

# Scrivere (e leggere) i requisiti di un prodotto software



*Prof. Paolo Ciancarini*  
*Corso di Ingegneria del Software*  
*CdL Informatica 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
- Analisi e classificazione dei requisiti
- Strumenti di gestione dei requisiti

# 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!



# Ma cos'è un desiderio?



# Come si esprimono i desideri?

## 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:

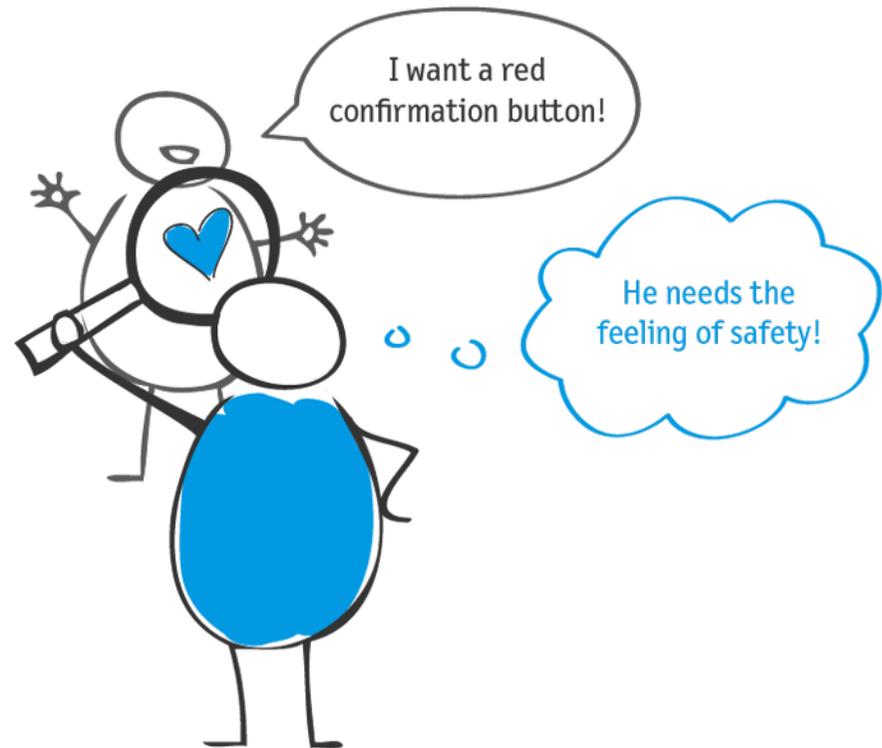
*Paolo **vuole** poter chiamare un taxi da una app e ottenere risposta con stima del tempo di arrivo, **per** calcolare se arriverà in tempo al suo appuntamento*

# Ma cos'è un desiderio? E un bisogno?



# Bisogni, desideri, requisiti

- I bisogni sono necessità insopprimibili (il bisogno di mangiare, dormire, respirare)
- I desideri sono invece sostituibili da altri desideri
- I requisiti sono descrizioni di funzionalità che vorremmo avere nei prodotti che usiamo



# Legge di Humphrey

*I clienti non sanno quello che stanno cercando finché non lo vedono.*

Potrebbero dirvi che hanno bisogno della funzione X. Non appena gli mostrate la funzione X, subito dopo capiranno che quello che vogliono in realtà è un'altra funzione Y o una diversa variante della funzione X

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

# Scopo e funzione

- I desideri o bisogni di un cliente possono essere espressi come **scopi**, cioè come obiettivi
  - Esempio: desidero viaggiare, ho bisogno di pianificare un tragitto in treno e comprare il biglietto
- Nel progettare un sistema che permetta di conseguire uno scopo dovremo specificare **una serie di funzioni**
- La relazione tra scopi e funzioni che li conseguono è l'oggetto dell'**analisi dei requisiti**

# Analisi dei requisiti (Wikipedia)

L'**analisi dei requisiti** è un'attività preliminare allo sviluppo (o alla modifica) di un sistema software, e serve a definire le funzionalità che il nuovo prodotto (o il prodotto modificato) deve 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.

L'analisi dei requisiti è la prima fase del processo di sviluppo, e deve concludersi con la stesura di una dettagliata **specificazione dei requisiti** che descrive le funzionalità del nuovo software nella loro interezza;

tale specifica guida le fasi successive di sviluppo, che complessivamente sono volte a realizzare quanto previsto da tale specifica

# 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.**

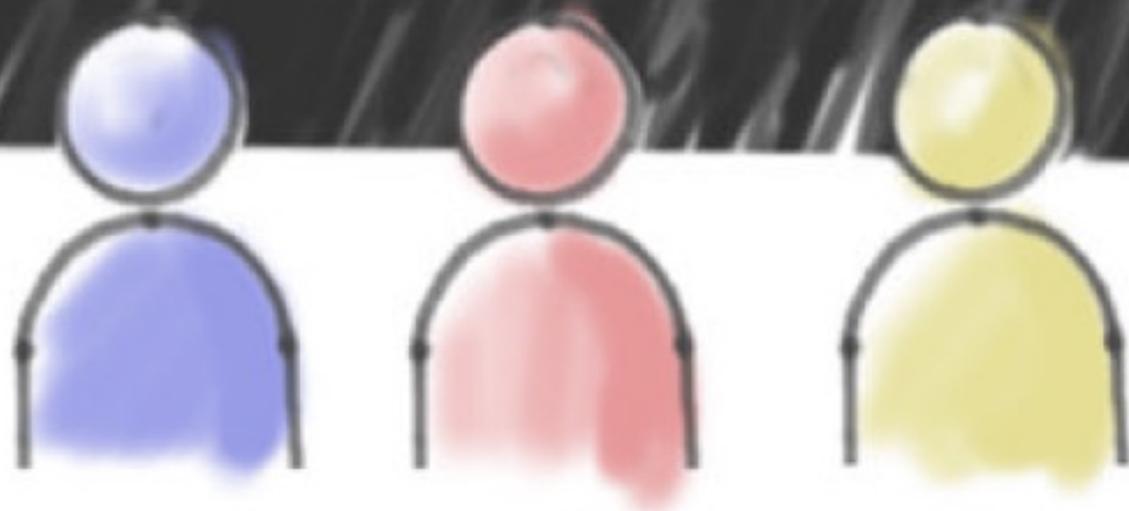
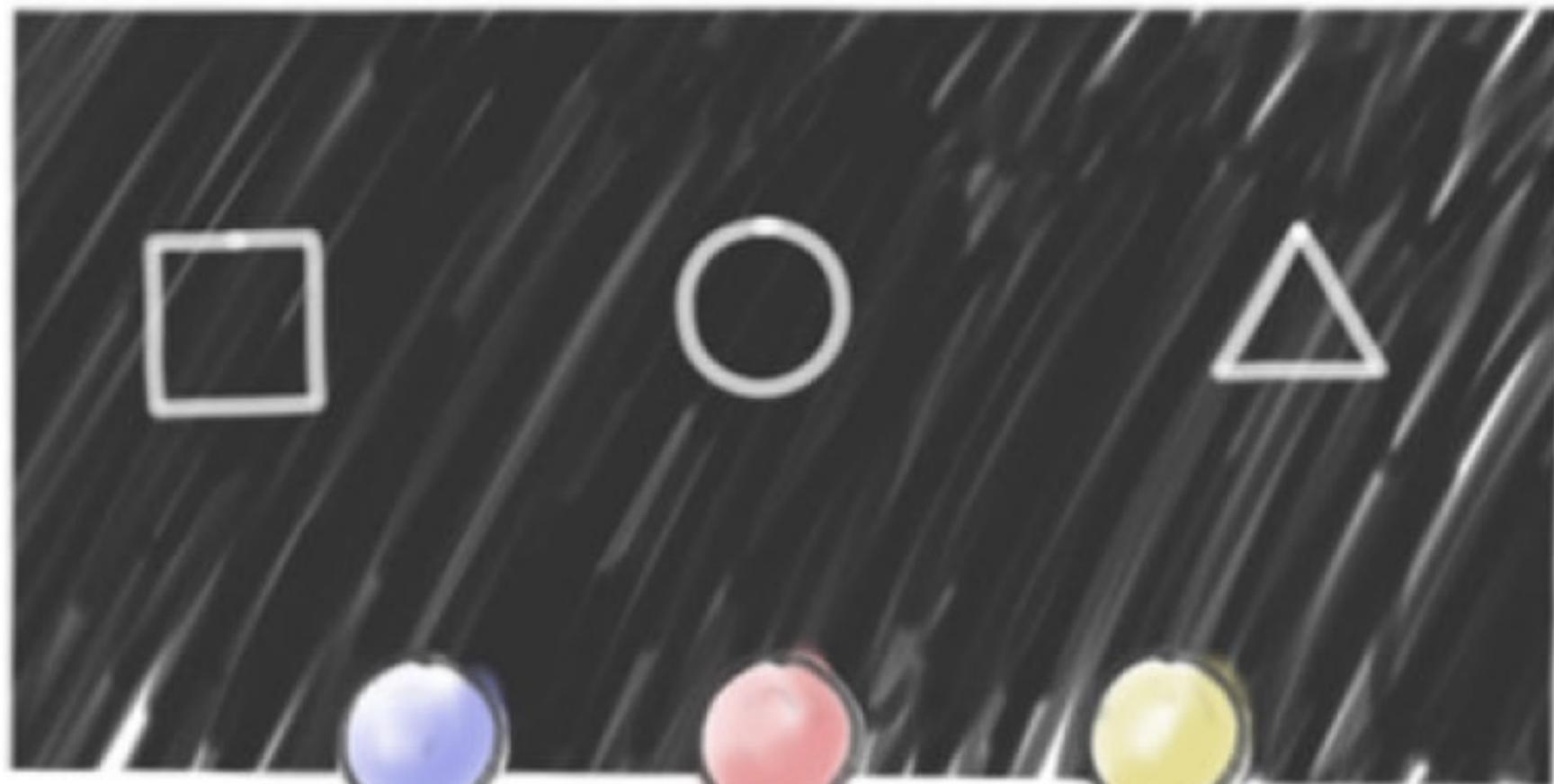
# Discussione

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

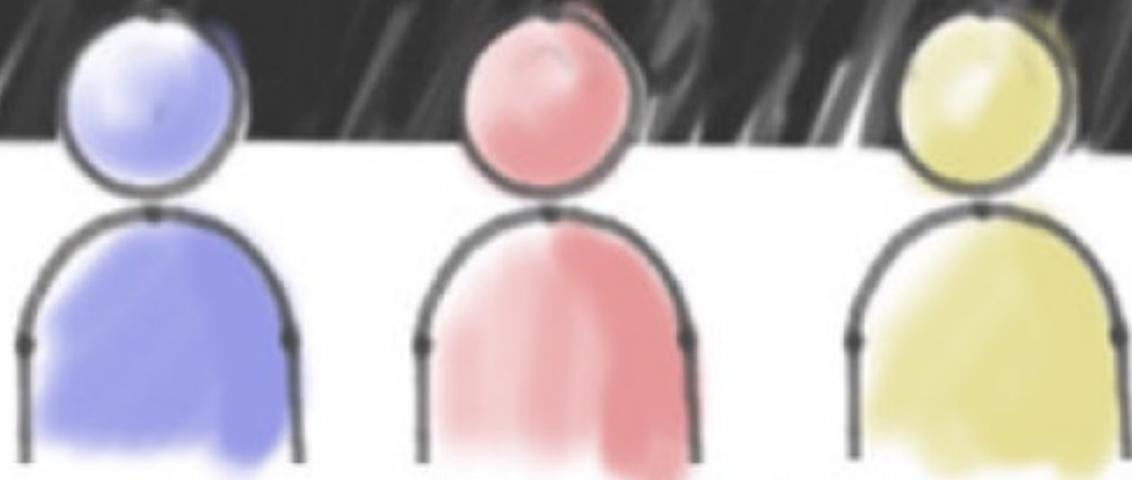




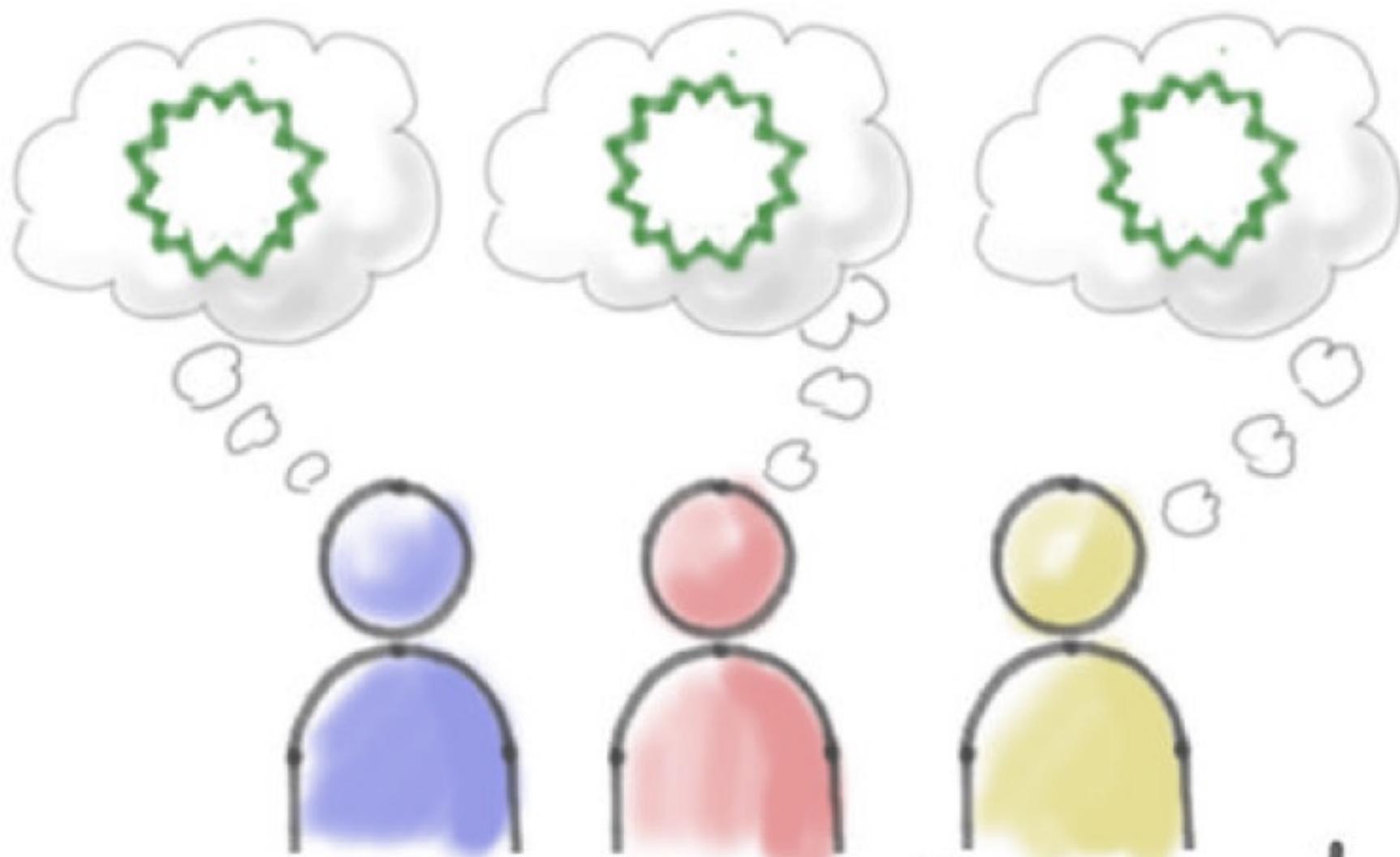
*I'm glad we all agree.*



oh...



ah ha!



*I'm glad we all agree!*

# 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

1. Il sistema gestirà due tipi di utenti: il bibliotecario e l'iscritto
2. Il sistema permetterà al bibliotecario di aggiornare la base di dati
3. Il sistema permetterà al bibliotecario di accedere mediante password alle operazioni d'aggiornamento
4. Il sistema permetterà all'iscritto di consultare i dati dei libri
5. Il sistema permetterà all'iscritto di accedere liberamente alle operazioni di consultazione
6. Il sistema fornirà a tutti un'interfaccia Web standard utilizzabile anche da casa.
7. Il sistema permetterà ad un iscritto di chiedere il prestito di uno o più volumi alla volta mediante un Web browser
8. Il sistema su richiesta invia all'iscritto la lista dei volumi disponibili.
9. Il sistema identificherà gli iscritti con un codice numerico, e col cognome, nome e data di nascita
10. . (continua) ...

