Ingegneria del Software

Requisiti

Prof. Paolo Giorgini

A.A. 2024/2025

Scopi e funzioni

- I bisogni di un cliente possono essere espressi come **scopi** (obiettivi di un business)
  - Esempio: voglio controllare gli accessi alle strutture UNITN, mi serve poter autorizzare gli studenti e controllare i loro accessi
- 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**

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 bro
- I libri sor sia da un codice n
- Il bibliote to, 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.

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

Michael Jackson (not the singer)

A + S -> R

A: assunzioni sull'ambiente

S: specifica del sistema

R: requisiti

IEEE 830-1993: A **requirement** is defined as:

A **condition** or **capability** needed by a user to solve a problem or achieve an objective

A condition or a capability that must be met or possessed by a system … to satisfy a contract, standard, specification, or other formally imposed document …

Esempio Requisiti

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

Questi sono requisiti: descrivono un programma che calcola Fibonacci(n)

Definizione e Specifica 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 (specifiche tecniche e funzionali caratterizzanti il sistema)
  - Scritto come contratto tra cliente e contraente
  - Si scrive per gli sviluppatori

Elicitazione dei Requisiti

- Processo di acquisizione di informazioni sul sistema da sviluppare
  - Stabilire **cosa** richiede il cliente ad un sistema software (scopo) **senza** definire **come** il sistema verrà costruito (funzioni)
  - può coinvolgere diverse persone (end-users, managers, programmatori, esperti dominio)
  - può anche comportare l'analisi della legislazione e/o di realizzazioni pre-esistenti
- Le diverse persone coinvolte in tale processo rappresentano gli ***stakeholders***

Problemi nell'analisi dei requisiti

- Stakeholders
  - difficilmente hanno una **chiara idea** di quello che esattamente vogliono (soprattutto all'inizio)
  - usano un loro **linguaggio** (molto spesso tipico del dominio) molte volte non noto all'analista
  - diversi stakeholders hanno **richieste diverse**, e quindi pongano requisiti in **conflitto**
- I requisiti del sistema possono dipendere da fattori esterni al sistema stesso
  - Es. esigenze derivate da motivi organizzativi aziendali, o politico/legislativi
- I requisiti (così come gli stakeholder) **possono cambiare** durante la fase di analisi

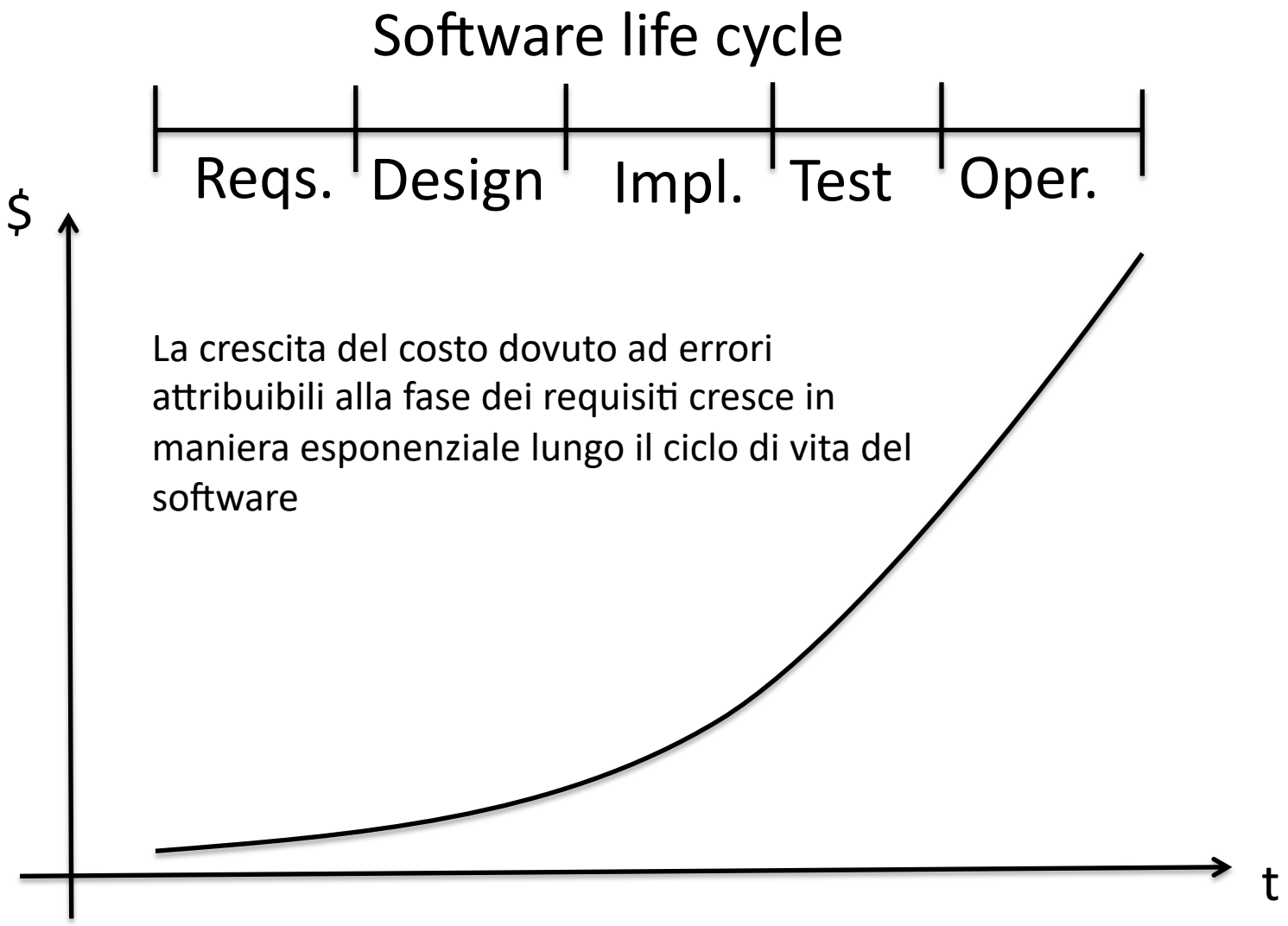
The Challenges of Getting Requirements Right

