazione di inserimento nuovo conto
- 2 il sistema richiede i codici degli intestatari
- 3 l'addetto fornisce i codici al sistema
- 4 il sistema fornisce le anagrafiche corrispondenti, e richiede le condizioni da applicare al conto
- 5 l'addetto specifica le condizioni e chiede l'inserimento
- 6 il sistema stampa il contratto con il numero assegnato al conto

Varianti:

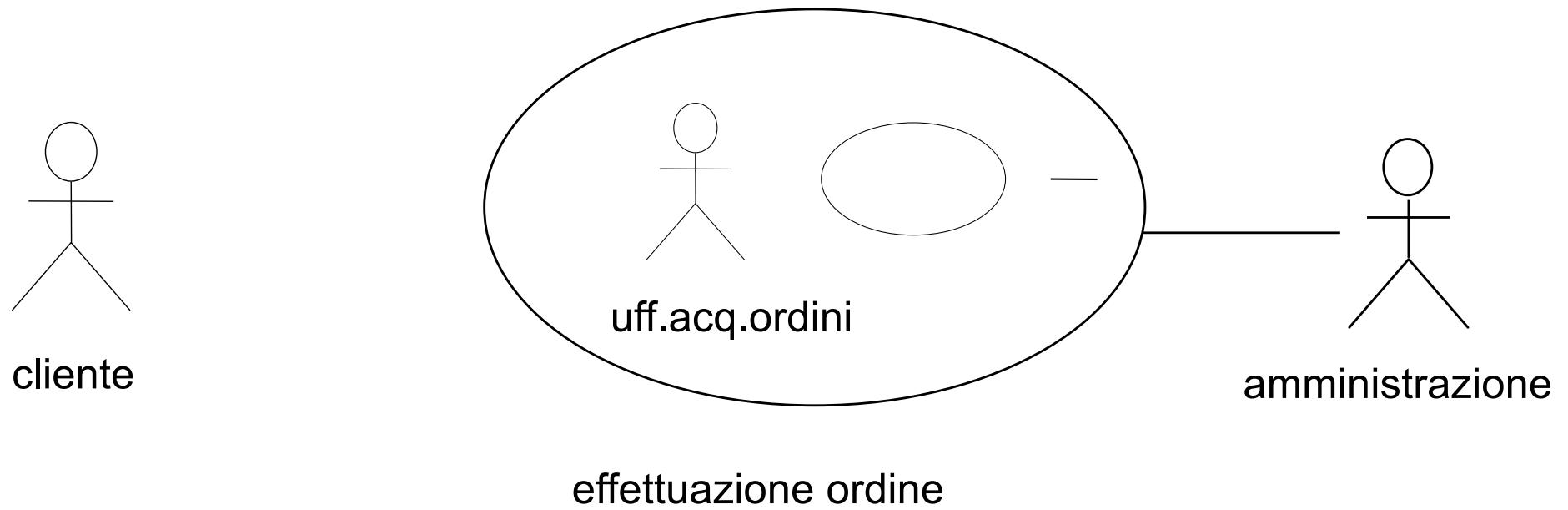
- 3 (a) se il sistema non riconosce il cliente, o se fornisce un'anagrafica imprevista, l'addetto può effettuare correzioni o terminare l'inserimento

Attori “business” e “di servizio”

Nel definire gli attori si possono adottare due diversi punti di vista che corrispondono a diversi livelli di astrazione:

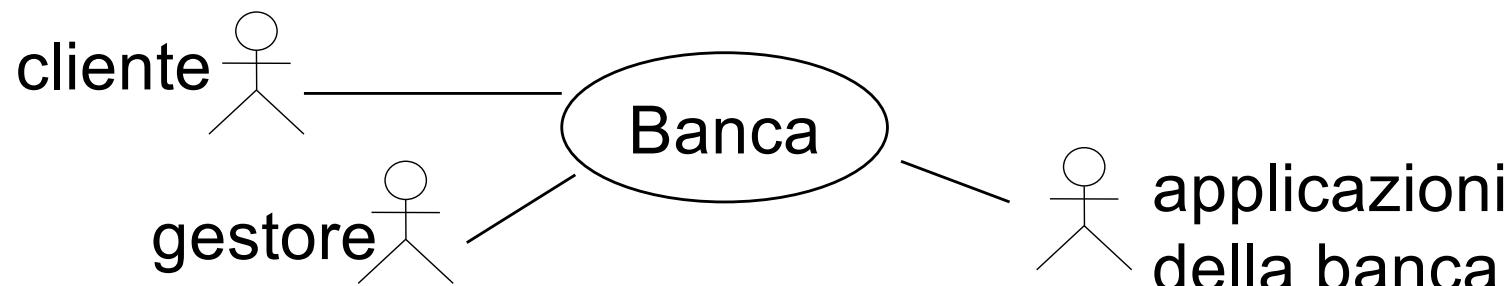
- uno indipendente da particolari soluzioni organizzative e tecnologiche (modello dell'ambito "business ")
- uno dipendente da una particolare soluzione organizzativa e tecnologica (modello dell'ambito dei servizi informatici)