Una descrizione funzionale è un requisito

- $F(0) = 1$
- $F(1) = 1$
- $F(n) = F(n-1) + F(n-2)$

Queste formule sono requisiti: descrivono una funzione che calcola  $\text{Fibonacci}(n)$ , e si possono usare per scrivere un programma ricorsivo

# Requisiti

- I requisiti sono una specifica di ciò che va implementato
- 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

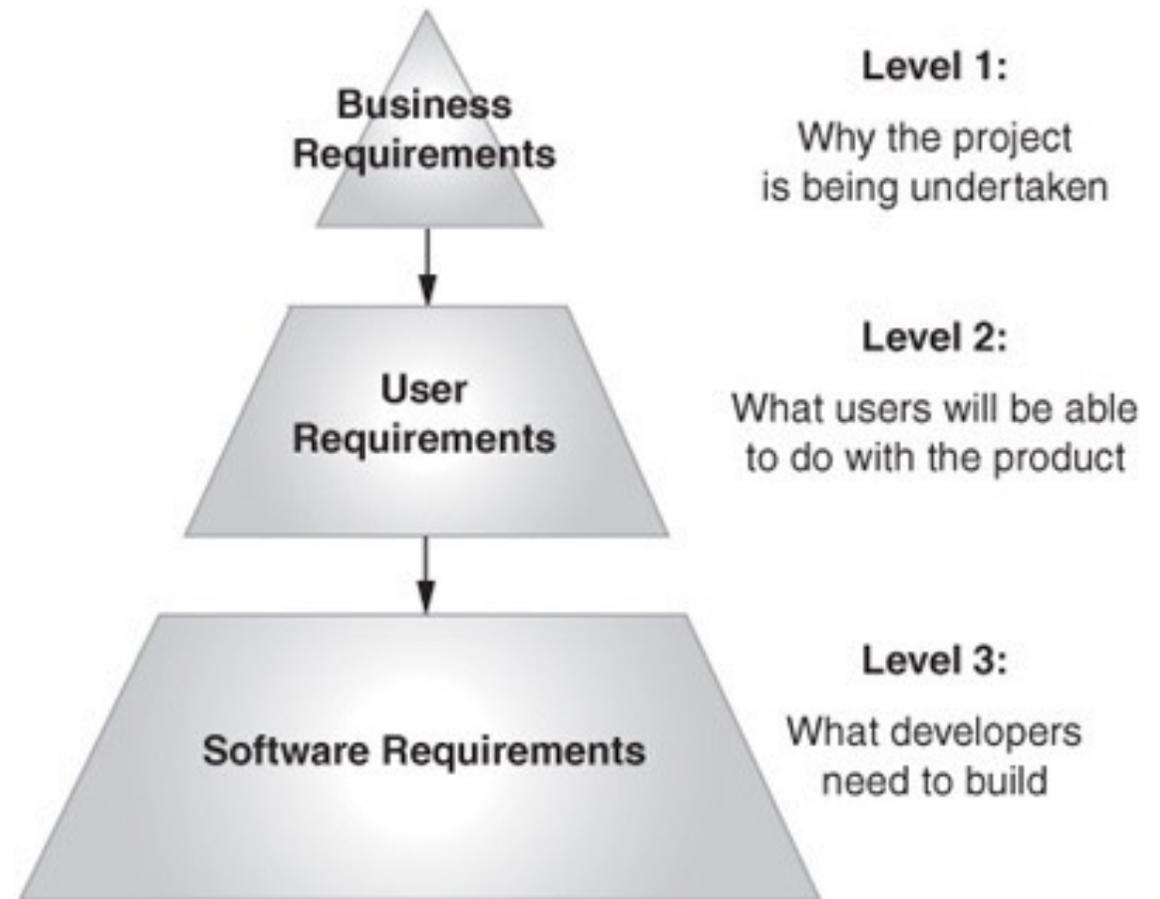
# Requisito

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

Lo «scope» del requisito dice <chi> vuole <cosa> e <perché>

I requisiti possono essere:

- di business
- di utente
- del software

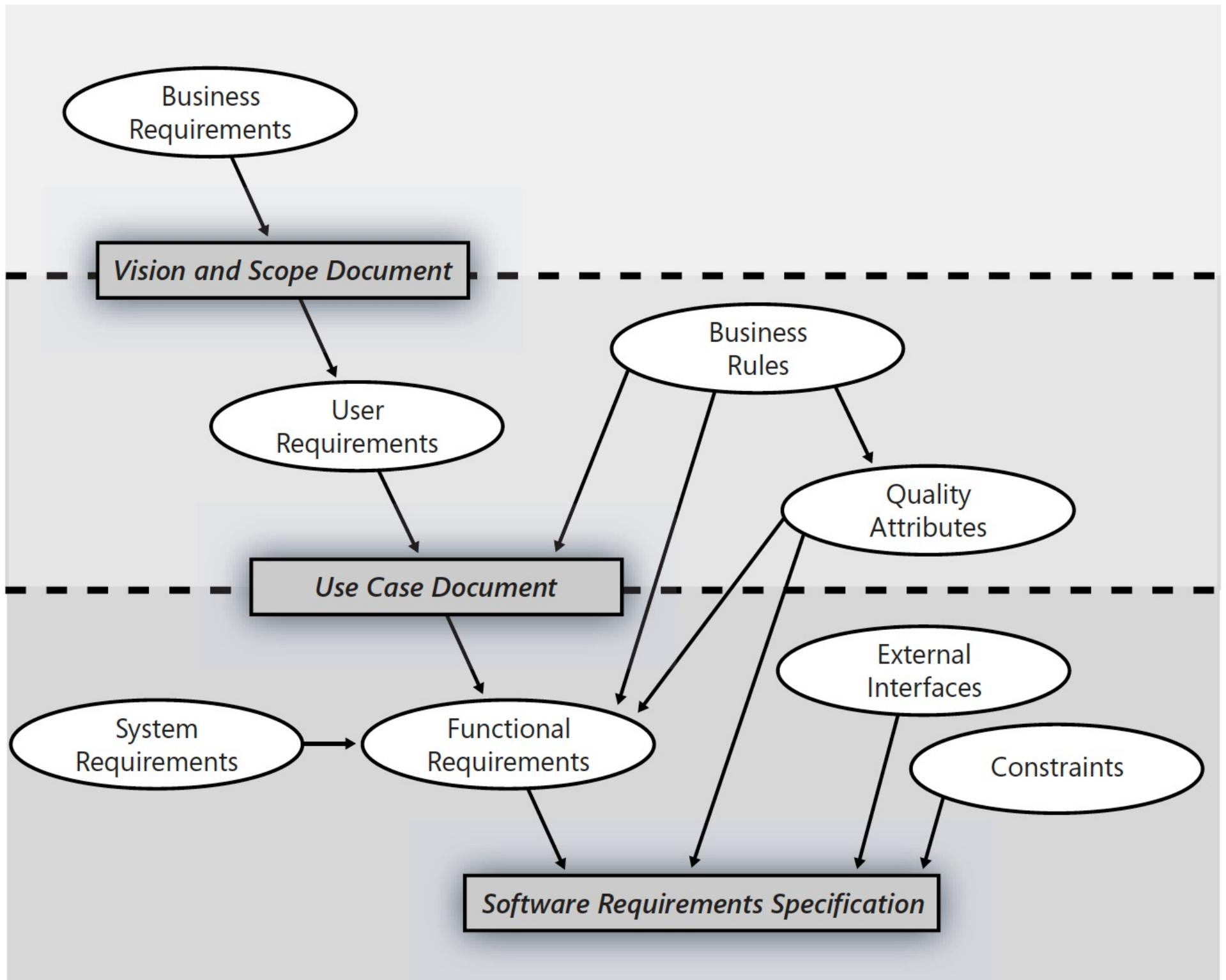


# Esempi

Requisito di business: *l'ateneo deve gestire le carriere di 100K studenti*

Requisito utente: *come studente voglio vedere la mia carriera per poter capire quali esami mi mancano alla laurea*

Requisito software: *il record Studente conterrà il campo Carriera di tipo vettore di stringhe*



# Discussione

Come definiamo i requisiti funzionali di un sistema?



# La forma del requisito

- **Tradizionale:** *«L'applicazione gestirà i prestiti dei volumi agli iscritti»*
- **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 le informazioni di ricerca, ...»*

I requisiti con le user stories

# La forma agile: le user stories

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

Come *<tipo di utente>*

Voglio *<fare qualcosa>*

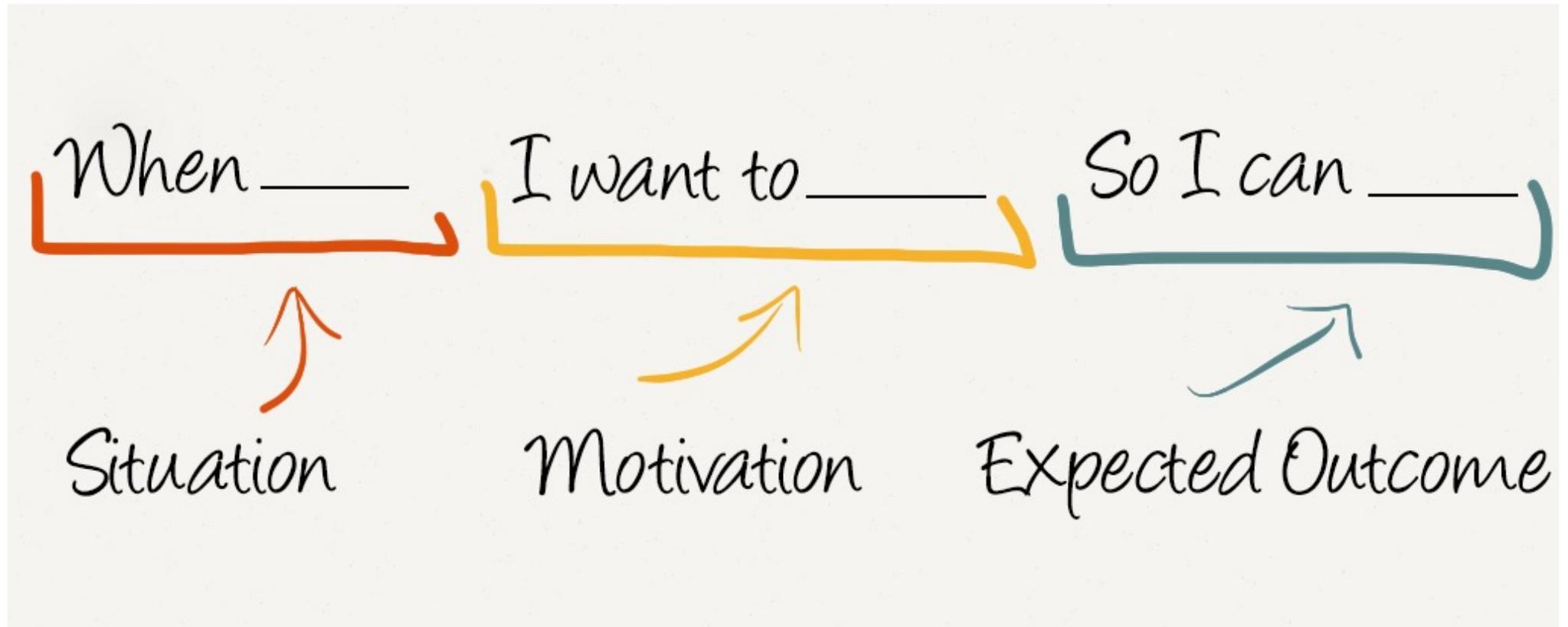
Per *<conseguire un obiettivo>*

Come *utente della biblioteca* (ruolo)

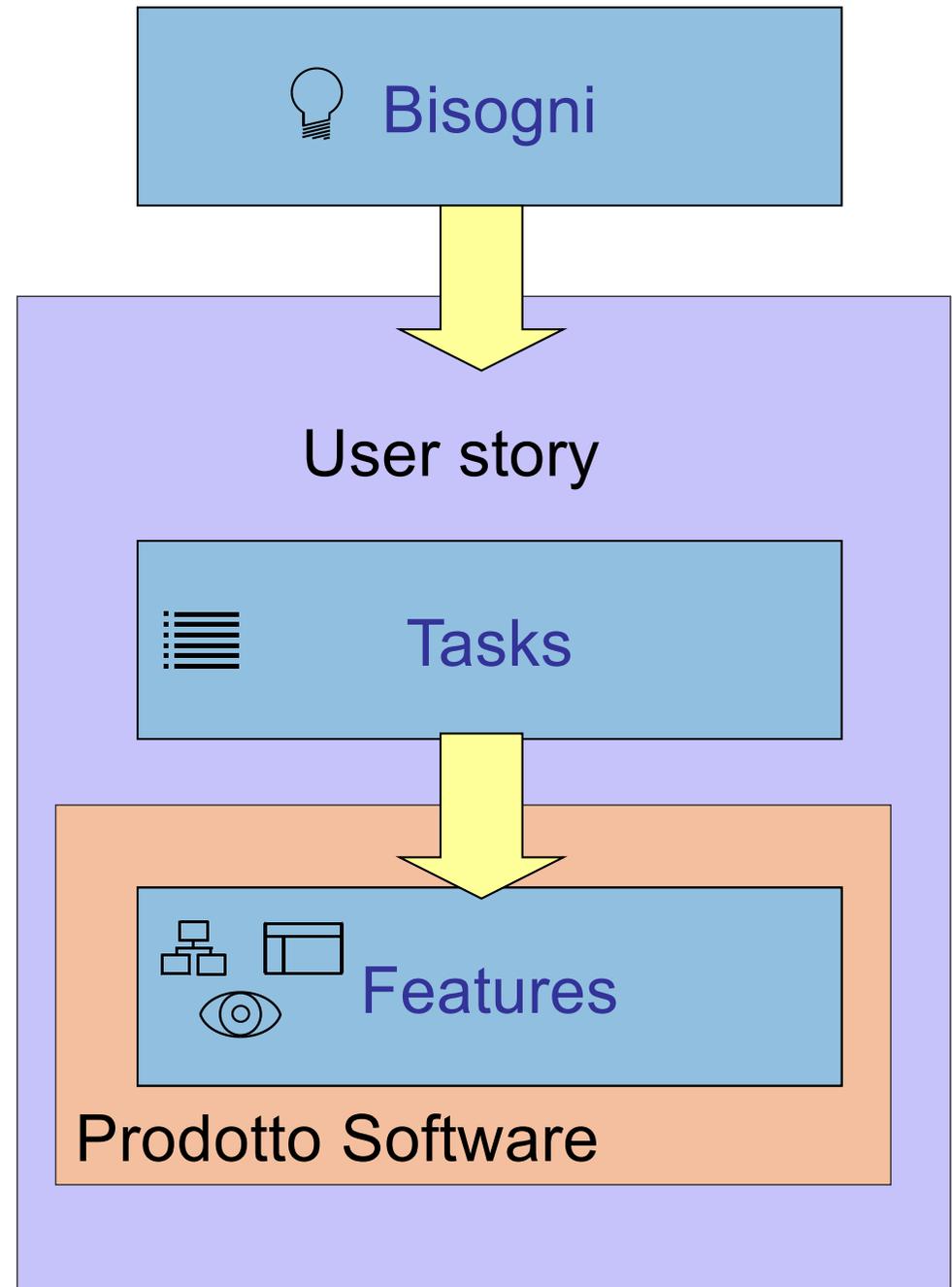
Voglio *consultare il catalogo* (scopo o funzione)

Per *prendere a prestito un volume* (valore ottenuto)

# Struttura di una user story



Dalle user stories al prodotto



# User story: esempio con condizioni

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

Potremmo aggiungere vari dettagli sulle condizioni di svolgimento della user story:

- 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

# User story: esempio con test

- Come utente del portale voglio iscrivermi inserendo il mio CV e lo stipendio desiderato 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 stories difettose: esempi

ID	Description	Issues
US <sub>1</sub>	As a User, I'm able to click a particular location from the map and thereby perform a search of landmarks associated with that latitude longitude combination	Not atomic: two stories in one
US <sub>2</sub>	As a care professional I want to see the registered hours of this week (split into products and activities). See: Mockup from Alice NOTE: - First create the overview screen - Then add validations	Not minimal, due to additional note about the mockup
US <sub>3</sub>	Add static pages controller to application and define static pages	Missing role
US <sub>4</sub>	As a User, I want to open the interactive map, so that I can see the location of landmarks	Conceptual issue: the end is in fact a reference to another story
US <sub>5</sub>	As a User, I'm able to edit any landmark	Conflict: US <sub>5</sub> refers to any landmark, while US <sub>6</sub> only to those that user has added
US <sub>6</sub>	As a User, I'm able to delete a landmark which I added	
US <sub>7</sub>	As a care professional I want to save a reimbursement. - Add save button on top right (never greyed out)	Hints at the solution
US <sub>8</sub>	As a User, I am able to edit the content that I added to a person's profile page	Unclear: what is content here?
US <sub>9</sub>	As an Administrator, I am able to view content that needs to be reviewed	The type of content is not specified
US <sub>10</sub>	Server configuration	In addition to being syntactically incorrect, this is not even a full sentence
US <sub>11</sub>	As an Administrator, I am able to add a new person to the database followed by	Viewing relies on first adding a person to the database
US <sub>12</sub>	As a Visitor, I am able to view a person's profile	
US <sub>13</sub>	As a care professional I want to see my route list for next/future days, so that I can prepare myself (for example I can see at what time I should start traveling)	Difficult to estimate because it is unclear what see my route list implies
US <sub>14</sub>	As an Administrator, I receive an email notification when a new user is registered	Deviates from the template, no "wish" in the means
EP <sub>A</sub>	As a Visitor, I'm able to see a list of news items, so that I can stay up to date on news	The same requirement is repeated both in epic EP <sub>A</sub> , and in a user story US <sub>14</sub>
US <sub>15</sub>	As a Visitor, I'm able to see a list of news items, so that I can stay up to date on news	