How the customer explained it

Conseguenze di errori nei requisiti

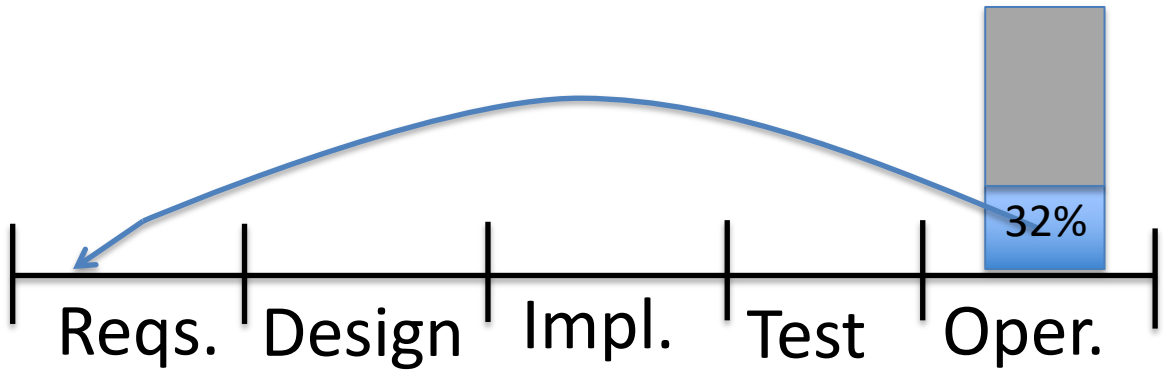
- Errori nei requisiti software possono avere conseguenze importanti
  - In media il 45% dei progetti software superano il budget pianificato [Oxford University, 2013]
  - 43% delle feature rilasciate non sono mai utilizzate
  - $70 vs $14K (fase req. vs fase di produzione)

