UNIVERSITÀ DEGLI STUDI DI SALERNO

Dipartimento di Ingegneria dell'Informazione ed Elettrica e Matematica applicata

Corso di Laurea in Ingegneria Informatica

APPUNTI DI INGEGNERIA DEL SOFTWARE

DI FRANCESCO PIO CIRILLO

https://github.com/francescopiocirillo

“Non ti abbattere mai”

DISCLAIMER

Questi appunti sono stati realizzati a scopo puramente educativo e di condivisione della conoscenza. Non hanno alcun fine commerciale e non intendono violare alcun diritto d'autore o di proprietà intellettuale.

I contenuti di questo documento sono una rielaborazione personale di lezioni universitarie, materiali di studio e concetti appresi, espressi in modo originale ove possibile. Tuttavia, potrebbero includere riferimenti a fonti esterne, concetti accademici o traduzioni di materiale didattico fornito dai docenti o presente in libri di testo.

Se ritieni che questo documento contenga materiale di tua proprietà intellettuale e desideri richiederne la modifica o la rimozione, ti invito a contattarmi. Sarò disponibile a risolvere la questione nel minor tempo possibile.

In quanto autore di questi appunti non posso garantire l'accuratezza, la completezza o l'aggiornamento dei contenuti p e non mi assumo alcuna responsabilità per eventuali errori, omissioni o danni derivanti dall’uso di queste informazioni. L’uso di questo materiale è a totale discrezione e responsabilità dell’utente.

Requirements Engineering

Requirements Engineering (l'Ingegneria dei Requisiti)

L'Ingegneria dei Requisiti è il processo di determinazione dei servizi che un cliente richiede da un sistema e delle limitazioni (constraints) sotto le quali deve operare e deve essere sviluppato il sistema.

> 💡 I **Requisiti** sono la descrizione dei servizi e delle limitazioni del sistema che sono generati durante il processo, descrivono cosa il software dovrebbe fare ma non come è costruito.

Il termine requisito è in realtà molto ampio, può essere una complessa e dettagliata formula matematica ma anche una statement astratta ad alto livello riguardo un servizio o un constraint.

I Requisiti possono essere:

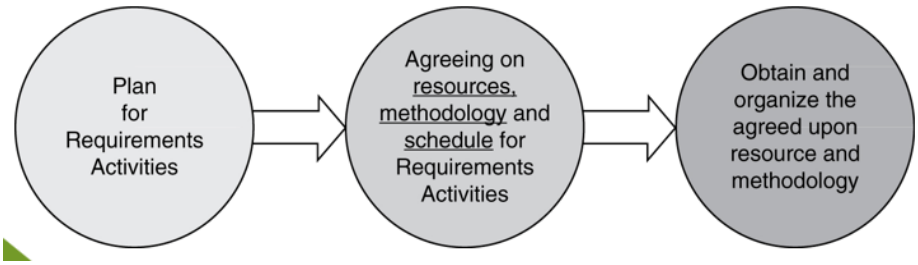
- la base per l'offerta di un contratto (nel qual caso devono essere liberi all'interpretazione)
- la base per un vero e proprio contratto (nel qual caso devono essere definiti in dettaglio).

Prima di mettere in pratica l'Ingegneria dei Requisiti è importante una **fase di pianificazione nella quale bisogna definire**:

- i Processi di Requirements Engineering che saranno utilizzati;
- le **Risorse** necessarie;
- una **Schedule** per il completamento delle attività di Requirements.

Una volta definiti questi dettagli bisogna effettuarne una review e farli accettare a tutte le parti coinvolte.

Alcune aziende svolgono il Requirements Engineering come un'attività separata dal resto dei progetti e infatti spesso è un'attività prezzata a parte rispetto allo svolgimento dei progetti, con la possibilità di ripiegare il prezzo di questa attività direttamente all'interno del prezzo completo del progetto se effettivamente si decide di proseguire oltre il Requirements Engineering.



traduzione seconda bolla: "mettersi d'accordo sulle risorse, metodologie e tempi per le attività di requirements"

Requisiti funzionali e non funzionali

Requisiti funzionali (cosa):

- dichiarazioni dei servizi che il sistema dovrebbe fornire;
- come il sistema dovrebbe reagire a determinati input e come il sistema dovrebbe comportarsi in determinate situazioni;
- potrebbero affermare cosa il sistema non dovrebbe fare.

Requisiti non funzionali (come):

- vincoli su servizi o funzioni offerti dal sistema come vincoli temporali,
- vincoli sugli strumenti di sviluppo, conformità a degli standard,
- vincoli di sicurezza ecc.

Spesso si applicano al sistema nel suo insieme piuttosto che a singole caratteristiche o servizi.