Modellazione dell'ambito di business



Dal punto di vista del cliente, l'ufficio acquisizione ordini fa parte del sistema (come intermediario)

Modellazione dell'ambito dei servizi



casi d'uso

- **cliente:**
 - disposizioni (bonifici, acquisto titoli, ecc.)
 - informativa
 - stipula contratto
- **gestore:**
 - verifica anomalie

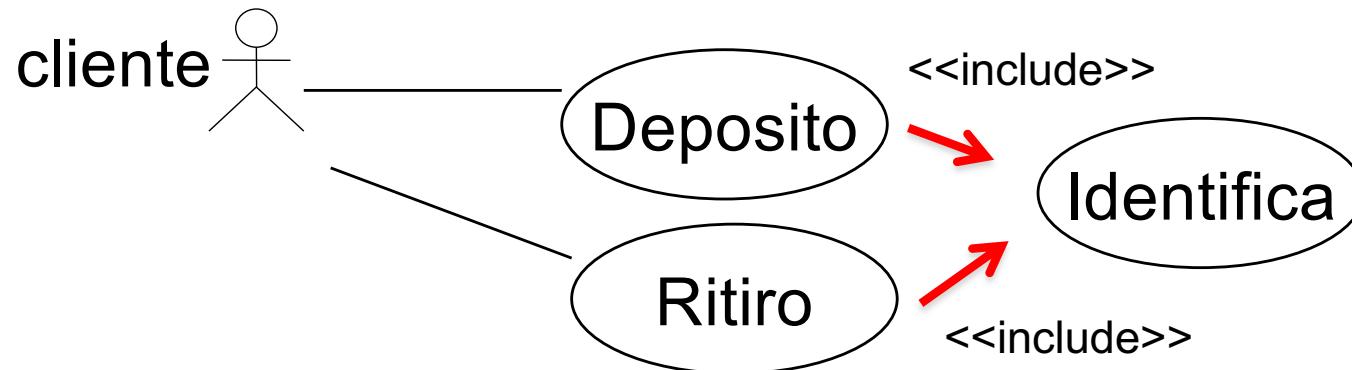
funzionalità interne sistema

- front-end specializzati
- front-end comune
- gestione pre-applicativa
- gestione contratti
- monitoraggio sistema

Relazioni speciali

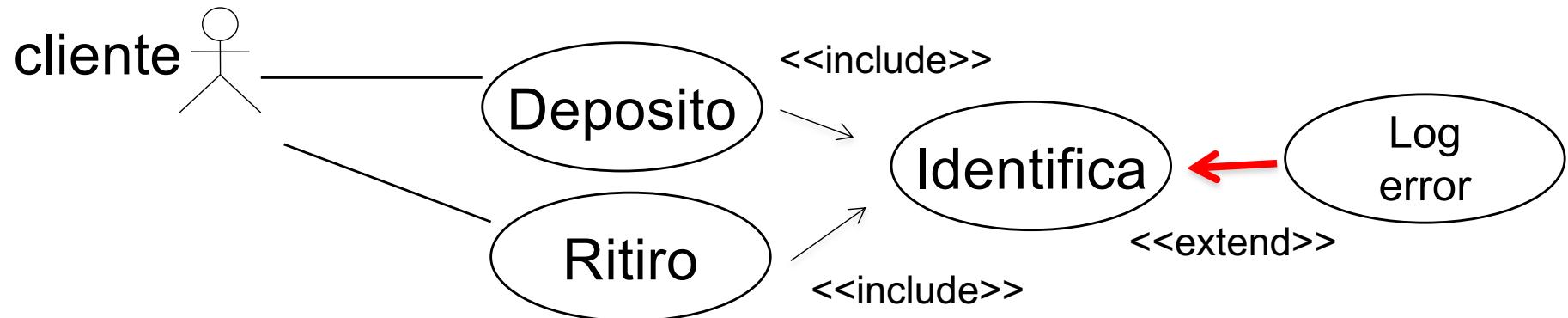
- Un caso d'uso “include” un altro caso d'uso
- Un caso d'uso “estende” un altro caso d'uso
- Un attore specializza un altro attore