# A cosa servono le user stories

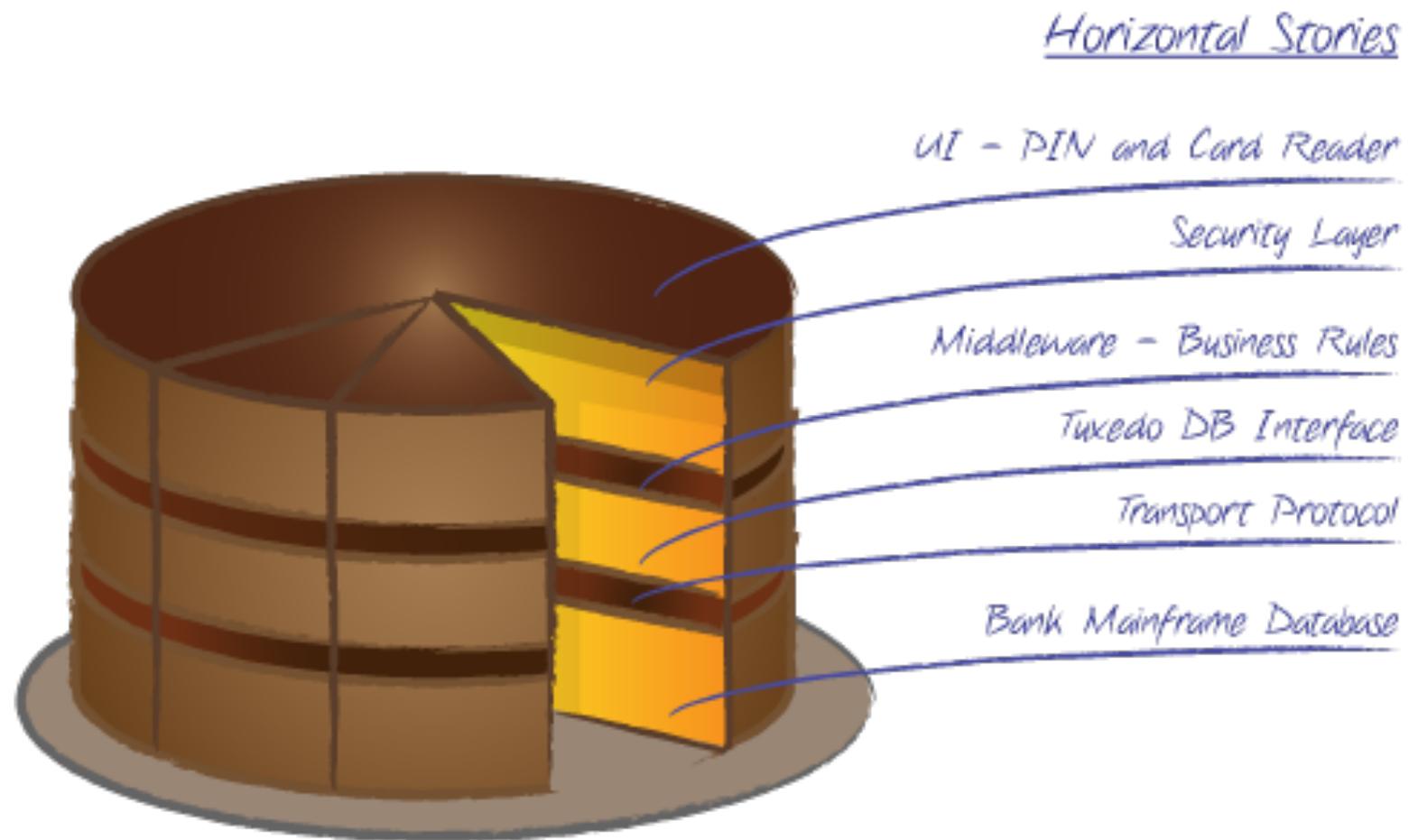
- Il Product Owner scrive le storie e la loro “Definition of Done”, cioè il criterio di accettazione
- Le storie debbono essere sufficientemente piccole da poter essere realizzate in uno o due settimane
- Il team di sviluppo sceglie una storia da realizzare, definisce i test e prepara tutto il necessario perché il PO possa accettare il software risultante, così come richiesto dalla Definition of Done

# 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)

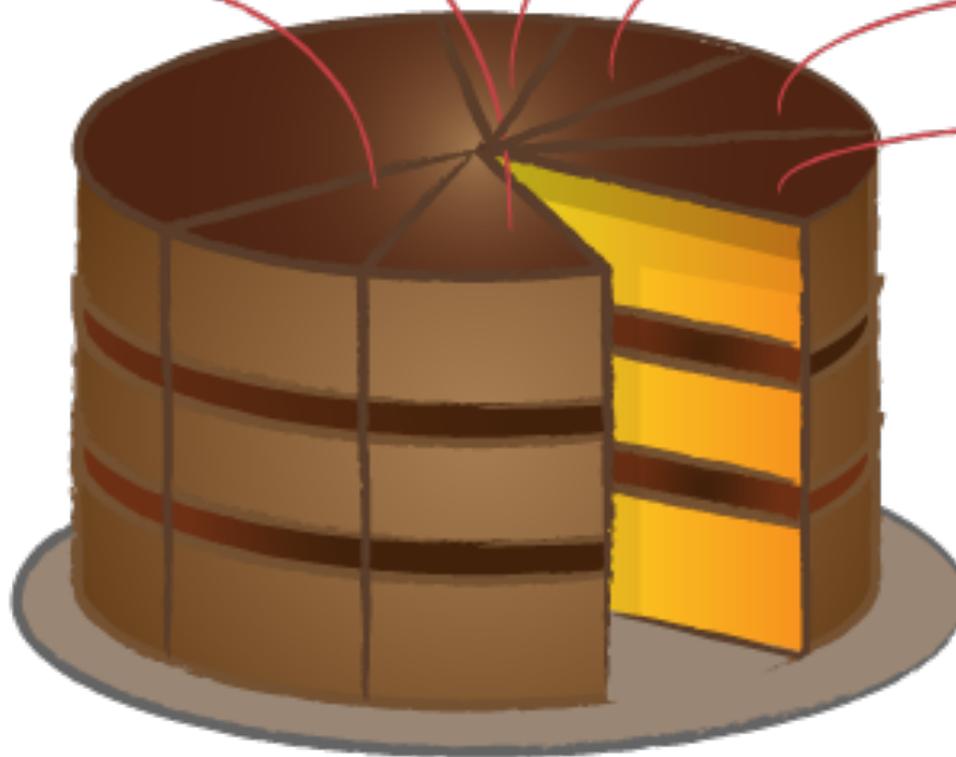
Printing Bank Statements

Cash Deposit

Cheque Deposit

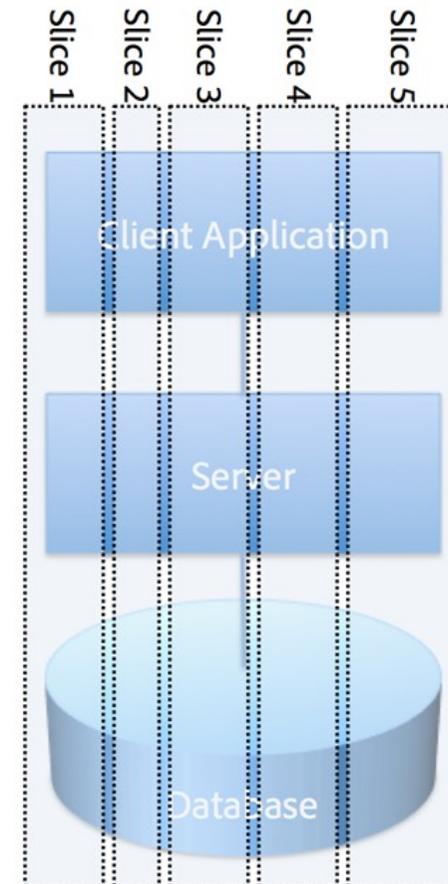
Bill Payment

Purchase Gift Certificates



# Vantaggi delle storie verticali

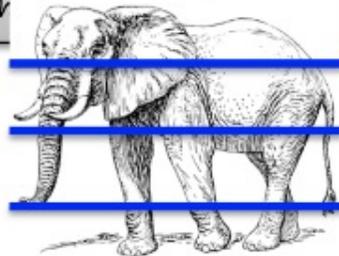
- **Ambiguità** La disambiguazione di termini vaghi può aiutare l'articolazione delle storie
- **Congiunzioni** Una funzione descritta mediante una congiunzione può facilmente essere decomposta
- **Accettazione** Il test di accettazione può indurre ad affettare una storia così da renderla più facilmente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



# Storie verticali vs orizzontali: esempio

## Vertical Slicing/Laminating

As a student, I want to purchase a parking pass so that I park close to school



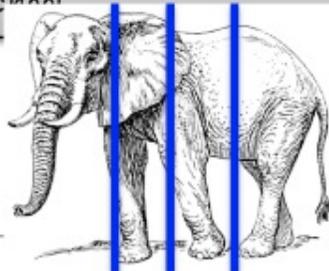
Integrate parking pass into datamodel

Create connection to payment service

Design GUI for purchase process

Define parking pass validation scheme

As a student, I want to purchase a parking pass so that I park close to school



Implement purchase process without payment

Integrate payment into the purchase process

Allow users to see their past orders

Define parking pass validation scheme

## 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

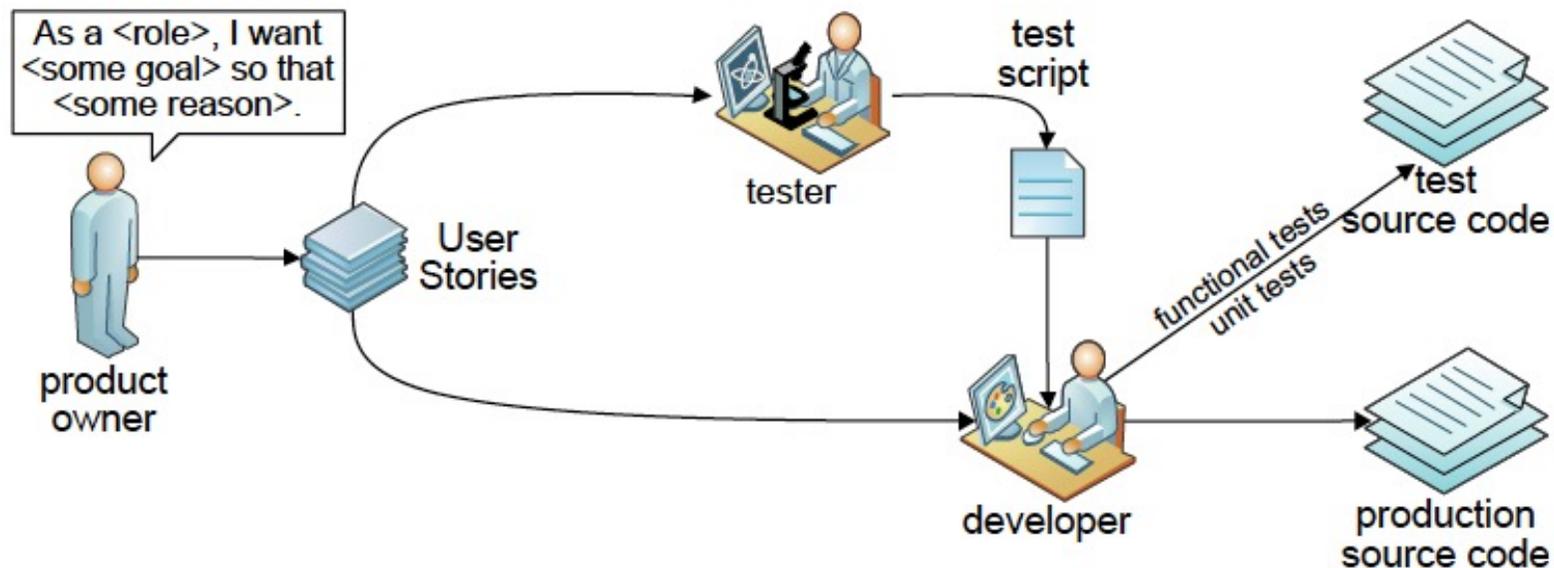
- Standard Login
- Reset Password
- Sign In - Authenticate user
- Password Requirement help
- In-line Field validation
- Logout Function
- 2-factor Auth
- Facebook Login
- Face ID login

## Le user story sono frammenti di una conversazione

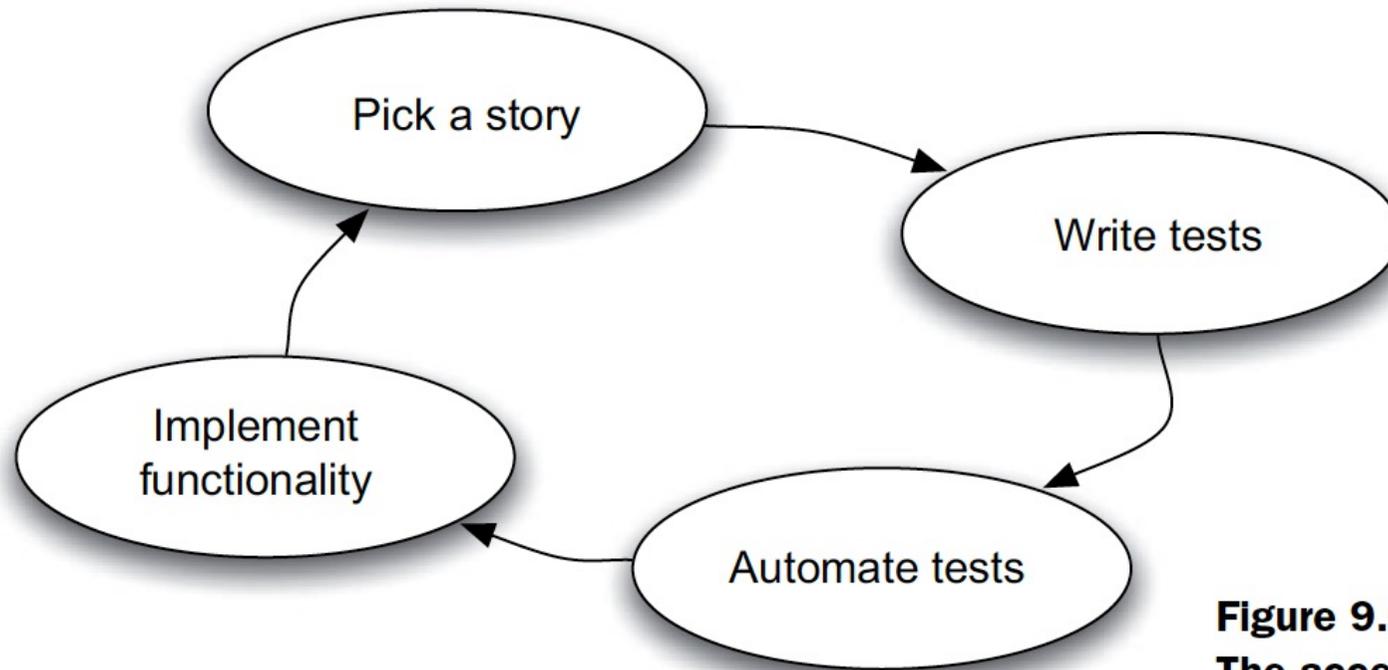
- Utente: come descrivo i miei desideri?
- Stakeholder: come posso far sì che il prodotto abbia successo?
- PM: come traccio e pianifico questo compito?
- BA: quali sono i dettagli di questa feature?
- UX: quali sono i bisogni dell'utente?
- Developer: quali task devo eseguire oggi?
- QA: come posso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

# User story nel processo agile

- As a <user\_type>
- I want to <function>
- so that I can <business value>
- (acceptance criteria of the business value)



# User story e Test Driven Development



**Figure 9.6**  
**The acceptance TDD cycle**

# Gherkin

Gherkin è un linguaggio testuale parte di Cucumber, uno strumento di test. Gherkin permette di descrivere test su casi d'uso senza dover dare dettagli sull'implementazione. Gherkin usa l'indentazione

**Feature:** Serve coffee

Coffee should not be served until paid for

Coffee should not be served until the button has been pressed

If there is no coffee left then money should be refunded

**Scenario:** Buy last coffee

**Given** there are 1 coffees left in the machine

**And** I have deposited 1\$

**When** I press the coffee button

**Then** I should be served a coffee

# Gherkin

Gherkin è un Domain Specific Language per scrivere test di accettazione che ha cinque costrutti principali:

- **Scenario** — etichetta del comportamento che descriviamo
- **Given** — stato iniziale dello scenario
- **When** — azione effettuata dall'utente
- **Then** — risultato testabile, di solito causato dall'azione indicata dal costrutto When
- **And** — usabile in uno qualsiasi dei tre operatori Given When Then

# Esempio

Scenario: User clicks the link

Given I am on the homepage    **precondizione**

When I click the provided link    **azione**

Then I should see the link click confirmation    **risultato testabile**

## Requirements

1 Work on one example at a time.

```
- password should be at least 8 characters  
- password should contain at least 1 digit  
- password should contain at least 1 PUNCTUATION  
MARK  
- I should get an error message telling me what I  
did wrong if I enter an insecure password
```

2 Have a conversation about the example.



## BDD

3 Write an automated scenario.

```
Scenario: Transferring money to  
Given my Current account has a  
And my Savings account has a ba  
When I transfer 500.00 from my  
Then I should have 500.00
```



4 The automated scenario should fail.



7 Move on to the next scenario.

```
- password should be at least 8 characters  
- password should contain at least 1 digit  
- password should contain at least 1 PUNCTUATION  
MARK  
- I should get an error message telling me what I  
did wrong if I enter an insecure password
```



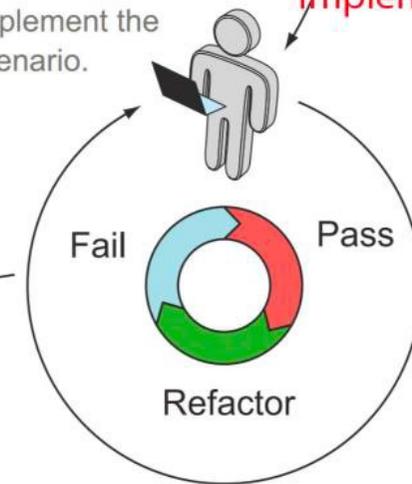
8 You have finished the story when all the examples are automated and pass.