Costo degli errori

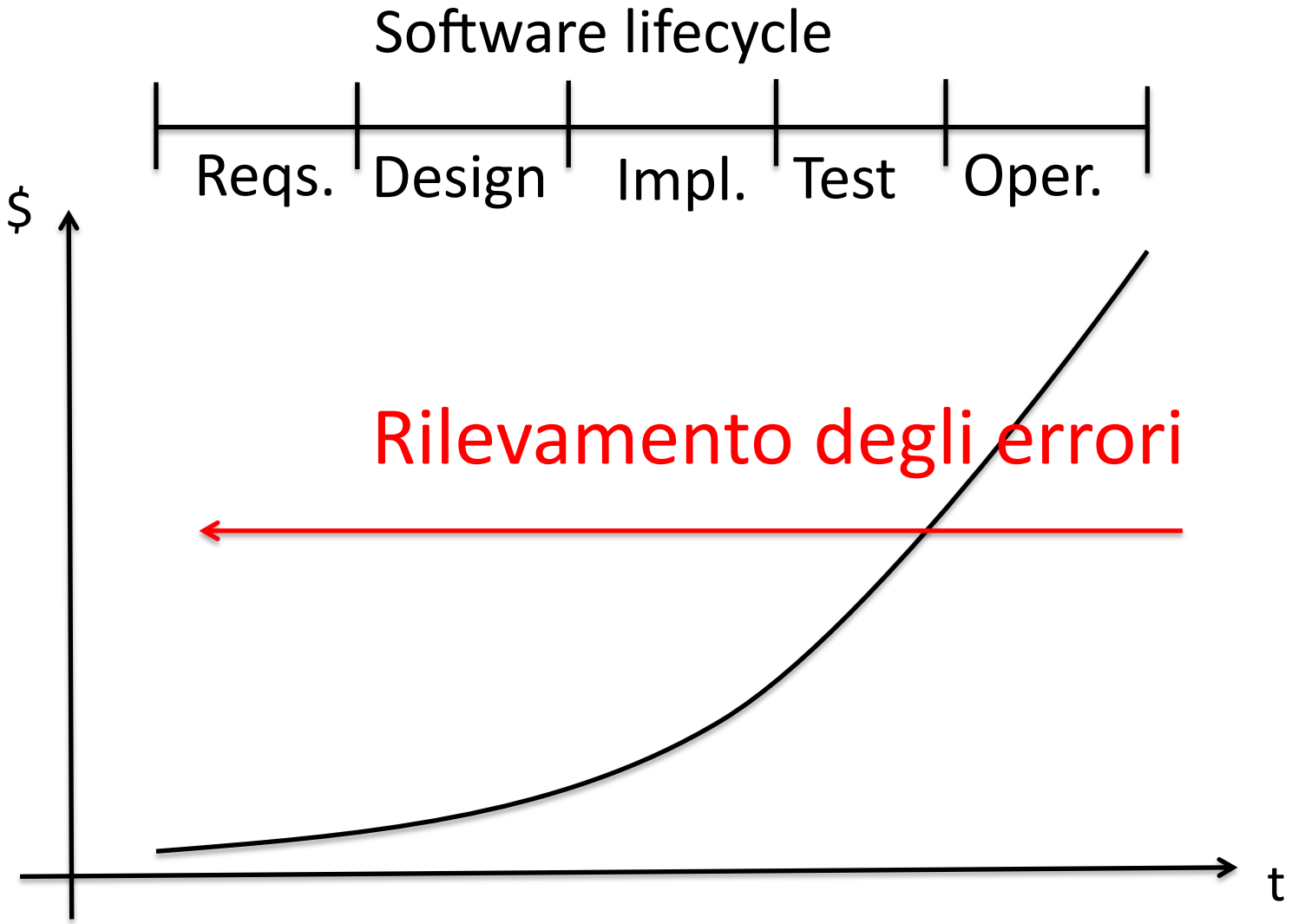


Crescita esponenziale

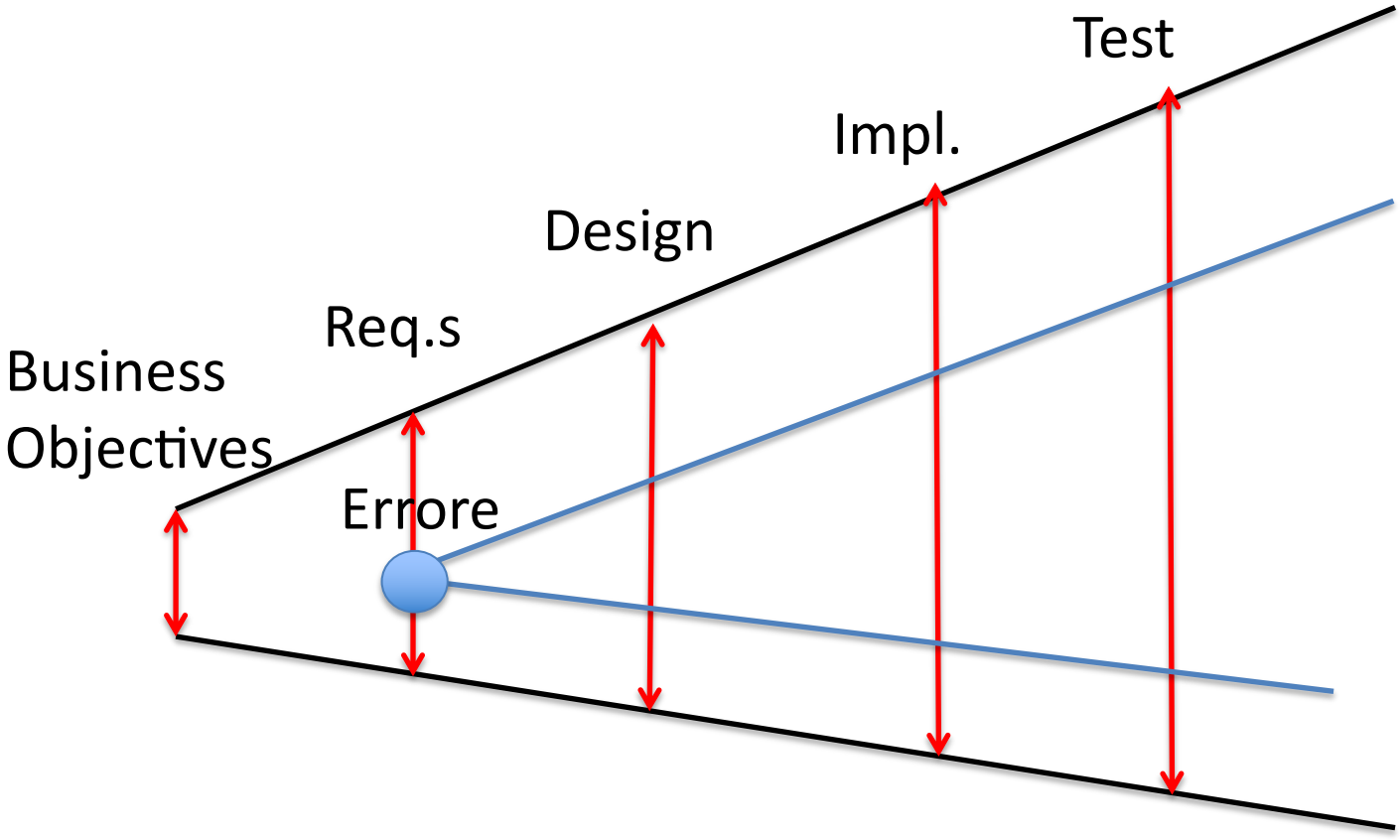
– 32% dei difetti di un software sono dovuti ad errori nei requisiti, ma rilevati soltanto quando il software è già operativo