Includere un caso d'uso



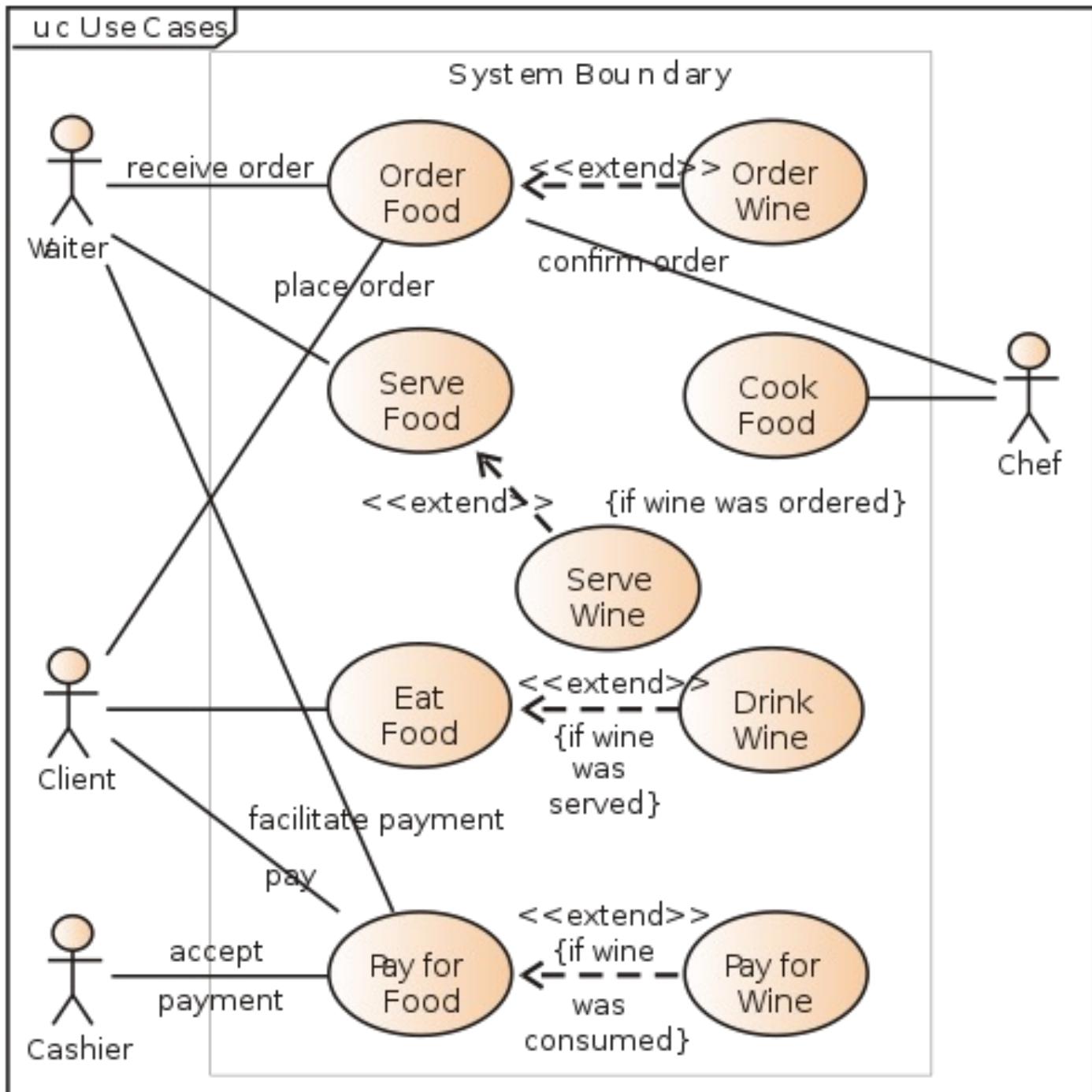
- Un caso d'uso “include” il comportamento di un altro caso d'uso
- (simile a “chiamata di procedura”)
- Notare il verso della freccia: “Deposito” e “Ritiro” includono “Identifica”