L'importanza della documentazione dei requisiti

Le attività di Progettazione e Implementazione necessitano di requisiti chiari, la documentazione è quindi molto importante, questa aiuta anche ad **evitare potenziali scope-creep** (vale a dire situazioni nelle quali il progetto tende ad estendersi sempre di più sforando in tempi e budget).

La Documentazione dei Requisiti permette di creare **casi e scenari di test,** questo è particolarmente vero per i grandi sistemi, dove spesso il team di test e il team di sviluppo non coincidono.

La Documentazione aiuta inoltre a **segmentare un progetto grande**, a prioritizzare le release e semplificare il project management.

Inoltre al di fuori delle attività di progettazione e sviluppo la documentazione dei requisiti permette di **creare materiale per lo user training**, materiale per il **marketing** e documentazione per il supporto e la manutenzione.

Una delle più importanti ragioni per le quali i progetti software falliscono sono i Requirements incompleti, incorretti o mal compresi.

Caratteristiche dei singoli Requirements

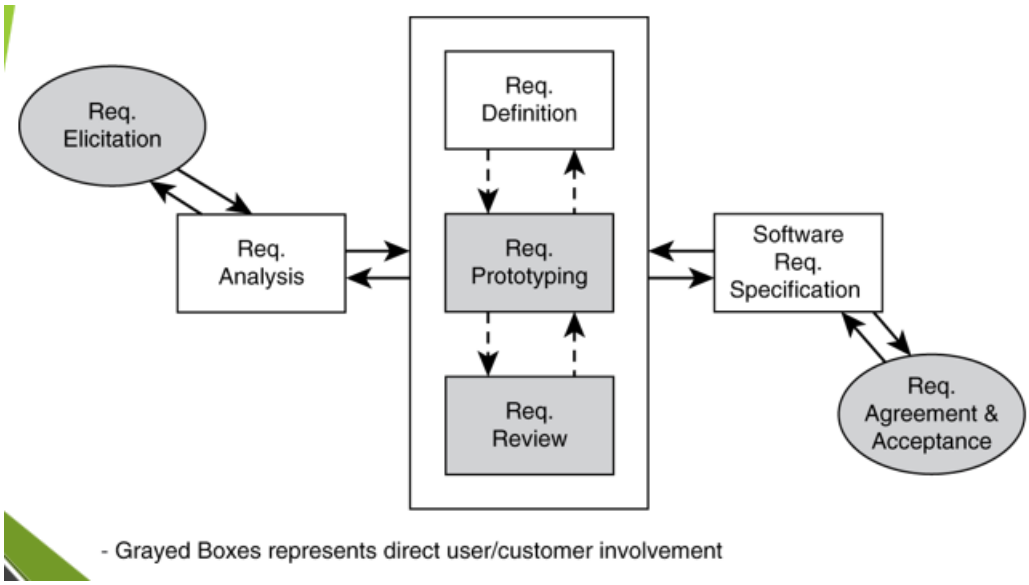
1. **Necessary**. Il requirement definisce una capacità essenziale, una caratteristica, una limitazione o un fattore di qualità.
2. **Appropriate**. La specifica intenzione e la quantità di dettaglio del Requirement è appropriata al livello dell'entità alla quale si riferise (livello di astrazione appropriato al livello dell'entità)
3. **Unambiguous.** Il requisito è affermato in un modo che può essere interpretato in un solo modo.
4. **Complete**. Il requisito descrive sufficientemente le capacità necessarie, le caratteristiche, le limitazioni e i fattori di qualità da rispettare per venire in contro alle necessità dell'entità.
5. **Singular**. Il requisito afferma una singola capacità, caratteristica, constraint, quality factor.
6. **Feasible**. Il requisito può essere realizzato entro i constraints del sistema (costi, schedule, limitazioni tecniche) con un rischio accettabile.
7. **Verifiable**. Il requisito è strutturato ed espresso in un modo che permette di verificare la sua realizzazione rispetto alla soddisfazione del cliente al livello al quale esistono i requisiti.

8. **Correct.** Il requisito è una rappresentazione accurata dell'entity need dal quale è trasformato.
9. **Conforming**. I singoli items si conformano ad uno standard e uno stile approvati per la scrittura dei requisiti, quando applicabili.

Caratteristiche di un Set di Requirements

1. **Complete**. Il set di requirements da solo descrive sufficientemente le capacità necessarie, le caratteristiche, i constraints e i fattori di qualità necessari per venire in contro alle entity needs senza necessità di ulteriori informazioni.
2. **Consistent**. Il set di requisiti contiene requisiti individuali che sono unici, non si sovrappongono né contraddicono tra loro e le unità e i sistemi di misurazione utilizzati sono omogenei.
3. **Feasible**.(fattibile) Il set di requisiti nella sua interezza può essere realizzato entro le limitazioni dell'entità (costi, schedule, limitazioni tecniche) con un rischio accettabile.
4. **Comprehensive**. Il set di requisiti è scritto in modo tale che è chiaro cosa ci si aspetta dall'entità e dalla sua relazione con il sistema del quale fa parte.
5. **Able to be validated** (verificabile). È possibile che soddisfare il set di requirements porterà al soddisfacimento dei bisogno dell'entità entro i constraints (costi, schedule, limitazioni tecniche, legali e legislative).