Costo degli errori



Volume degli artefatti



Requisiti

DESCRIVERE UN REQUISITO

- 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 E' UN REQUISITO

- L'architettura del sistema
- Vincoli tecnologici (es. linguaggio di implementazione)
- Il processo di sviluppo
- L'ambiente di sviluppo
- Aspetti di riusabilità

Categorie di requisiti

- **Requisiti funzionali**
  - descrivono le funzionalità del sistema software, in termini di *servizi* che il sistema software deve fornire, di come il sistema software *reagisce* a specifici tipi di input e di come si *comporta* in situazioni particolari

Es.1 Il sistema software deve fornire un appropriato visualizzatore per i documenti memorizzati

Es.2 L'utente deve essere in grado di effettuare ricerche sia sull'intero insieme di basi di dati che su un loro sottoinsieme

Es.3 Ad ogni nuovo ordine deve essere associato un identificatore unico (Order_ID)

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**

Completezza e Coerenza

- Le specifiche dei requisiti funzionali di un sistema devono essere completi e coerenti
  - **Completi**: tutti i servizi richiesti dagli utenti devono essere definiti
  - **Coerente**: i requisiti non devono avere definizioni contradditorie
- Difficoltà a raggiungere la completezza e la coerenza per sistemi grandi e complessi
  - Si possono commettere errori e omissioni
  - Stakeholder con necessità differenti e spesso incoerenti

Categorie di requisiti (2)

- **Requisiti non funzionali**
  - descrivono *proprietà* del sistema software in relazione a determinati servizi o funzioni e possono anche essere relativi al *processo*
  - non riguardano le specifiche funzioni del sistema ma piuttosto proprietà del sistema come:
    - caratteristiche di efficienza, affidabilità, security, safety, ecc.
    - caratteristiche del *processo di sviluppo* (standard di processo, uso di ambienti CASE, linguaggi di programmazione, metodi di sviluppo, ecc.)
    - Caratteristiche esterne (interoperabilità con sistemi di altre organizzazioni, vincoli legislativi, ecc.)

Criticità dei requisiti non funzionali

- Limitano le proprietà complessive del sistema
  - **Criticità**: gli utenti si possono adattare o possono aggirare problemi legati a funzioni che non corrispondono appieno alle loro necessità, ma non soddisfare i requisiti non funzionali può significare che l'intero sistema sia inutilizzabile
    - Es. Se il sistema di gestione del traffico aereo non soddisfa i requisiti di affidabilità non potrà ottenere la certificazione di sicurezza necessaria per operare

Esempi di Requisiti non funzionali

Es.1 Il tempo di risposta del sistema all'inserimento della password utente deve essere inferiore a 10 sec

Es.2 I documenti di progetto (deliverable) devono essere conformi allo standard XYZ-ABC-12345

Es.3 Il sistema software non deve rilasciare ai suoi operatori nessuna informazione personale relativa ai clienti, tranne nominativo e identificatore

Esempio del bancomat

- Si consideri un sistema bancomat. Il servizio deve mettere a disposizione le funzioni di prelievo, saldo, estratto conto. Il sistema deve essere disponibile a persone portatori di Handicap, deve garantire un tempo di risposta inferiore al minuto, e deve essere sviluppato su architettura X86 con sistema operativo compatibile con quello della Banca. Le operazioni di prelievo devono richiedere autenticazione tramite un codice segreto memorizzato sulla carta. Il sistema deve essere facilmente espandibile, e adattabile alle future esigenze bancarie. .....