Estendere un caso d'uso



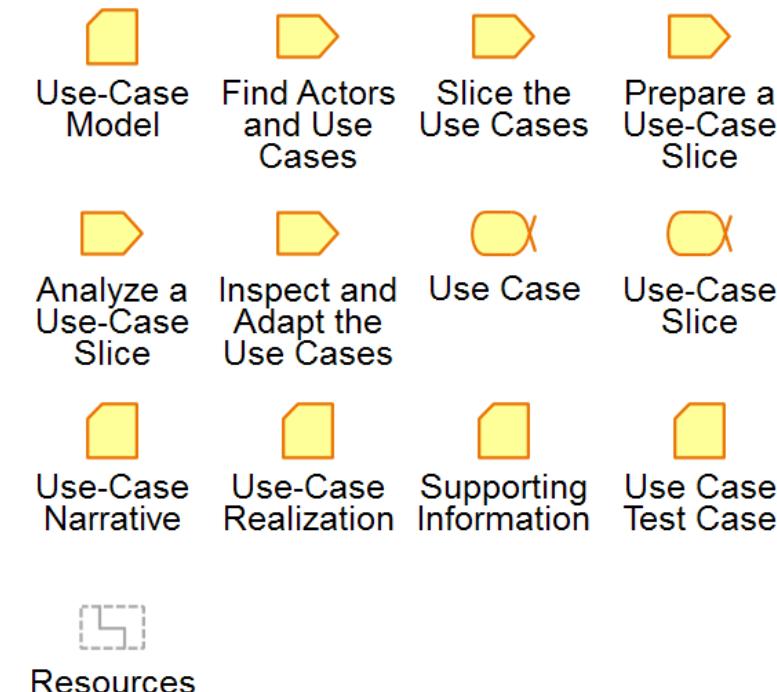
- Un caso d'uso “estende” il comportamento di un altro caso d'uso
- (simile a “funzione opzionale”)
- Notare il verso della freccia: “Log error” estende “Identifica”

Esempio



Use Case 2.0 Essentials

A scalable, agile practice that uses use cases to capture a set of requirements and drive the incremental development of a system to fulfill them.





Use Case

All the ways of using a system to achieve a particular goal for a particular user.



Goal Established

Story Structure Understood

Simplest Story Fulfilled

Sufficient Stories Fulfilled

All Stories Fulfilled

Relates to:  Requirements



IVAR JACOBSON
INTERNATIONAL
Generated by IJI Practice Workbench™

2018.09



Use-Case Narrative

Tells the story of how the system and its actors work together to achieve a particular goal.

Briefly Described

Bulleted Outline

Essential Outline

Fully Described

Describes:  Use Case and
 Use-Case Slice



IVAR JACOBSON
INTERNATIONAL
Generated by IJI Practice Workbench™

2018.09



Slice the Use Cases

Identify Use-Case Slices based on Use Cases to aid implementation.

- ❑ Use Case: Goal Established or beyond
 - ❑ Use-Case Narrative: Briefly Described or beyond



- ❑ Use Case: Story Structure Understood or beyond (contributes to)
 - ❑ Use-Case Narrative: Bulleted Outline or beyond (optional)
 - ❑ Use-Case Slice: Scoped (one or more)



Use-Case Slice

One or more stories selected from a use case to form a work item that is of clear value to the customer.



Relates to: ❑ Use Case

Riferimenti

- Wiegers, *Software Requirements Essentials*, Pearson, 2023
- Beatty & Chen, *Visual Modeling for Software Requirements*, Microsoft Press, 2012
- Cockburn, *Writing Effective Use Cases*, Addison-Wesley, 2001
- Leffingwell and Widrig, *Managing Software Requirements: A Use Case Approach*, Addison-Wesley, 2003

Domande?