Schema delle attività di Requirements Engineering



REQUIREMENTS ELICITATION

Alcuni Requirements sono forniti ai Software Engineers, questi sono ad esempio i requisiti iniziali del prodotto/sistema ma anche ad esempio i requisiti già preliminarmente stabiliti per una seconda o terza follow-on release di una sequenza pianificata di un prodotto software. Requisiti dati sono anche quelli che vengono forniti come parte di una richiesta di preventivo per lo sviluppo di un software.

I requisiti potrebbero anche essere stabiliti dai Software Engineers stessi, questo perché spesso gli utenti hanno una comprensione dei requisiti limitata unicamente ai loro tasks lavorativi e tendono talvolta a fornire requisiti incompleti e contraddittori, inoltre la logica aziendale e gli obiettivi da raggiungere potrebbero non essere del tutto chiari all'utente, stabilire questi punti aiuta a dare priorità ai giusti aspetti del progetto.

Il processo di Elicitazione dei Requisiti coinvolge staff tecnici che lavorando con i clienti comprende **qual è il dominio dell'applicazione e quali sono i servizi che il sistema dovrebbe fornire come anche i suoi constraints operazionali**.

In generale il processo di Requirements Elicitation può coinvolgere, end-users, managers, ingegneri impegnati nella manutenzione, esperti del dominio, sindacati and so on, queste figure prendono il nome di **stakeholders** (che significa azionisti).

High-Level Requirements Elicitation

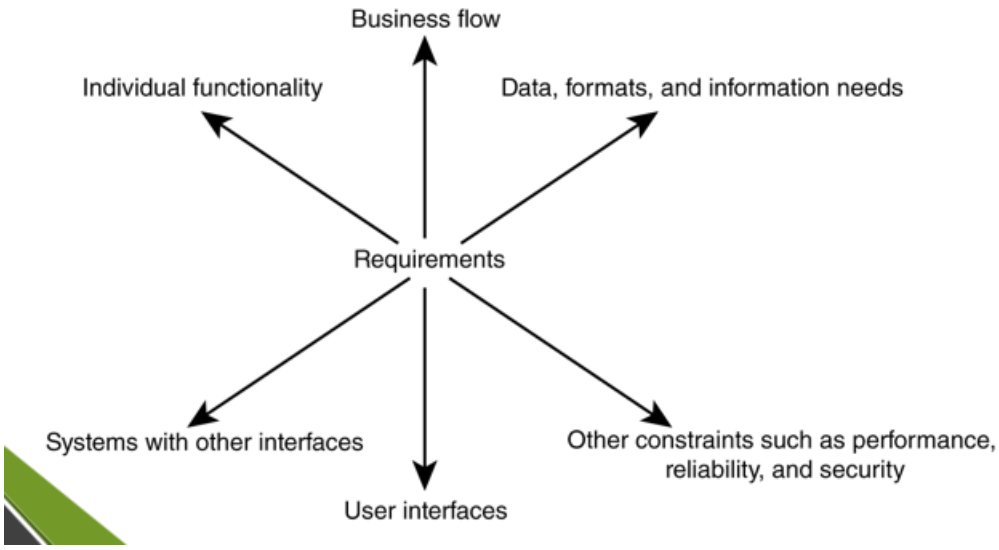
I Software Engineer che interagiscono con il Business Management e gestiscono i requisiti sono spesso chiamati **Business Analysts**.

È necessario comprendere la percezione e gli obiettivi di business e management legati al progetto software con un focus sui seguenti punti:

- opportunità e bisogni di business;
- giustificazione del progetto;
- scope (ambito);
- principali limitazioni;
- principali funzionalità;
- fattori di successo;
- caratteristiche utente.

Detailed Requirements Elicitation

Elicitazione e raccolta dei dettagli riguardanti le necessità e i desideri degli utenti.



▼ Definizione di chatgpt del business flow

Il business flow, o flusso di lavoro aziendale, è il percorso sequenziale delle attività o dei processi che avvengono all'interno di un'organizzazione per raggiungere un obiettivo specifico. Questo può coinvolgere diverse fasi o passaggi che vengono eseguiti in modo ordinato per completare un'operazione o produrre un determinato risultato. Essenzialmente, rappresenta il percorso che le informazioni, le risorse e le azioni seguono all'interno di un'azienda.