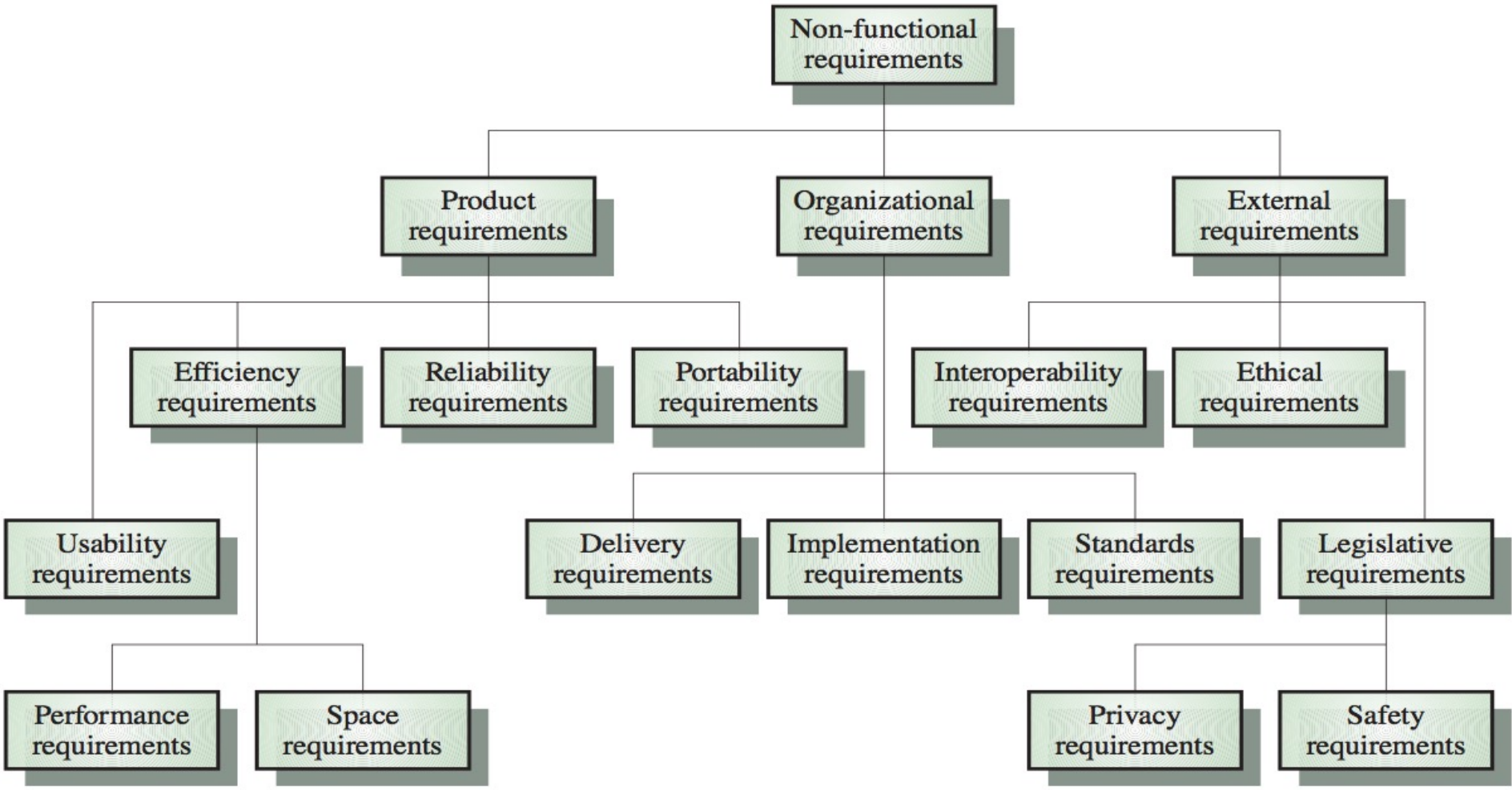
Requisiti funzionali

- Si consideri un sistema bancomat. Il servizio deve mettere a disposizione le funzioni di prelievo, saldo, estratto conto. Il sistema deve essere disponibile a persone portatori di Handicap, deve garantire un tempo di risposta inferiore al minuto, e deve essere sviluppato su architettura X86 con sistema operativo compatibile con quello della Banca. Le operazioni di prelievo devono richiedere autenticazione tramite un codice segreto memorizzato sulla carta. Il sistema deve essere facilmente espandibile, e adattabile alle future esigenze bancarie. .....