6 You are done with this scenario when the automated scenario passes.



5 Implement the scenario.

## Implementation



BDD unit tests drive the development.

I requisiti coi casi d'uso

# I requisiti con UML: i casi d'uso

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



Ivar Jacobson

# Scenari

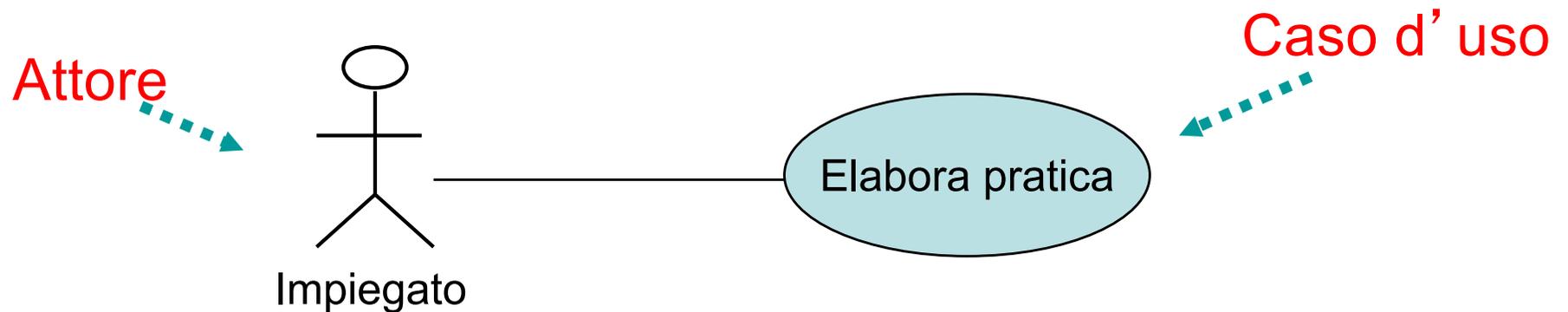
Un caso d'uso si descrive sotto forma di **scenario di interazione** (cioè un dialogo) tra un utente e il sistema

## **Esempio:**

- il cliente richiede l'elenco dei prodotti
- il sistema propone i prodotti disponibili
- il cliente sceglie i prodotti che desidera
- il sistema fornisce il costo totale dei prodotti selezionati
- il cliente conferma l'ordine
- il sistema comunica l'accettazione dell'ordine
- L'attenzione si focalizza sull'interazione, non sulle attività interne al sistema

# Rappresentazione grafica

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

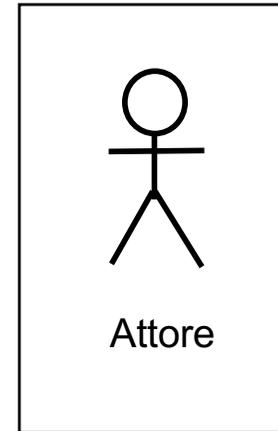
**display order  
details**

**error message:  
order number  
not found**

# Formato di uno scenario

<b>Use Case ID:</b>			
<b>Use Case Name:</b>			
<b>Created By:</b>		<b>Last Updated By:</b>	
<b>Date Created:</b>		<b>Date Last Updated:</b>	
<b>Actors:</b>			
<b>Description:</b>			
<b>Trigger:</b>			
<b>Preconditions:</b>			
<b>Postconditions:</b>			
<b>Normal Flow:</b>			
<b>Alternative Flows:</b>			
<b>Exceptions:</b>			
<b>Includes:</b>			
<b>Priority:</b>			
<b>Frequency of Use:</b>			
<b>Business Rules:</b>			
<b>Special Requirements:</b>			
<b>Assumptions:</b>			
<b>Notes and Issues:</b>			

# 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)!

# Come trovare gli attori

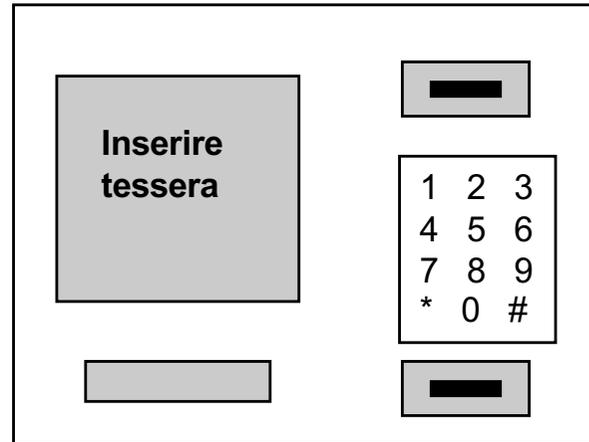
- Chi o cosa è interessato al sistema?
- Chi o cosa modificherà i dati del sistema?
- Chi o cosa vuole informazioni dal sistema?
- Chi o cosa si interfaccerà col sistema?

# 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?

# Istanze di attori

Paolo  
agisce  
come  
attore



Laura  
agisce  
come  
attore

Use-Case Model



Attore



Caso d'uso

# Caso d'uso

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

Si usa per

- Descrivere requisiti d'utente (analisi iniziale)
- Controllare il sistema (testing e verifica)

# Caso d'uso

- Ogni sequenza (detta *scenario*) rappresenta l'interazione di entità esterne al sistema (*attori*) con il sistema stesso o sue componenti
- Separa il *flusso principale* dalle varianti *alternative*

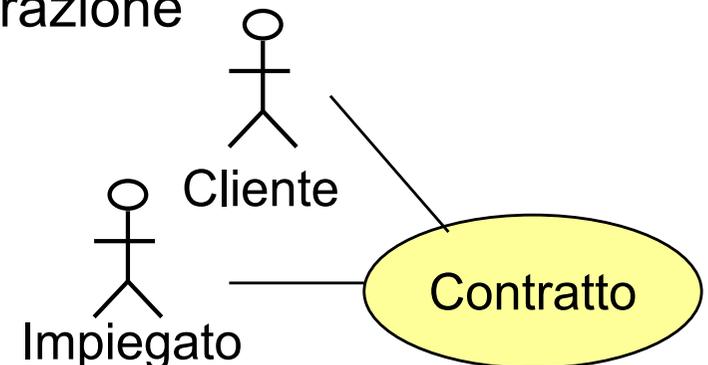
# Esempio

**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
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 mantengono il sistema?
- I requisiti funzionali sono tutti coperti dai Casi d'Uso che abbiamo trovato?

# Fonti di informazione per i casi d'uso

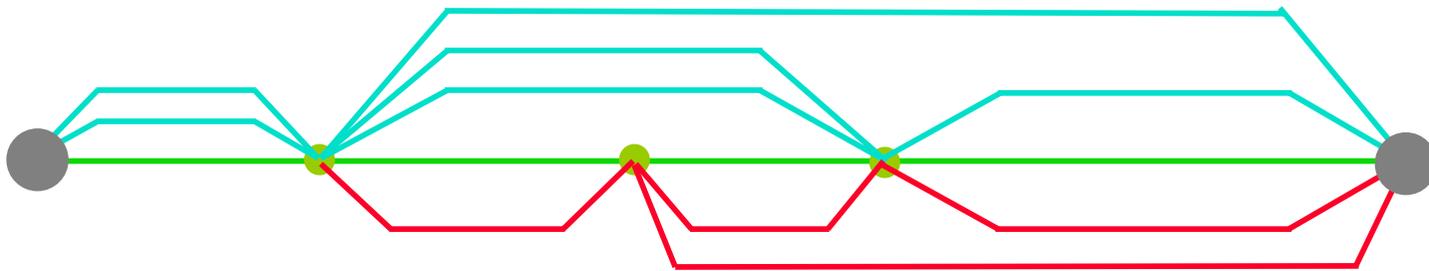
- Documenti di specifica del sistema
- Bibliografia del dominio del sistema
- Interviste con gli esperti del dominio
- Conoscenza personale del dominio
- Sistemi simili già esistenti

# 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 dovrebbe essere come un dialogo tra l'attore e lo use case
- Tutti i documenti sono scritti in modo comprensibile per il cliente

# Flusso degli eventi

- Ogni caso d'uso
  - Ha una sequenza di transizioni normale o di base
  - Può avere varie sequenze alternative
  - Ha varie sequenze di transazioni eccezionali per la gestione di situazioni erranee



# Flusso degli eventi

- Descrive solo gli eventi relativi al caso d'uso, e non quel che avviene in altri casi 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

# Casi d'uso e scenari

**Scenario base:** è di solito 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

# Come scrivere un caso d'uso

- Assegnare un nome al caso d'uso
- Descrivere gli attori
- Descrivere la condizione iniziale
- Flusso degli eventi
- Descrivere la condizione d'uscita
- 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

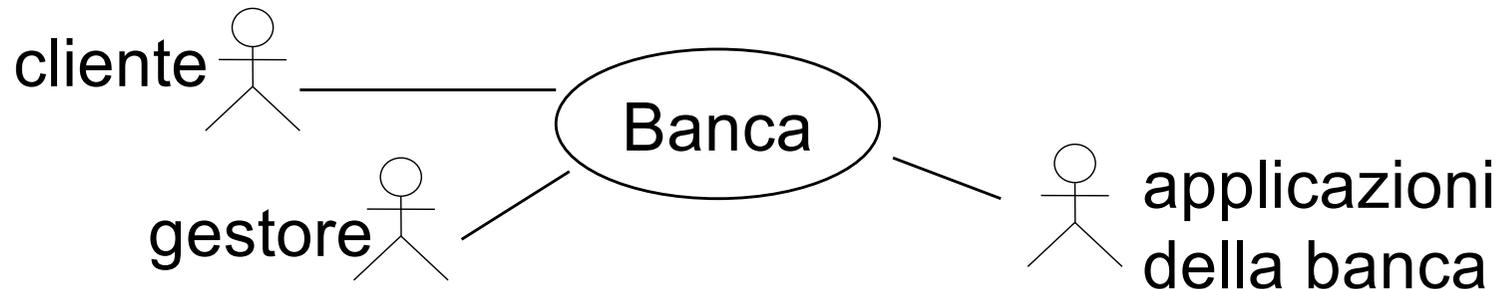
(5) l'addetto crea il nuovo conto corrente

- 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

# Dalle funzionalità interne ai casi d'uso



## casi d'uso

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

## funzionalità interne sistema

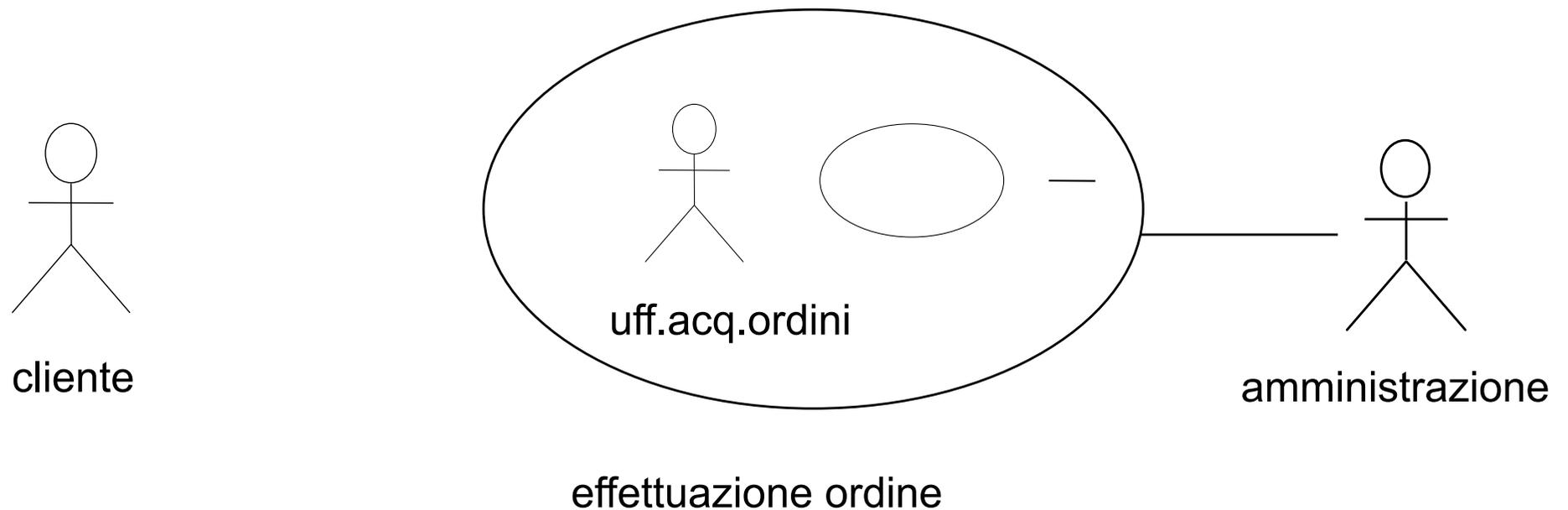
- front-end specializzati
- front-end comune
- gestione pre-applicativa
- gestione contratti
  
- monitoraggio sistema

# Attori “business” e “di servizio”

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

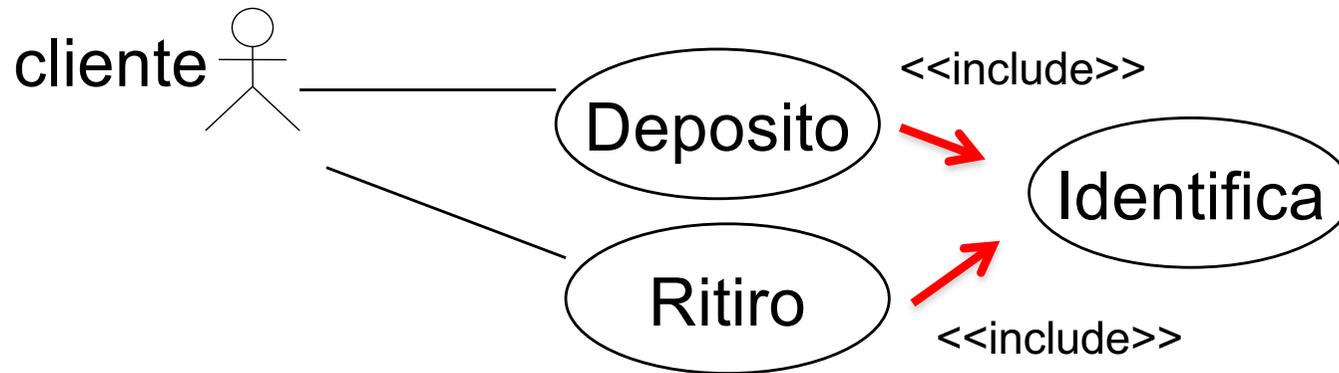
- uno indipendente da particolari soluzioni organizzative e tecnologiche (modello dell'ambito "business workflow")
- uno legato ad una particolare soluzione organizzativa e tecnologica (modello dei servizi del sistema informatico)

# Modellazione dell'ambito di business



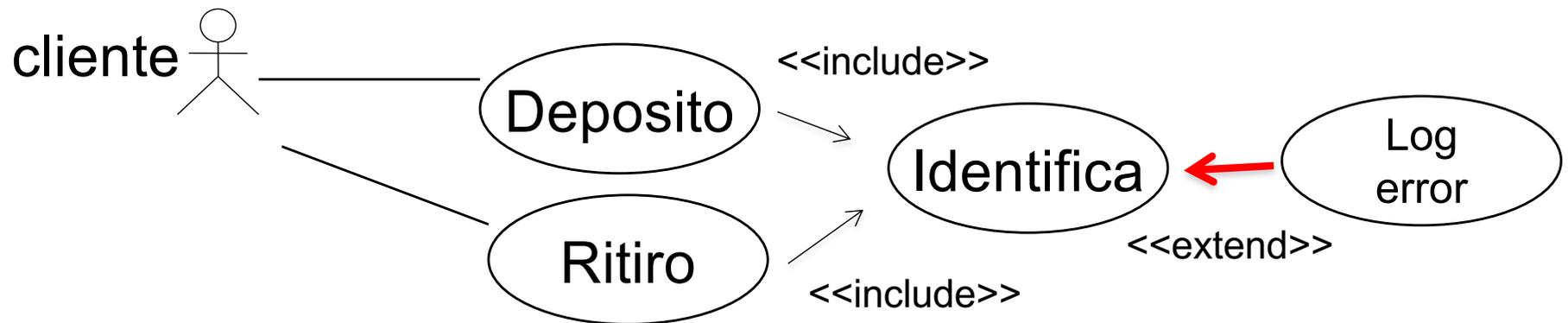
Dal punto di vista del cliente, l'ufficio acquisizione ordini fa parte del sistema (come intermediario)

# Includere un caso d'uso



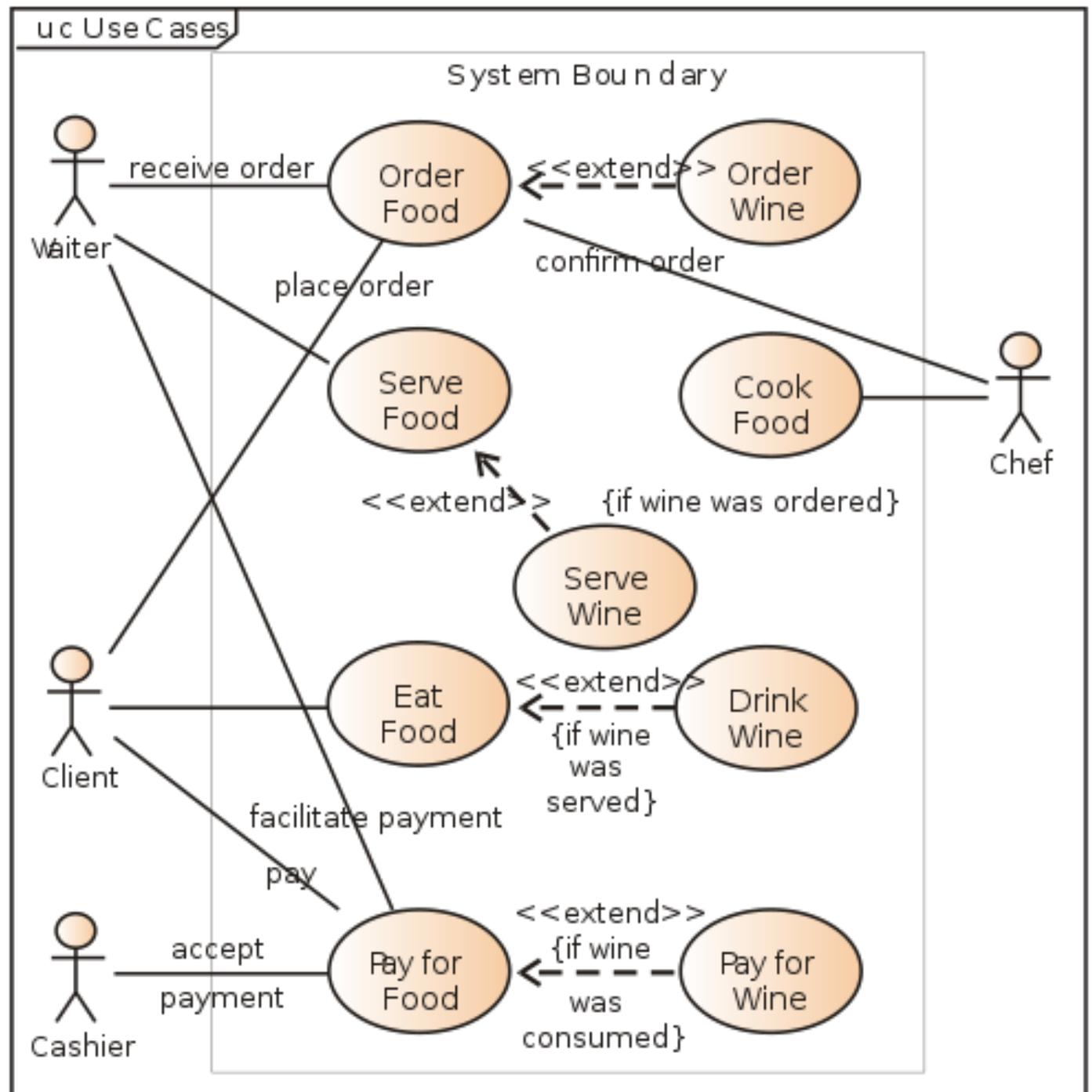
- Un caso d'uso "include" il comportamento di un altro caso d'uso
- (simile a "chiamata di procedura")
- Notare il verso della freccia:  
"Deposito" e "Ritiro" includono "Identifica"

# 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



# Analisi dei requisiti

- 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.

# Regole per scrivere il documento dei requisiti

- Definire il glossario (nomi e verbi)
- Unificare i termini sinonimi
- Specificare ciò che è troppo generico
- Introdurre eventuali elementi mancanti
- Specificare lo scenario delle operazioni
- Usare i verbi al futuro

# Glossario

termine	descrizione	dati
Libro o volume	Elemento della biblioteca	ISBN, titolo, autori, casa editrice, anno edizione, voce di classificazione
Catalogo	Lista dei libri catalogati mediante il sistema di classificazione	Libro, classificazione, collocazione
iscritto	Utente della biblioteca	numero tessera, cognome, nome, dati nascita, indirizzo
prestito	Cessione di un libro ad un iscritto con l'impegno di restituirlo entro un certo giorno	id. libro, id. iscritto, data cessione, data restituzione

# Lista delle operazioni (scenari della qualità)

id	operazione	frequenza	agente
1	Inserimento nuovo libro	50/settimana	bibliotecario
2	Iscrizione nuovo utente	5/settimana	bibliotecario
3	Prestito di un libro	200/giorno	bibliotecario
4	Restituzione di un libro	180/giorno	bibliotecario
5	Controllo prestiti scaduti	5000/giorno	bibliotecario
6	Consultazione catalogo	1000/giorno	tutti

# Il documento dei requisiti

- L'applicazione gestirà i prestiti dei volumi agli iscritti.
- L'applicazione dovrà usare il sistema di classificazione per descrivere il contenuto di ciascun volume con i relativi metadati bibliografici (autore, titolo, editore, anno, ISBN) e associare un contatore di copia.
- L'applicazione permetterà l'accesso via web a) ai bibliotecari per aggiornare la base di dati, b) agli utenti per consultare i metadati dei volumi.
- ...

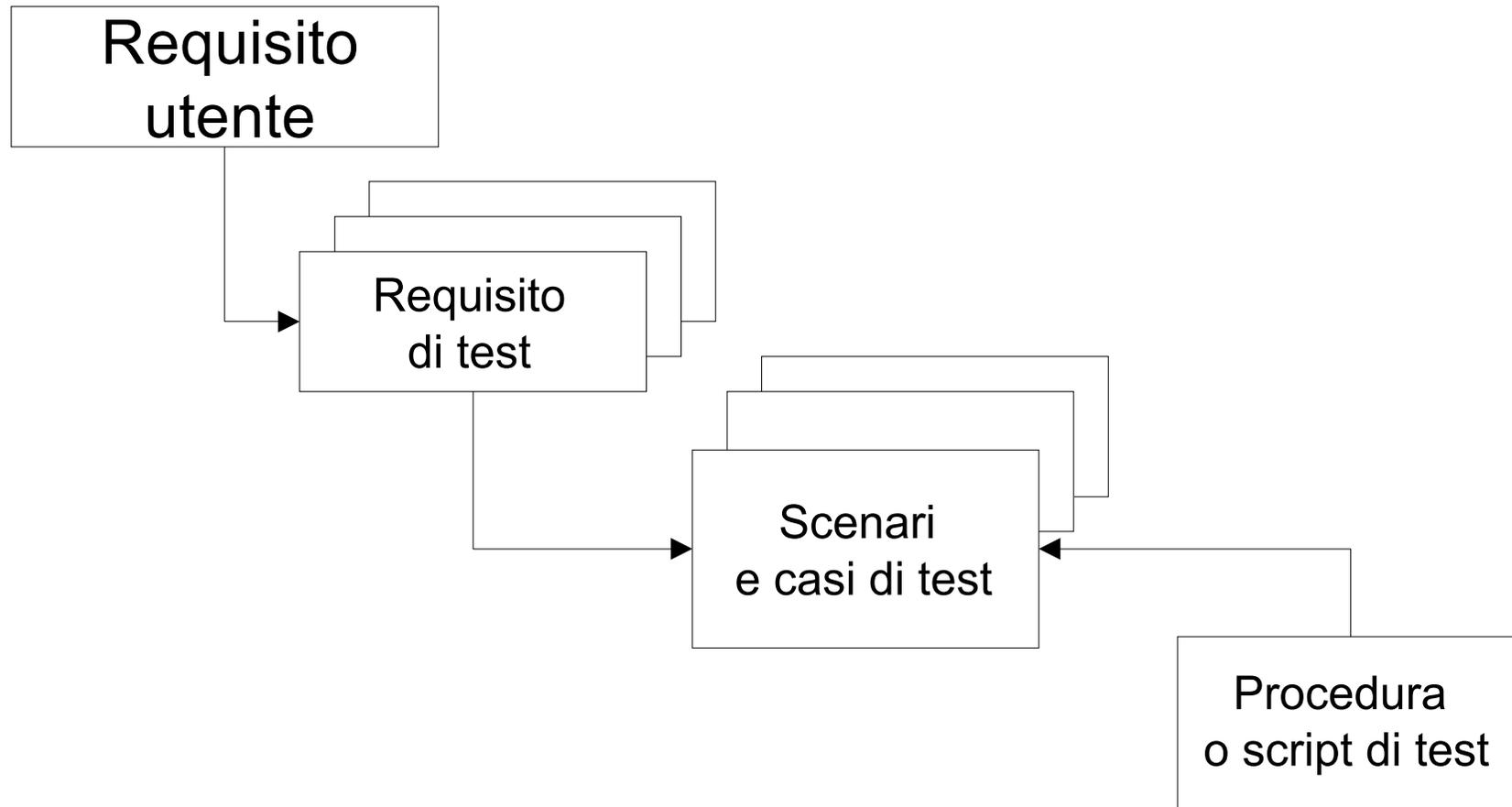
# Requisiti e test

- Ogni requisito deve avere un risultato osservabile
- Ogni requisito dovrà trovare “soddisfazione” nel codice
- Il codice dovrà essere controllato (*testato*) per “verificare” che soddisfi il requisito per cui è stato scritto

# Esempio

- Requisito: l'applicazione gestirà i prestiti dei volumi agli iscritti
- Possibile risultato osservabile: quando qualcuno chiede un libro in prestito bisogna registrare la richiesta e se il volume è disponibile dare il libro e registrare il prestito con la scadenza
- Possibile test:  
`richiesta_prestito("utente_Paolo", "Promessi sposi")`

# Requisiti e test



# Esempio: bancomat

## Requisiti:

1. “Il bancomat permetterà di prelevare denaro”
2. “Si potranno prelevare da 20€ a 300€”
3. “I prelievi saranno in multipli di 20€”

## Esercizio:

- Solo tre requisiti.
- Cosa testiamo?
- Ci sono dei requisiti impliciti che occorre esplicitare?

# Testare i prelievi

## Requisiti sui test:

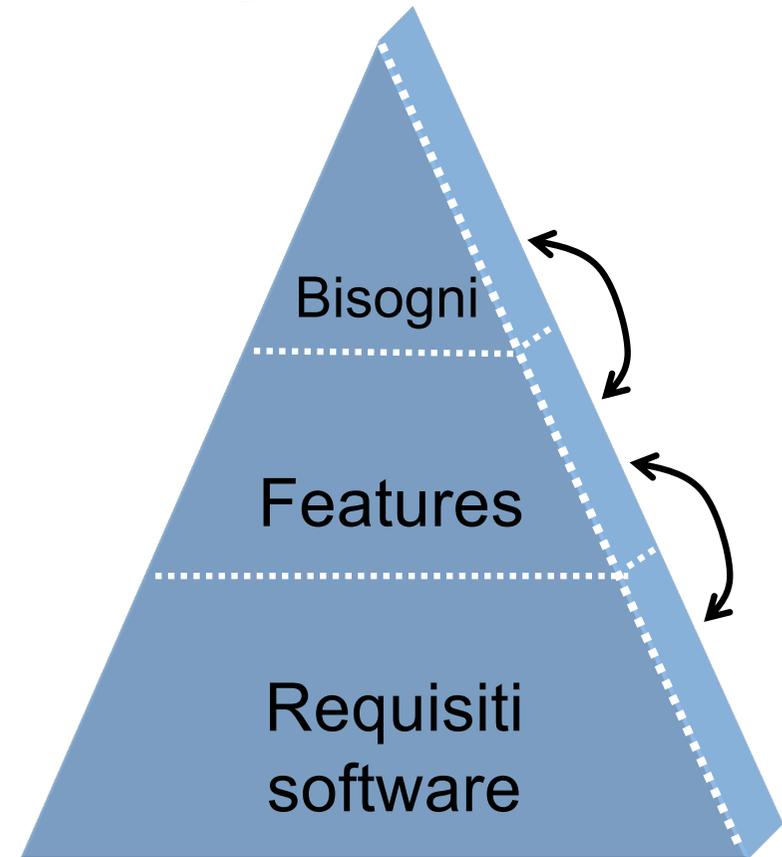
- “Si controllerà che esista l’ opzione di prelievo”
- “Si controllerà di poter fare prelievi di multipli di 20€, tra 20€ e 300€”
- “Si controllerà che sia proibito prelevare <20€”
- ” Si controllerà che sia proibito prelevare >300€”
- ” Si controllerà che sia proibito prelevare multipli di 20€ > 300€ ”
- ” Si controllerà che sia proibito prelevare non-multipli di 20€”
- ” Si controllerà che siano proibite combinazioni strane (es.: tanti zeri, tanti 9, 19.999999)”
- “Si controllerà che il denaro prelevato sia pari a quello richiesto”
- ” Si controllerà che il prelievo sia coperto dal saldo esistente nel conto”

- *Questi sono requisiti sui test e NON test perché non descriviamo i dati da usare (es: 21, 40, 60, 9999)*
- *I dati verranno decisi quando eseguiremo i casi di test*

# La piramide dei requisiti

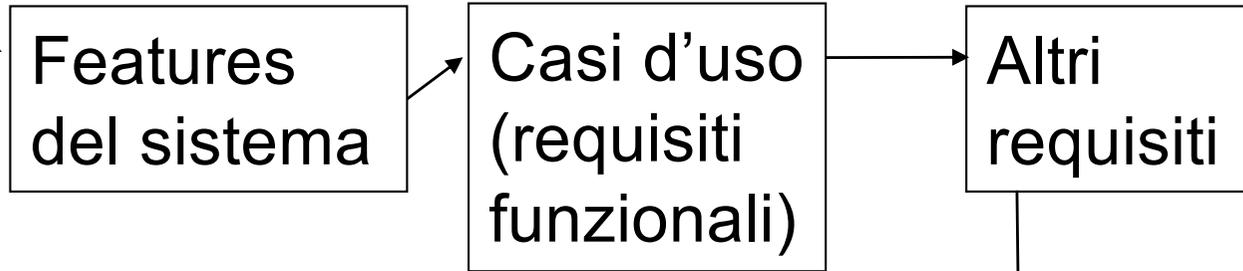
In cima sono i *bisogni* del cliente.  
Esempio: “I dati dei clienti dovranno essere conservati per 10 anni”.

Al bisogni danno risposta le “features”:  
ogni feature corrisponde ad un bisogno  
ed ha proprietà (requisiti funzionali o  
meno) che il sistema finito dovrà  
possedere e che verranno controllate  
mediante test

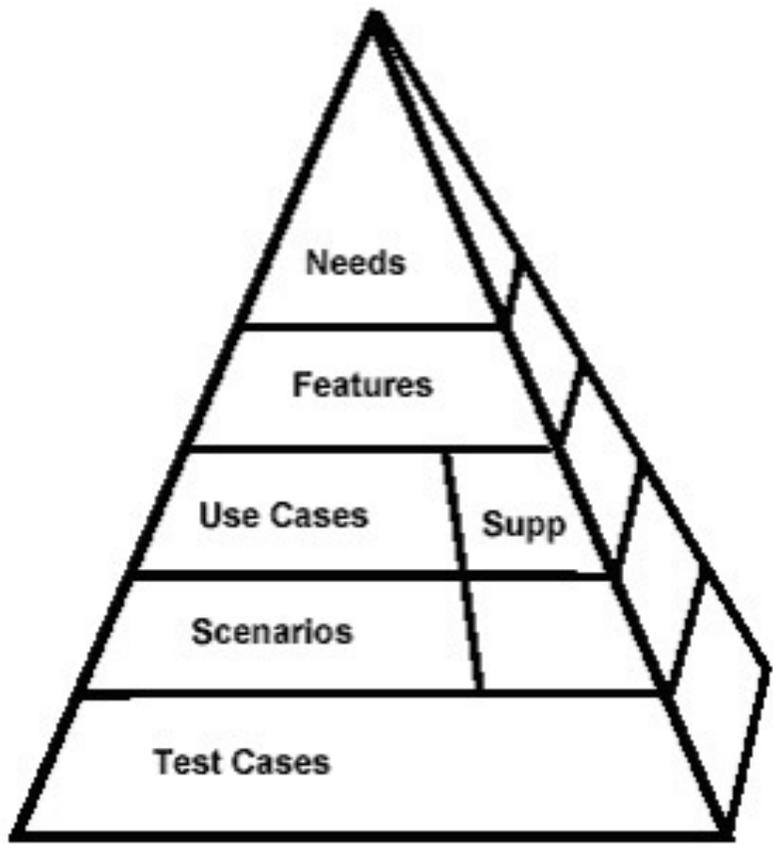


Bisogni  
dell'utente

# Dai bisogni ai requisiti



- Requisiti UI
- Requisiti non funzionali
- Requisiti di sicurezza



# Feature

Una *feature* è un **servizio** che soddisfa uno o più **bisogni del cliente**.