Requisiti non funzionali

- Si consideri un sistema bancomat. Il servizio deve mettere a disposizione le funzioni di prelievo, saldo, estratto conto. Il sistema deve essere disponibile a persone portatori di Handicap, deve garantire un tempo di risposta inferiore al minuto, e deve essere sviluppato su architettura X86 con sistema operativo compatibile con quello della Banca. Le operazioni di prelievo devono richiedere autenticazione tramite un codice segreto memorizzato sulla carta. Il sistema deve essere facilmente espandibile, e adattabile alle future esigenze bancarie. .....

Classificazione Reqs. non funzionali



Verificabilità

- Requisiti non funzionali difficili da verificare
  - Obiettivi **generali**: Facilità d'uso, la capacità di ripristino del sistema, risposta rapida, etc.
  - Obiettivi **vaghi** che causano problemi agli sviluppatori in quanto lasciano loro l'interpretazione e la successiva discussione quando il sistema viene consegnato

- **Obiettivo di sistema**
  - Il sistema dovrebbe essere semplice da usare per i controllori esperti, e organizzato in modo che gli errori dell'utente siano minimizzati
- **Requisito non funzionale verificabile**
  - I controllori esperti dovrebbero essere capaci di usare tutte le funzioni del sistema dopo un addestramento di due ore. Dopo questo addestramento, gli utenti esperti non dovrebbero superare, in media, i 2 errori al giorno

Misurazione quantitative

Proprietà	Misura
Velocità	Transazioni elaborate al secondo Tempi di risposta a utenti/eventi Tempo di refresh dello schermo
Dimensione	Kbyte, Mbyte, Gbyte Numero di circuiti RAM
Facilità d'uso	Tempo di addestramento Numero di maschere di aiuto
Affidabilità	Tempo medio di malfunzionamento Probabilità di indisponibilità Tasso di malfunzionamento Disponibilità
Robustezza	Tempo per il riavvio dopo il malfunzionamento Percentuale di eventi causanti malfunzionamento Probabilità di corruzione dei dati dopo malfunzionamento
Portabilità	Percentuali di dichiarazioni dipendenti dall'architettura di destinazione Numero di architetture di destinazione

Proviamo insieme

- Requisiti funzionali
  - ….

- Requisiti non funz.
  - ….

Il documento di *analisi dei requisiti* (o documento di *specifica)*

- Documento ufficiale che descrive in dettaglio le caratteristiche del sistema da sviluppare
- Include sia la definizione dei requisiti che la loro specifica
- Descrive **COSA** il sistema deve fornire (**dominio *del problema)*** e non **COME** il sistema deve essere sviluppato (**dominio *della soluzione)***

La forma del requisito

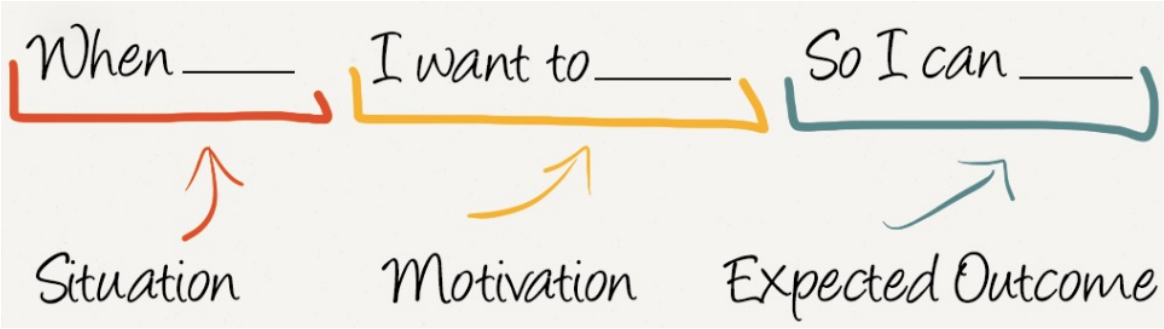
- **Classica**: *l'app permetterà di gestire le autorizzazioni di accesso alle strutture UNITN*
- **Userstory**: *come studente voglio richiedere l'autorizzazione per andare a lezione*
- **Caso d'uso**: uno studente interroga la funzione ricerca edificio per individuare l'edificio dove si trova l'aula, l'app chiede informazioni di ricerca …