Per esempio: *"Il sistema offre un database relazionale per gestire i dati persistenti"*

La nozione di *feature* è utile specie quando si deve trattare con sistemi sw fatti di componenti riusabili, cioè riconfigurabili o modificabili

Software Features						
	Pro	Suite	Plot	Pipes	SetUps	Dev
Pldot	x	x	x	x	x	x
Envelope	x	x	x	x	x	x
Lines 5/1 5/2 5/3 5/9 6/1 6/5 1-1h/1	x		x		x	x
Lines 5/2ex 5/9ex 6/6 6/7	x		x		x	x
Flow	x		x		x	x
Dot Refresh	x	x	x		x	x
Main Channel Line	x		x		x	x
Trading Types	x		x		x	x
Support/Resistance Nearby & Furtherout	x		x		x	x
Time Frames Any 3 Intraday + 5 Daily & Higher	x	x	x	x	x	x
Higher Time Frame Overlays on Lower Time Frame	x	x	x	x	x	
Walk Forward Testing	x	x	x	x	x	
Full Control of TradeStation Paint Layers			x			
Full Accuracy Daily on Intraday Plotting	x	x	x	x	x	x
Refresh Bands Including Higher Time Period Overlays		x	x		x	
Blues Including Higher Time Period Overlays		x	x		x	
Mother Goose Including Higher Time Period Overlays		x	x		x	
59zones Including Higher Time Period Overlays		x	x		x	
Pipes Including Higher Time Period Overlays			x	x	x	
Quarterly, Yearly and Multi-Year Bar Charts & Overlays		x				
Bar Pct Complete Utility		x				
Alerts - Price Nears Any Combination of DG lines					x	
Alerts - Radar Screen Scanning					x	
Set-Ups - Pre Determined Price & Structure					x	
Complete DG2 Data Base Access			x		x	x
New Concepts Such As:			x		x	x
Rvals - Values for Structure Relationships			x		x	x
WIE - Where is Envelope			x		x	x
AD - Accumulation & Distribution Indication			x		x	x
1/4/2011						
EL Function Full Access to DG2 Database						x

# Feature

IEEE 829 (*Standard for Software and System Test Documentation*):

**Feature:** caratteristica distintiva di un componente software

**Esempio:** in MS Word, sono feature i meccanismi di editing, di gestione degli indici, di impaginazione, ecc.

# Esempi di *feature*

- Feature: *Il prodotto include un ambiente speciale per editare formule matematiche*
- Feature: *Il prodotto può impaginare documenti o con layout predefiniti o definiti da utente*
- Feature: *Il prodotto può importare un documento in formato RTF*

# Requisito (requirement)

- Una frase che definisce il risultato di una funzione o processo, utilizzabile da qualcuno
- Esempio:

**Req. 3.4.B** The database shall support the generation and control of configuration objects; that is, objects which are themselves groupings of other objects in the database. The configuration control facilities shall allow access to the objects in a version group by the use of an incomplete name

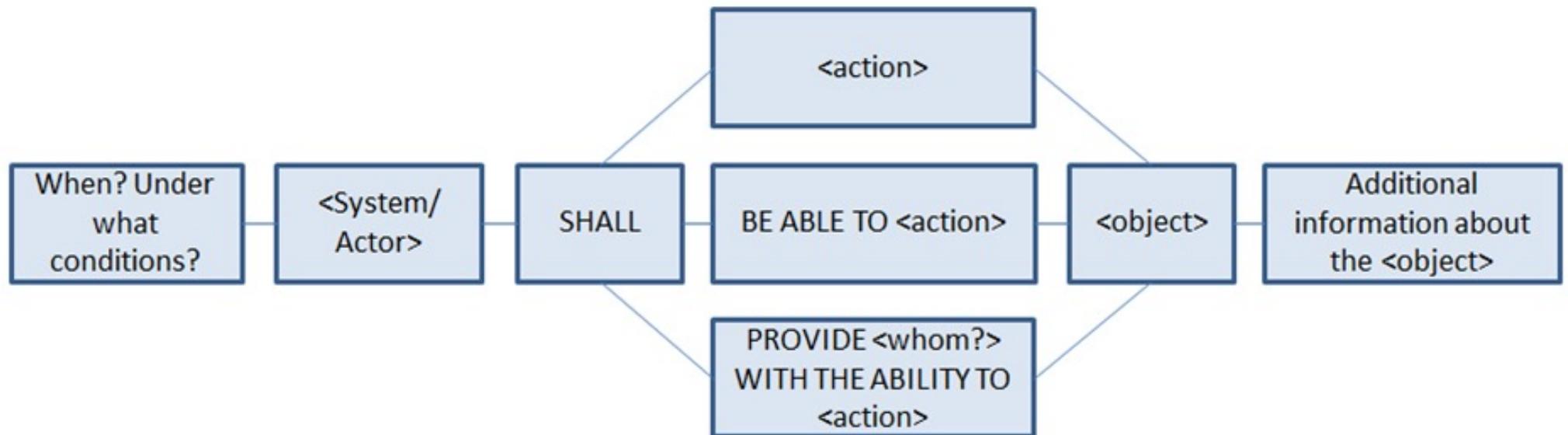
# Una lista di requisiti

## Requirements Document

- REQ001 System shall have fields for firstname, middle initial and last name.
- REQ002 System shall display a name if there is one in the stored profile.
- REQ003 System shall require name is completed.
- REQ004 System shall have a field for position or title.
- REQ005 System shall require title is completed.
- REQ006 System shall display a position or title if there is one in the stored profile.
- REQ007 System shall have a field for email address.
- REQ008 System shall have a field for alternate email address.
- REQ009 System shall display an email address if there is one in the stored profile.
- REQ010 System shall display an alternate email address if there is one in the stored profile.
- REQ011 System shall require email address is completed.
- REQ012 System shall require alternated email address is completed.
- REQ013 System shall have a field for daytime phone number.
- REQ014 System shall display a phone number if there is one in the stored profile.
- REQ015 System shall require phone number is completed.
- REQ016 System shall validate all characters in the phone number field are digits when user exits the field.
- REQ017 System shall display an error message if not all characters in the phone number field were digits.
- REQ018 System shall have a field for a fax number.
- REQ019 System shall require fax is completed.
- REQ020 System shall display a fax number if there is one in the stored profile.
- REQ021 System shall validate all characters in the fax number field are digits when user exits the field.
- REQ022 System shall display an error message if not all characters in the fax number field were digits.
- REQ023 System shall have two fields for a street address.
- REQ024 System shall require the first street address field is completed.
- REQ025 System shall display an address if there is one in the stored profile.
- REQ026 System shall have a field for city.
- REQ027 System shall require the city field is completed.
- REQ028 System shall display a city if there is one in the stored profile.
- REQ029 System shall have a field for state.
- REQ030 System shall display a state if there is one in the stored profile.
- REQ031 System shall require the state field is completed.
- REQ032 System shall have a field for zip code.
- REQ033 System shall display a zip code if there is one in the stored profile.
- REQ034 System shall require the zip code field is completed.

# I requisiti sono frasi strutturate, cioè seguono uno schema prefissato

La International Requirements Engineering Board (IREB) ha definito uno schema – detto *boilerplate* – per dare forma standard ai requisiti:



**Esempio:** Quando la biblioteca è aperta un iscritto potrà consultare il catalogo in cui potrà cercare i volumi disponibili

# Che cos'è un requisito?

- Un requisito può essere una **descrizione** astratta e generica di un servizio o vincolo di sistema, oppure una **specificata** concreta e dettagliata di una funzionalità del sistema
- I requisiti hanno infatti una doppia funzione:
  - Base per un'*offerta di contratto*: devono essere aperti a varie interpretazioni
  - Base per un *contratto*: devono essere dettagliati e non soggetti ad interpretazioni arbitrarie

# A Software Requirement is ... (IEEE 610)

1. A condition or capability **needed** by a user to solve a problem or achieve an objective
2. A condition or capability that must be met or possessed by a system or component to **satisfy** a contract, standard, specification, or other formally imposed documents
3. A **documented** representation of a condition or capability as in (1) or (2)

# Cosa sono i requisiti?

I requisiti sono **proprietà** di un sistema o prodotto **desiderate** dai suoi committenti e **verificabili**

- Quali proprietà?
  - funzionali oppure non-funzionali
- Che succede se alcuni requisiti confliggono?
  - Priorità / preferenze
- Chi sono i committenti e come si registrano i relativi requisiti?
  - Stakeholders che usano viste specializzate
- Come si verifica che un requisito sia soddisfatto?
  - Testing (sviluppatori)
  - Validazione (committenti)

# Esempi di requisiti

- Requisito: *l'editor permetterà di gestire testi matematici*
- Requisito: *l'editor permetterà di impaginare automaticamente la lista delle figure e l'indice analitico*
- Requisito: *L'editor potrà interoperare con un programma di foglio elettronico importando le tabelle*

# Differenza tra feature e requisito

- Un **requisito** descrive una capacità che deve possedere il sistema; è scritto per **testare** l'implementazione in relazione agli scenari di uso del sistema stesso
- Una **feature** è un insieme di requisiti logicamente correlati; di solito descrive una funzionalità a grana grossa di un prodotto
- Esempio:
  - *Feature*: carrello in un sito di e-commerce
  - *Requisiti*:
    - L'utente potrà aggiungere elementi al carrello
    - L'utente potrà rimuovere elementi dal carrello
    - L'utente potrà iniziare il pagamento dal carrello
    - ...

# II documento dei requisiti

SRS:  
software  
requirements  
specification  
document

## Table of Contents

1 Introduction.....	4
1.1 Purpose.....	4
1.2 Document Conventions.....	4
1.3 Project Scope.....	4
1.4 References.....	4
2 System Description.....	4
3 Functional Requirements.....	4
3.1 System Features.....	4
3.1.1 System Feature 1.....	5
3.1.2 System Feature 2.....	5
3.2 Use Cases.....	5
3.2.1 Use Case Diagrams.....	5
3.2.2 Use Case 1.....	5
3.2.3 Use Case 2.....	5
3.3 Entity Relationship Diagrams.....	5
3.4 Data Dictionary.....	6
3.4.1 Entity 1.....	6
3.4.2 Entity 2.....	6
4 External Interface Requirements.....	6
5 Technical Requirements (Non functional).....	6
5.1 Performance.....	6
5.2 Scalability.....	6
5.3 Security.....	6
5.4 Maintainability.....	6
5.5 Usability.....	6
5.6 Multi lingual Support.....	6
5.7 Auditing and Logging.....	6
5.8 Availability.....	6
6 Open Issues.....	7

# I requisiti sono “bifronti”

Se un'organizzazione deve assegnare un **contratto** per lo sviluppo di un sistema software, scrive un documento (“**capitolato**”) per definire i suoi scopi in modo sufficientemente astratto da non predisporre una particolare soluzione e poter esporre un bando

I requisiti devono essere scritti in modo che più possibili contraenti possano rispondere al bando, offrendo diversi modi di realizzare la soluzione coerentemente col capitolato

Dopo che il contratto è stato assegnato, il contraente vincitore deve scrivere per il cliente un “**documento tecnico**” più dettagliato rispetto al capitolato, che possa essere allegato al contratto

Entrambi i documenti (capitolato e documento tecnico) possono essere definiti “*documento dei requisiti*” del sistema da sviluppare

# Specifica e definizione dei requisiti

- **Definizione dei requisiti**
  - Descrizione in linguaggio naturale dei bisogni e dei vincoli operativi di un sistema
  - Si scrive per i clienti
- **Specifica dei requisiti**
  - Documento strutturato che dettaglia i servizi attesi dal sistema
  - Scritto come contratto tra cliente e contraente
  - Si scrive per gli sviluppatori

# Domande utili per scrivere i requisiti

- Cosa deve fare il software?
- Che interfacce ha con i suoi utenti, con l'hw, con altri prodotti sw?
- Che prestazioni deve esibire il software?
- Quali attributi (es. interfaccia accessibile) dovrà avere?
- Quali vincoli dovrà soddisfare?

# Cosa non è un requisito?

- L'architettura del sistema
- Vincoli tecnologici (es. linguaggio di implementazione)
- Il processo di sviluppo
- L'ambiente di sviluppo
- Il sistema operativo di riferimento
- Aspetti di riusabilità e portabilità
- **Nota:** le descrizioni del dominio del mondo reale fanno parte dei requisiti (anche se il cliente non le richiede)

# Tipi di requisito

- **Requisiti funzionali:** Descrivono le interazioni tra il sistema ed il suo ambiente, indipendentemente dall'implementazione
  - La piattaforma e-learning tiene traccia delle attività dello studente
- **Requisiti non funzionali:** Proprietà del sistema misurabili dall'utente e non direttamente correlate al suo comportamento funzionale
  - La piattaforma deve gestire almeno 100 utenti contemporaneamente
  - Se un utente fa una domanda per email deve avere risposta entro 48h
  - La piattaforma deve essere disponibile agli studenti 24/7
- **Vincoli** (“Pseudo requisiti”): Imposti dal cliente o dall'ambiente operativo in cui funzionerà il sistema
  - Il linguaggio di programmazione dev'essere Java
  - La piattaforma deve interfacciarsi con documenti scritti con Word su Windows XP

# Requisiti funzionali

- Un requisito funzionale definisce una funzione di un sistema, inclusi il suo ingresso e la sua uscita
- I requisiti funzionali determinano lo sviluppo di codice
- Il testing del codice rispetto ai requisiti funzionali costituisce la **verifica**

# Esempi di requisiti funzionali

- *L'utente dovrà essere in grado di cercare o in tutti i database o in un loro sottoinsieme*
- *Il sistema fornirà i visualizzatori appropriati in modo che l'utente possa leggere i documenti archiviati*
- *Ad ogni fattura verrà associato un identificatore unico (ORDER\_ID) che l'utente potrà copiare nell'area di memoria permanente del suo conto*

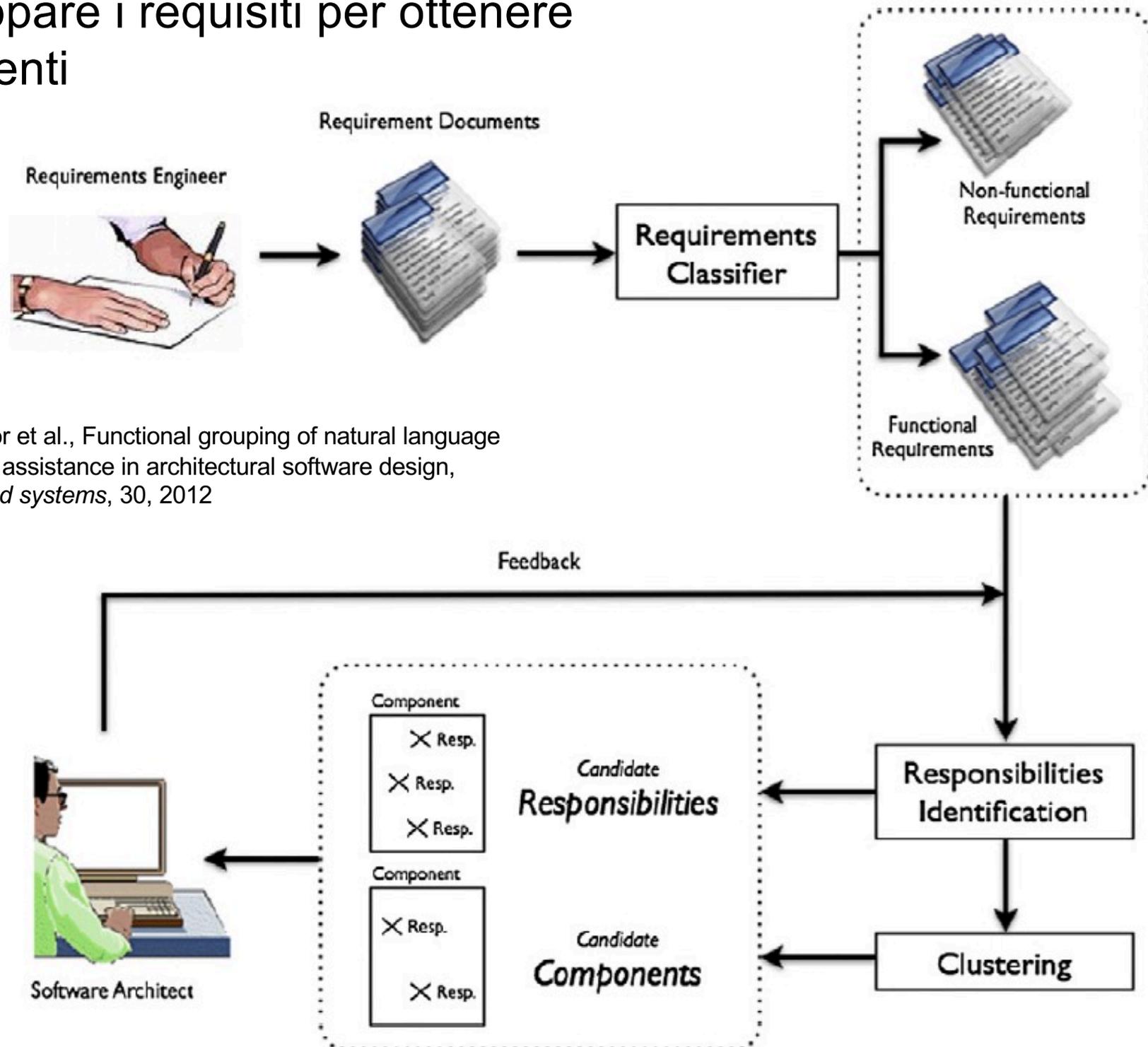
# Requisiti non funzionali

- I requisiti non funzionali sono di solito più critici di quelli funzionali: se non vengono soddisfatti il sistema non verrà utilizzato
- I requisiti non funzionali definiscono proprietà del sistema o vincoli sul suo sviluppo
  - Esempi di proprietà prestazionali: tempo di esecuzione, tempo di risposta, dimensioni della memoria necessaria, throughput.
  - Esempi di proprietà di “*dependability*”: affidabilità, robustezza, sicurezza, integrità, manutenibilità
  - Esempi di vincoli : funzionalità di dispositivi I/O, tipo di processore, ecc.
- Si possono anche specificare vincoli di processo, cioè sul metodo di sviluppo, sul linguaggio di programmazione, sugli strumenti da usare

# Esempi di requisiti non funzionali

- Requisito di prodotto
  - Quando l'utente ha scelto il tipo di biglietto la transazione dev'essere completata entro due secondi
  - Per ogni messaggio dovrà essere possibile usare l'insieme di caratteri standard ASCII
- Requisito organizzativo
  - Il processo di sviluppo ed i documenti consegnati saranno conformi al processo ed alle strutture di documento descritte nello standard ISO-XYZ-2007
- Requisito esterno
  - Il sistema non permetterà ai suoi operatori di conoscere alcuna informazione personale sui clienti, eccetto il nome ed il numero di riferimento interno

# Raggruppare i requisiti per ottenere componenti



fonte: Casamayor et al., Functional grouping of natural language requirements for assistance in architectural software design, *Knowledge-based systems*, 30, 2012

# Sbagliare i requisiti è costoso

*Trovare gli errori durante la fase dei requisiti può far risparmiare fino a 200 volte rispetto allo stesso errore trovato durante la manutenzione*



Barry Boehm- '76, 88

**Oltre metà di tutti i difetti di un sistema sw possono essere attribuiti ad errori fatti durante la stesura dei requisiti**

# Attività di elicitazione dei requisiti

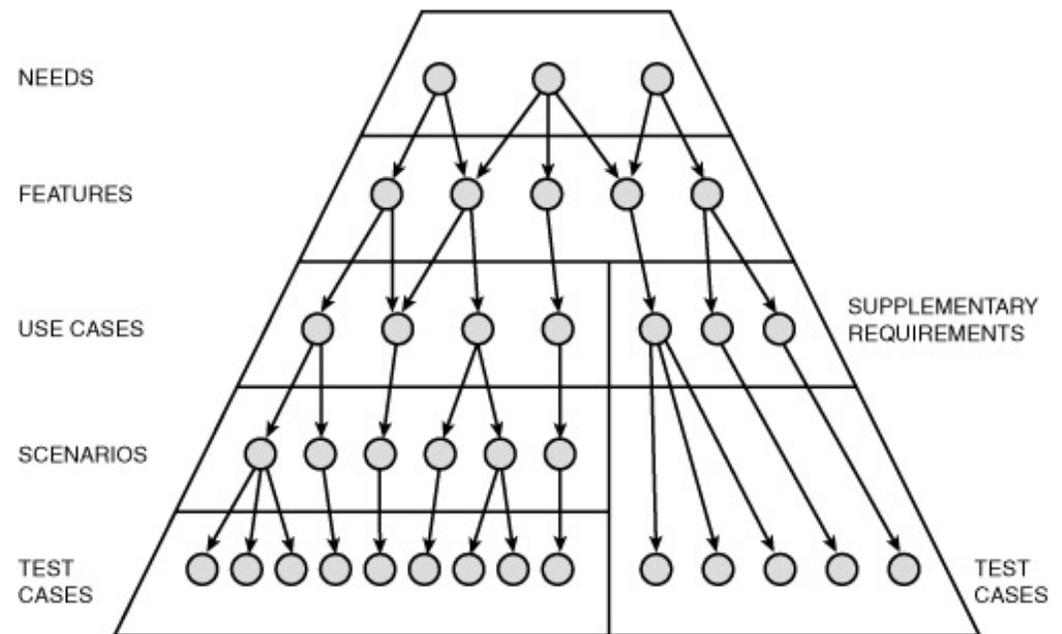
- Identificare gli attori
- Identificare i casi d'uso
- Identificare gli scenari
- Identificare le relazioni tra i casi d'uso
- Raffinare i casi d'uso
- Identificare i requisiti non funzionali
- Identificare gli oggetti partecipanti

# Attività di analisi dei requisiti

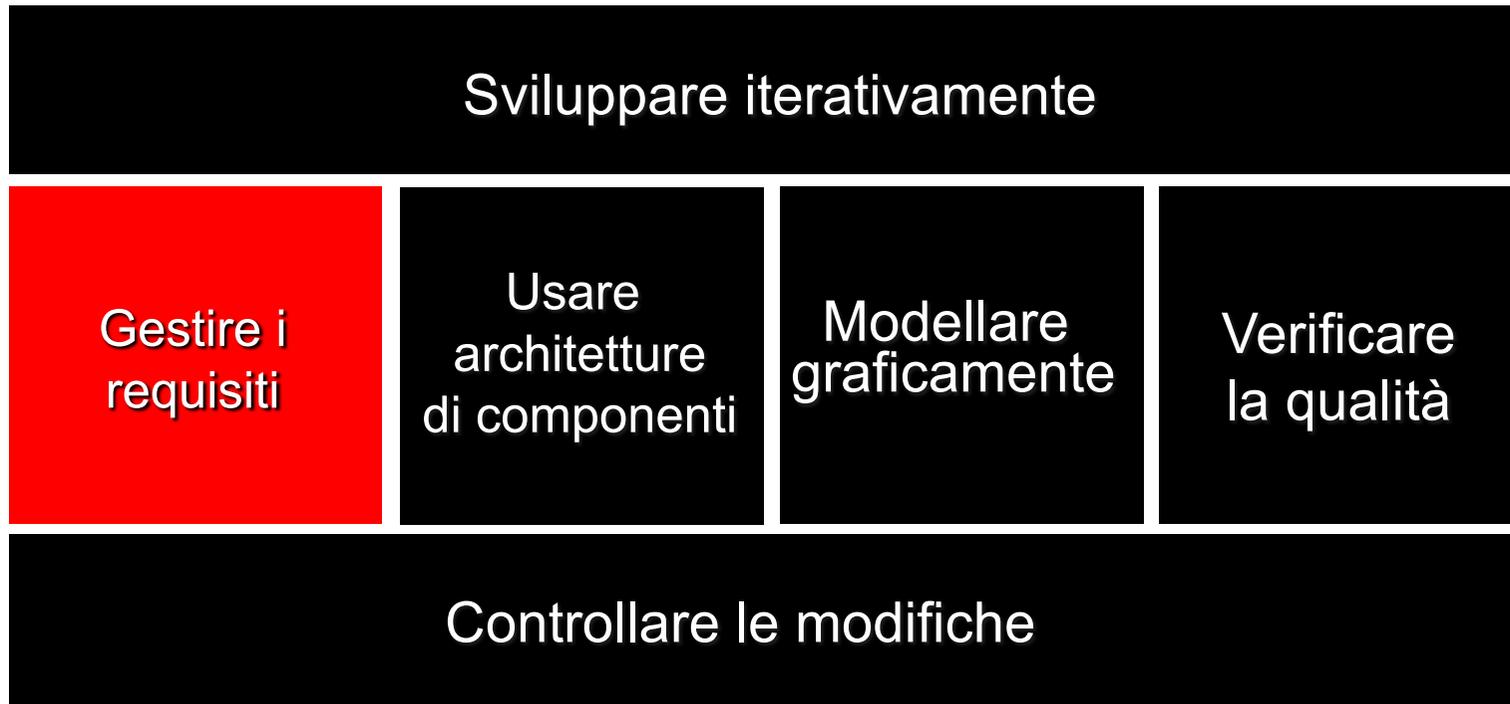
- Classificare
- Chiarire
- Normalizzare
- Ordinare
- Tracciare

# Dai bisogni degli utenti al test

- Un requisito è una funzionalità che il sistema finito dovrà esibire e che verrà verificata (con test)
- Una *feature* è un servizio fornito dal sistema per soddisfare uno o più bisogni (requisiti) del cliente
- Un caso d'uso descrive una sequenza di azioni, eseguite da un sistema, per produrre un risultato che ha valore per un utente
- I casi di test vengono eseguiti su scenari di casi d'uso



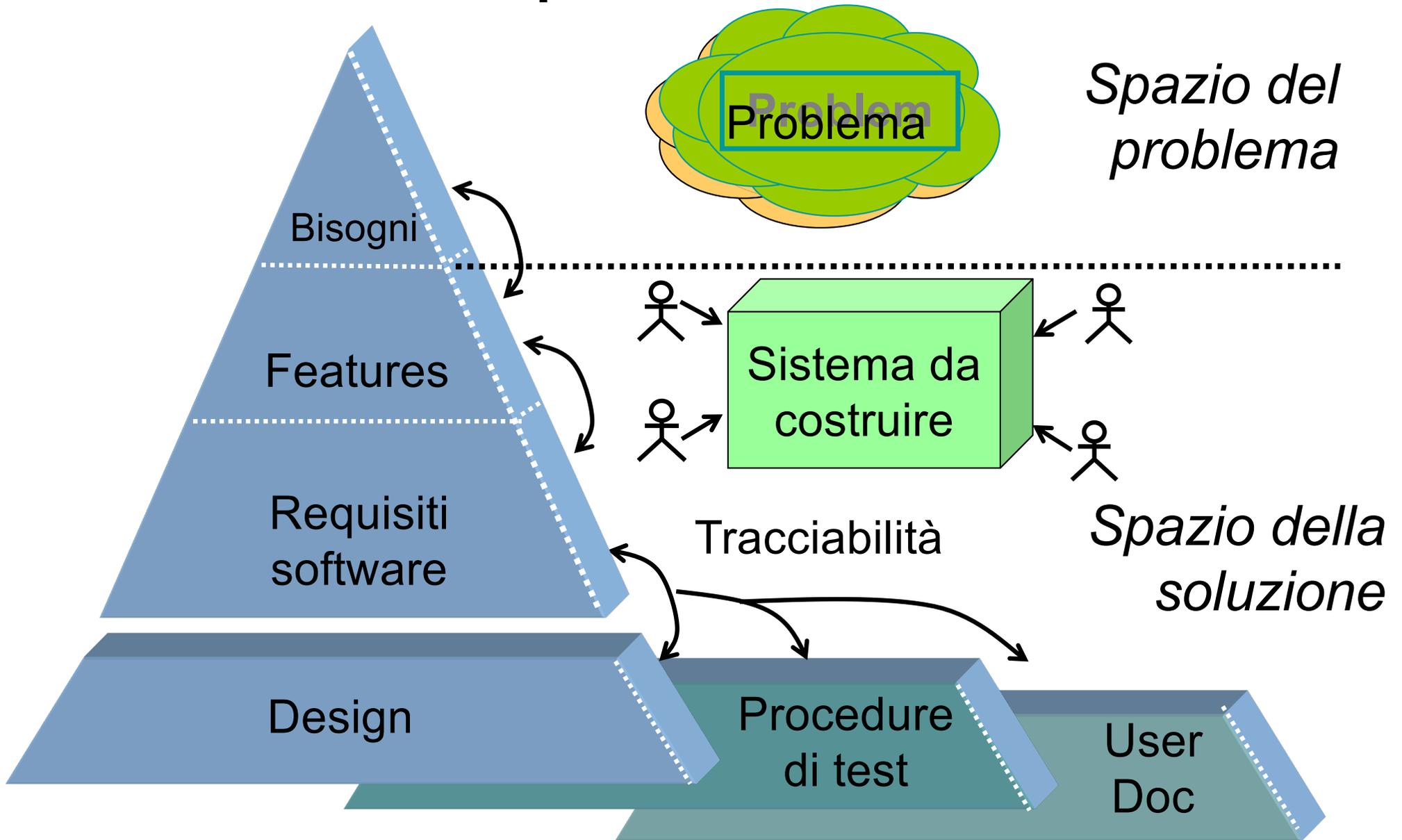
# Principi guida dello sviluppo software



# Gestione dei requisiti (requirements management)

- Gli **strumenti di gestione dei requisiti** permettono di creare un database dei requisiti
- Le operazioni sui requisiti sono: la stesura del requisito, il suo tracciamento, il controllo delle versioni, il testing
- Esistono molti strumenti, anche open source  
(es. rmtoo, vedi [rmblog.accompa.com/2012/04/free-open-source-requirements-management-tool/](http://rmblog.accompa.com/2012/04/free-open-source-requirements-management-tool/))

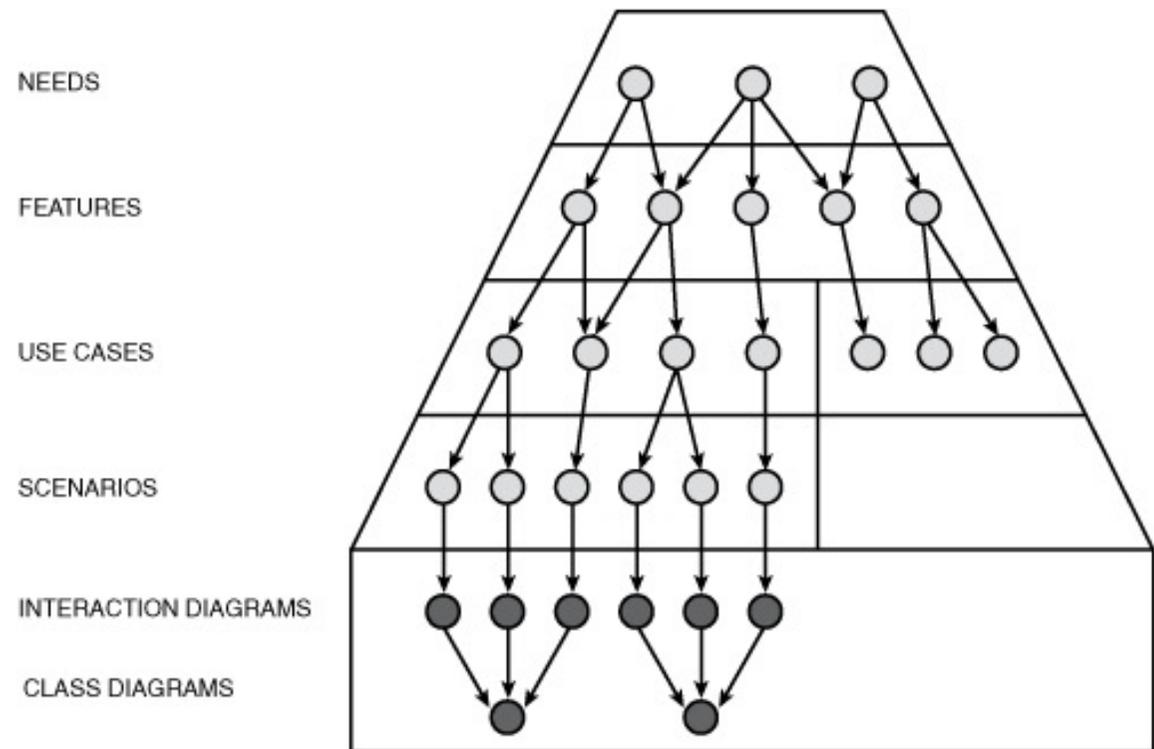
# Gestire i requisiti



# Tracciabilità dei requisiti funzionali

La tracciabilità serve a:

- Verificare che il sistema soddisfi **tutti** i requisiti del cliente
- Verificare che il sistema soddisfi **solo** i requisiti del cliente
- Gestire le modifiche dei requisiti in corso d'opera – quando cambia un requisito, e dunque il codice, dovremo rifare i test ad esso relativi



# Tracciabilità

## Requisito

```
1.1 The system shall
blah, blah...
1.2 If the co-worker
is blah, blah, the
system shall inform
the user ...
1.2.1...
```

Input:  
CoWorker record in  
which blah = 1, and  
...  
Expected output:  
blah = 2

## Caso di test

....