User stories

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

  As a <*user type*>
  I want <*to do something*>
  so that <*I get some value*>

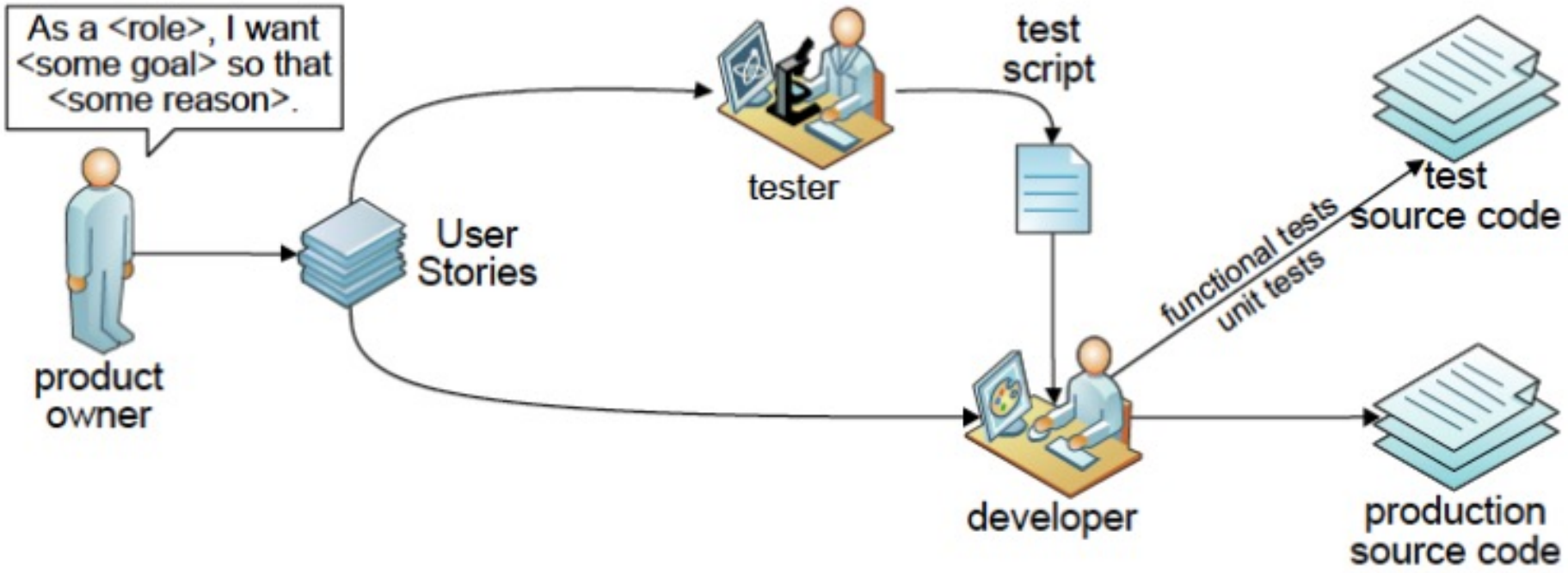
  Come *studente UNITN (ruolo)*
  Voglio *richiedere autorizzazione (scopo o funzione)*
  Per *posso accedere all'aula (valore ottenuto)*



A cosa servono le user story

- 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 e prepara tutto il necessario perché il PO possa accettare il software risultante, così come richiesto dalla Definition of Done

User story nel processo agile



I requisiti con UML: use case

- *Studio degli scenari operativi degli utenti di un sistema*
  - "modi" in cui il sistema può essere utilizzato (cioè definiscono le operazioni o funzioni che il sistema mette a disposizione dei suoi utilizzatori)
- Rappresenta un **requisito funzionale**
- Esplicita cosa ci si aspetta da un sistema ("what?")
- Nasconde il comportamento del sistema ("how?")
- E' una sequenza di **azioni** che producono un risultato osservabile da un **attore**

Use case

Si usa per

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

- 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*

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**

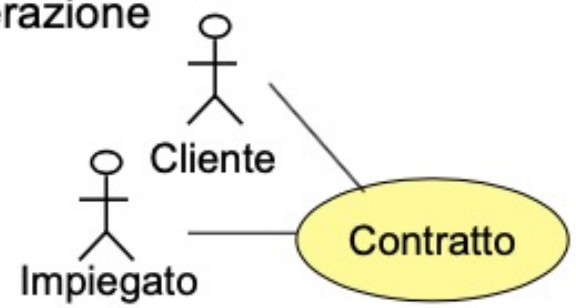
Esempio

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



Use case 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

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 (b) se il cliente (o uno dei diversi intestatari) non è censito l'addetto provvede a registrarlo, richiede al cliente la firma e ne effettua la memorizzazione via scanner

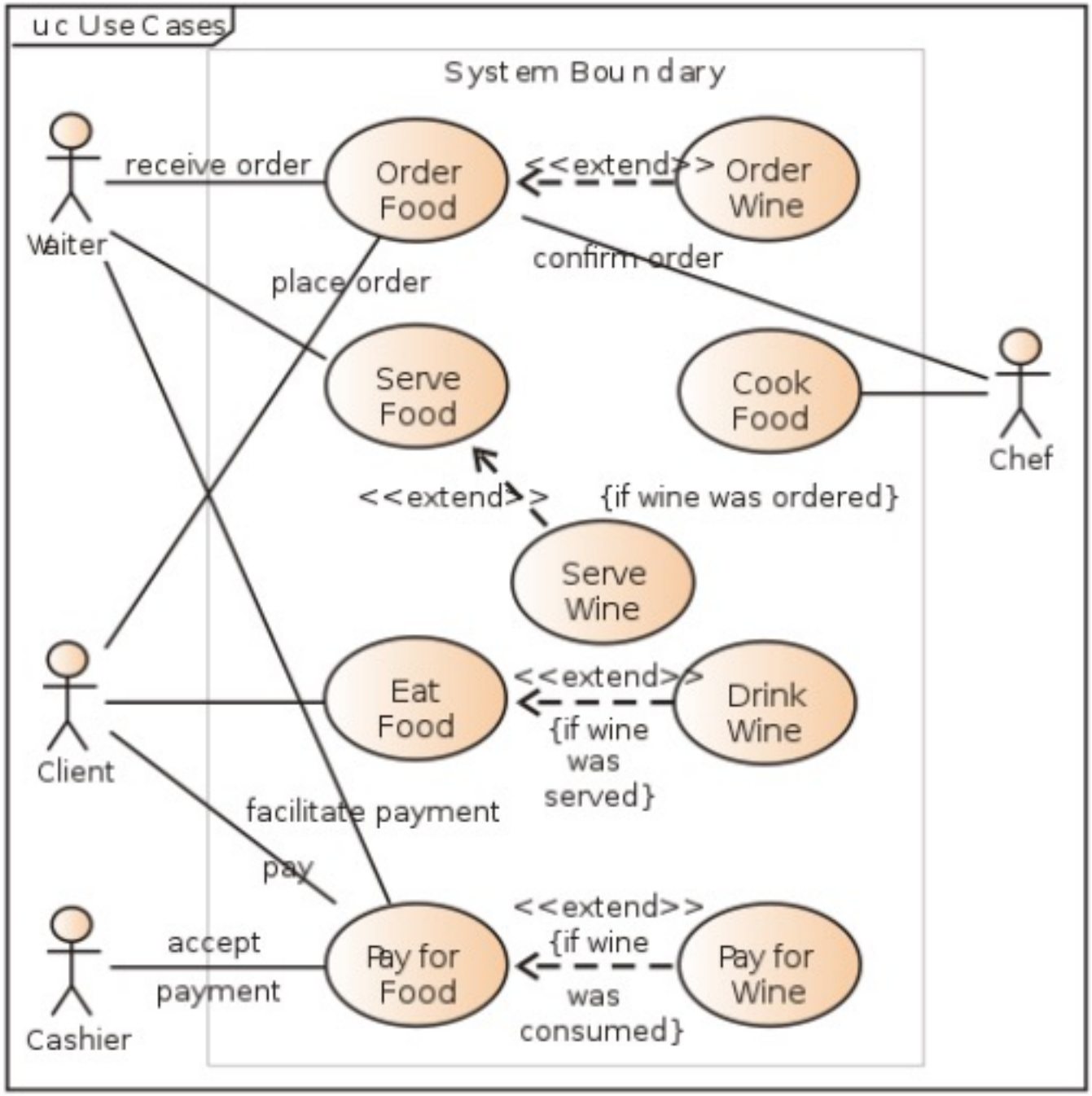
Esempio: apertura conto corrente (+)

(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


uc Use Cases
System Boundary
Waiter
receive order
Order Food
<<extend>>
Order Wine
confirm order
place order
Serve Food
Cook Food
Chef
<<extend>>
{if wine was ordered}
Serve Wine
Client
Eat Food
<<extend>>
Drink Wine
{if wine was served}
facilitate payment
pay
Cashier
accept payment
Pay for Food
<<extend>>
{if wine was consumed}
Pay for Wine

Gestione dei requisiti

- Conflitti
- Relazioni
- Tracciabilità
- Cambiamenti
- Documentazione