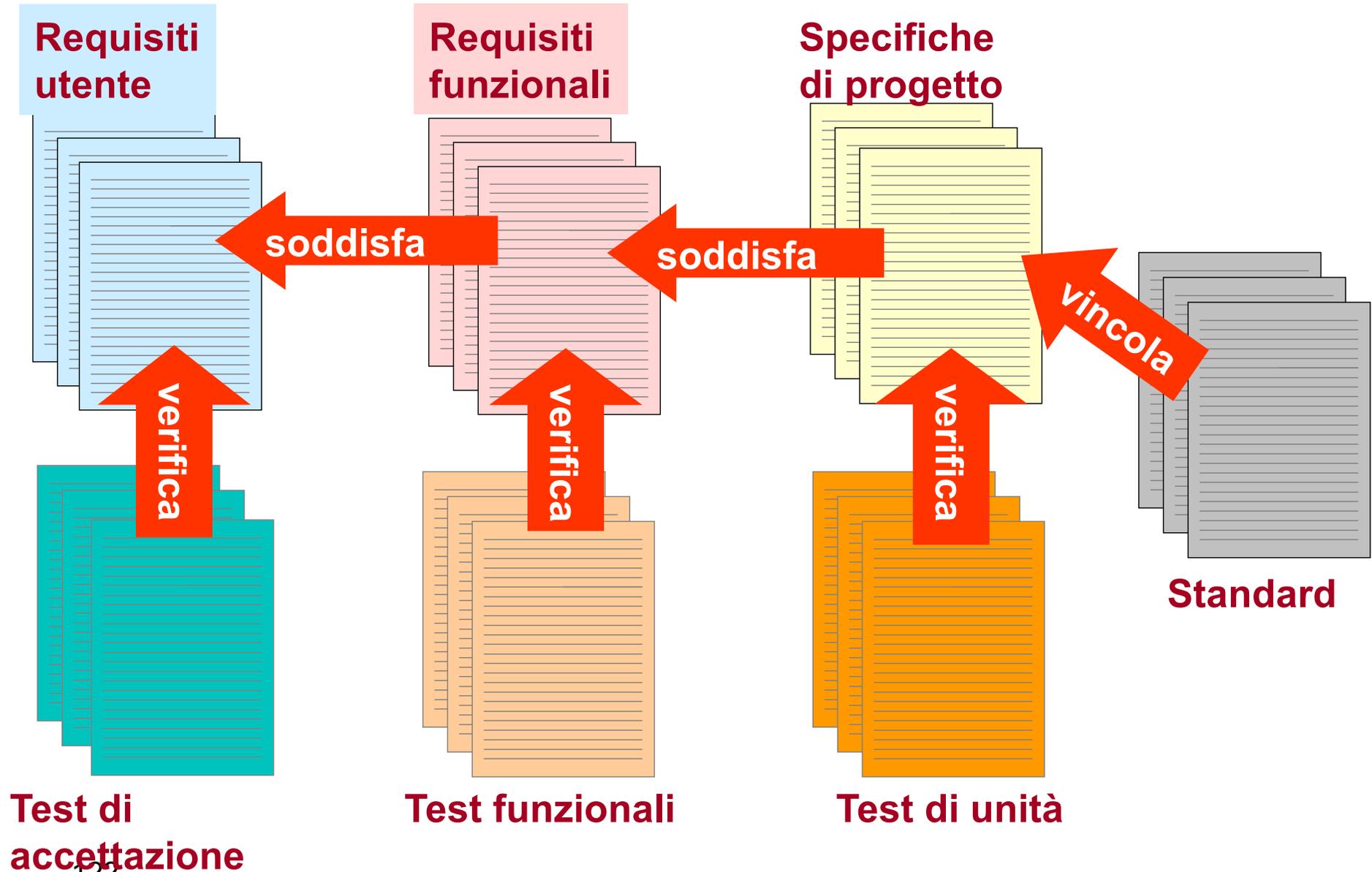
```
if !(fizzwick(cw)) {
  for (i=0;i=max;i++)
    update(cw, i);
  ...
else
  ...
```

## Codice

# Tracciamento dei requisiti

- La tracciabilità dei requisiti è la capacità di descrivere e seguire il ciclo di vita di un requisito funzionale dall'origine al suo uso e viceversa
- La tracciabilità documenta le relazioni tra gli artefatti creati durante lo sviluppo, quali i requisiti, le specifiche, i modelli, i test ed i componenti sviluppati
- La relazione principale che riguarda i requisiti nell'ambito del processo di sviluppo è il “*soddisfacimento*”: come viene soddisfatto un requisito dagli artefatti di sviluppo?
- Altra relazione è “*verifica*”: come si verifica con un test che un requisito è stato soddisfatto?

# La tracciabilità



# Matrice di tracciamento dei requisiti

Requirement Identifiers	Reqs Tested	REQ1 UC 1.1	REQ1 UC 1.2	REQ1 UC 1.3	REQ1 UC 2.1	REQ1 UC 2.2	REQ1 UC 2.3.1	REQ1 UC 2.3.2	REQ1 UC 2.3.3	REQ1 UC 2.4	REQ1 UC 3.1	REQ1 UC 3.2	REQ1 TECH 1.1	REQ1 TECH 1.2	REQ1 TECH 1.3
<b>Test Cases</b>	321	3	2	3	1	1	1	1	1	1	2	3	1	1	1
<b>Tested Implicitly</b>	77														
<b>1.1.1</b>	1	x													
<b>1.1.2</b>	2		x	x											
<b>1.1.3</b>	2	x											x		
<b>1.1.4</b>	1			x											
<b>1.1.5</b>	2	x												x	
<b>1.1.6</b>	1		x												
<b>1.1.7</b>	1			x											
<b>1.2.1</b>	2				x		x								
<b>1.2.2</b>	2					x		x							
<b>1.2.3</b>	2								x	x					
<b>1.3.1</b>	1										x				
<b>1.3.2</b>	1										x				
<b>1.3.3</b>	1											x			
<b>1.3.4</b>	1											x			
<b>1.3.5</b>	1											x			
<b>etc....</b>															
<b>5.6.2</b>	1														x

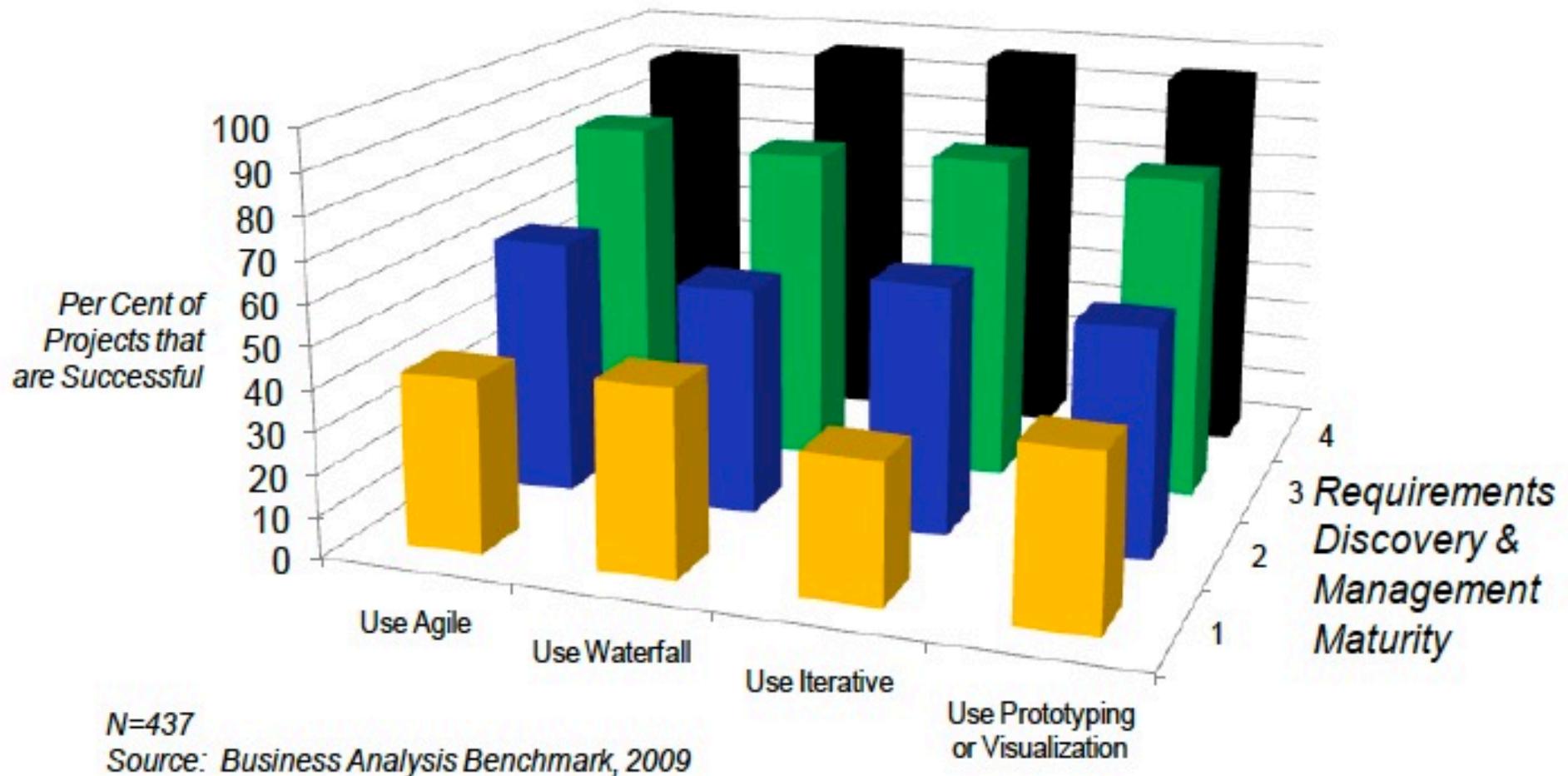
Fonte: Wikipedia

# Use Cases in DOORS

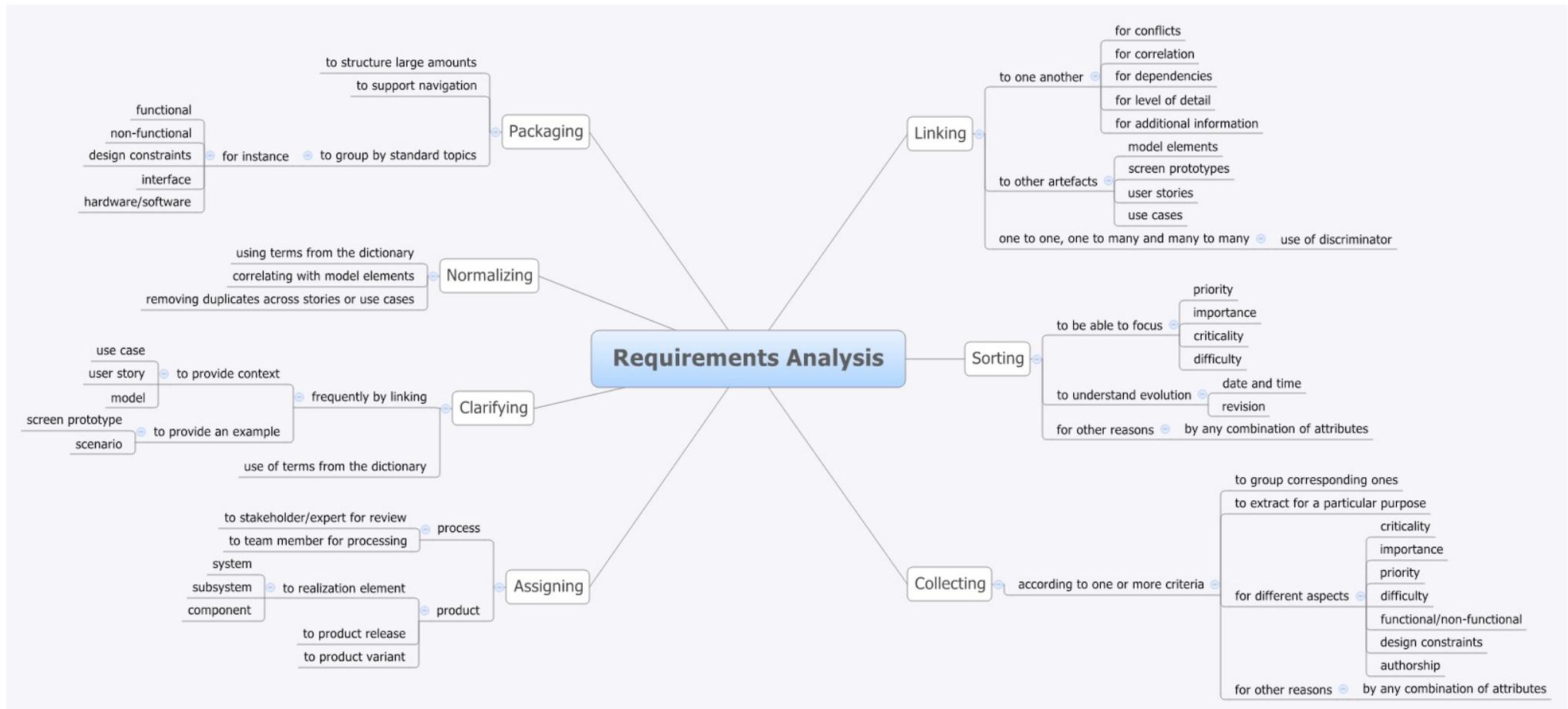
ID	Object Type	Feature Number	Step	Actor In Cont	Receiving Act	Packet/IN SRD for PINVPN1.0 Features
eSAE.R2 4.1196	Use Case	61746				<b>4.3 Use Cases</b>
eSAE.R2 4.1245						<b>4.3.1 Create A String Attribute Use Case</b>
eSAE.R2 4.1249		61746				<b>4.3.1.1 Use Case Level</b>
eSAE.R2 4.1250		61746				User creates a 'String' attribute of CHAR/VARCHAR/DB_Type in an existing Service Table
eSAE.R2 4.1231		61746				<b>4.3.1.2 Short Description</b>
eSAE.R2 4.1232		61746				The DMT shows in the GUI, the 'String' data type added in one of the Service Tables as CHAR (or VARCHAR)
eSAE.R2 4.1199		61746				<b>4.3.1.3 Primary Actor</b>
eSAE.R2 4.1200		61746				<b>Service Developer</b>
eSAE.R2 4.1201		61746				<b>4.3.1.4 Identifier</b>
eSAE.R2 4.1202		61746				uc-61746-01
eSAE.R2 4.1203		61746				<b>4.3.1.5 Pre-conditions</b>
eSAE.R2 4.1204		61746				A data model for a Service exists with some tables and attributes. Steps highlighted in "YELLOW" are new functionality.
eSAE.R2 4.1208		61746				<b>4.3.1.6 Main Scenario</b>
eSAE.R2 4.1209		61746	1	Service Developer	DMT	Service Developer opens an existing service data model from within the DMT
eSAE.R2 4.1272		61746	2	DMT	Service Developer	DMT brings up the service data model in the GUI
eSAE.R2		61746	3	Service	DMT	Service Developer selects an Attribute object in the Design Pane and selects its properties

Username: Darshak Kothari      Exclusive edit mode

# Importanza del processo di gestione dei requisiti



# Mappa concettuale: l'analisi dei requisiti



da: Open Requirement Management Framework

# Sommario

- Le feature di un prodotto sono di solito poche decine, i requisiti possono essere centinaia o anche migliaia
- Le feature danno una descrizione ad alto livello delle responsabilità e dei servizi che offre un sistema per rispondere ai bisogni degli stakeholder
- I requisiti servono per definire le feature; la distinzione più importante è tra funzionali e non
- I casi d'uso sono una tecnica per analizzare e classificare i requisiti funzionali
- Le user stories sono la tecnica agile per descrivere i requisiti

# Feature

- Quando qualcuno costruisce un **prodotto** deve definire le *feature* del prodotto
- Le *feature* saranno la base per confrontare il prodotto con altri prodotti simili o per decidere come il prodotto va modificato per soddisfare nuovi requisiti

# Requisiti

- Quando qualcuno chiede a qualcun altro di creare un **sistema** software occorre stabilire i *requisiti* di ciò che verrà costruito
- I requisiti saranno la base del testing e di ogni verifica

# Requisiti vs User Story

- I requisiti descrivono il lavoro degli sviluppatori, dal loro punto di vista
- Le User Story descrivono i desideri del cliente, ovvero ciò di cui ha bisogno, dal suo punto di vista

# Domande di autotest

- Cos'è una *user story*?
- Cos'è un *caso d'uso*?
- Cos'è uno *scenario*?
- Qual è la differenza tra “*extend*” e “*include*”?
- Cos'è una *feature* di un prodotto sw?
- Che cos'è un requisito di sistema?
- Quale struttura ha il documento della Specifica dei Requisiti Software (SRS)?
- Cos'è un requisito *non funzionale*?
- Perché è importante “*tracciare*” i requisiti?

# Letture raccomandate

Zielczynski, *Traceability from Use Cases to Test Cases*, 2006

[www.ibm.com/developerworks/rational/library/04/r-3217/](http://www.ibm.com/developerworks/rational/library/04/r-3217/)

K. Wiegers, *Automating Requirements Management*, 1999

[www.processimpact.com/articles/rm\\_tools.html](http://www.processimpact.com/articles/rm_tools.html)

# Riferimenti

- Capitolo 2 del SWEBOK. “Requirements Engineering”
- IEEE830-1998 *Recommended Practice for Sw Requirements Specifications*
- Beatty & Chen, *Visual Modeling for Software Requirements*, Microsoft Press, 2012
- Berenback et al, *Software & Systems Requirement Engineering in Practice*, McGrawHill, 2009
- Cockburn, *Writing Effective Use Cases*, Addison-Wesley, 2001
- Cohn, *User stories applied*, Addison-Wesley 2004
- Leffingwell and Widrig, *Managing Software Requirements: A Use Case Approach*, Addison-Wesley, 2003
- Wynne, *The Cucumber Book*, 2012

[docs.ludost.net/PragmaticProgramming/The%20Cucumber%20Book%20Behaviour-Driven%20Development%20for%20Testers%20and%20Developers.pdf](https://docs.ludost.net/PragmaticProgramming/The%20Cucumber%20Book%20Behaviour-Driven%20Development%20for%20Testers%20and%20Developers.pdf)

# Siti utili

- [github.com/cucumber/cucumber/wiki/Gherkin](https://github.com/cucumber/cucumber/wiki/Gherkin)
- [Cucumber cukes.info/](http://Cucumber.cukes.info/)
- [www.telelogic.com/corp/products/doors/](http://www.telelogic.com/corp/products/doors/)
- [sourceforge.net/projects/eclipsesrs/](https://sourceforge.net/projects/eclipsesrs/)
- [sourceforge.net/projects/osrmt](https://sourceforge.net/projects/osrmt)
- [reqtracer.sourceforge.net](http://reqtracer.sourceforge.net)
- [www.flonatel.de/projekte/rmtoo/](http://www.flonatel.de/projekte/rmtoo/)
- [www.volere.co.uk/tools.htm](http://www.volere.co.uk/tools.htm)
- [www.volere.co.uk/template.htm](http://www.volere.co.uk/template.htm)
- [www.analysttool.com/](http://www.analysttool.com/)
- [www.visual-paradigm.com/VPGallery/usecase/index.html](http://www.visual-paradigm.com/VPGallery/usecase/index.html)
- [www.workspace.com/workspace/Requirements-Management-Software.html](http://www.workspace.com/workspace/Requirements-Management-Software.html)

## Publicazioni di ricerca sull'Ingegneria dei requisiti

- Int. Conf. on Requirements Engineering
- Requirements Engineering Journal

# Domande?