Problemi con l'elicitazione dei requisiti

1. Gli stakeholder non sanno (a volte) cosa vogliono veramente;
2. Gli stakeholder esprimono i requisiti a modo loro;
3. Diversi stakeholder possono avere requisiti in conflitto;
4. Fattori organizzativi e politici possono influenzare i requisiti del sistema;
5. I requisiti cambiano durante il processo di analisi;
6. Possono emergere nuovi stakeholder e l'ambiente aziendale può cambiare.

Interviste ed Etnografia

Le interviste sono un **potente strumento** per acquisire una comprensione generale di cosa fanno gli Stakeholders e come potrebbero interagire con il sistema, una interview può essere:

- closed (basata su una lista predeterminata di domande)
- open (nella quale vari problemi sono esplorati insieme agli stakeholders).

Al fine di rendere le interviste efficaci è fondamentale **avere una mente aperta** ed evitare di approcciare l'intervista con dei preconcetti riguardo i requisiti, focalizzandosi invece sull'ascolto degli stakeholders e spingendoli a parlare del sistema suggerendo dei requisiti piuttosto che semplicemente chiedendogli cosa vogliono.

Alcuni possibili problemi possono essere rappresentati dall'uso degli stakeholders di un linguaggio non semplice da comprendere per il software engineer.

Importante è l'analisi del modo i cui le persone lavorano, molti requisiti sono più facili da ricavare da una comprensione profonda di fattori organizzativi e sociali difficili da spiegare ma facili da osservare. L'osservazione permette inoltre di capire processi già esistenti e identificare possibili nuove funzioni da aggiungere al sistema.

REQUIREMENTS ANALYSIS

Una volta elicitati e conservati i requirements questi sono ancora un set di dati non organizzato.

L'operazione di Requirements Analysis consiste nel categorizzare i requisiti, assegnargli diverse priorità e assicurarsi che siano coerenti e completi.

Una prima categorizzazione dei requisiti è quella che li divide in requisiti funzionali e non funzionali, ma un'altra più dettagliata si concentra sulle 6 dimensioni (prime cinque funzionali e ultima non funzionale) dei requisiti:

- funzionalità individuale;
- flusso di lavoro (scenari di "utilizzo");
- necessità di dati e informazioni;
- interfacce utente;
- altre interfacce con sistemi/piattaforme esterni;
- ulteriori vincoli come prestazioni, sicurezza, affidabilità, ecc.

Si può usare uno schema di categorizzazione con dei prefissi e dei numeri per meglio categorizzare i requisiti:

Requirement Area	Prefix	Requirement Statement Numbering
Individual functionality	IF	IF-1.1, IF-1.2, IF-1.3, IF-2.1
Business flow	BF	BF-1, BF-2
Data and data format	DF	DF-1.1, DF-1.2, DF-2.1
User interface	UI	UI-1.1, UI-1.2, UI-2.1
Interface to systems	IS	IS-1
Further constraints	FC	FC-1

Si inizia con la categoria **Business Flow** (BF) e si associano gli altri requisiti al business flow attraverso il primo numero. Si usa il secondo numero quando c'è più di un requisito di una stessa categoria legato al Business Flow. Ad esempio IF-1.1, IF-1.2 e IF-1.3 sono funzionalità individuali del Business Flow BF-1.

Quando poi **più di un** Business Flow è interessato da un requisito si sceglie quello più legato.

Se al contrario un requisito non si integra con **nessun** business flow si referenzia un business flow null.

Use Case Diagram

I Use Case Diagrams possono essere utilizzati per descrivere visivamente:

- le funzionalità del sistema principali
- i ruoli (attori) che interagiscono con ogni funzionalità
- le boundary (confine, limite) del sistema.

In questa fase ogni Use Case è descritto solo brevemente, la descrizione è poi arricchita nelle fasi successive con:

- **precondizioni e postcondizioni** per le funzionalità;
- **flusso di eventi** di ogni funzionalità;
- condizioni di errore e **flussi alternativi**.

Requirements Prioritization

Spesso alcune limitazioni rendono impossibile sviluppare tutti i requisiti (tempo, risorse, capacità tecniche ecc...) e quindi bisogna dare delle **priorità** ai requisiti. Nei modelli iterativi e agili l'assegnazione di priorità aiuta a decidere quali features vanno sviluppate prima.

Ci sono alcuni criteri che aiutano ad assegnare le priorità:

- richieste e **necessità del cliente**/utente attuale;
- condizioni della concorrenza e del **mercato attuale**;
- anticipazione dei **bisogni futuri dei nuovi clienti**;
- vantaggi sulle **vendite**;
- esistenza di **problemi critici** nel prodotto attuale.

I criteri sono spesso soggettivi, le priorità dovrebbero essere assegnate con l'aiuto di molti stakeholders (per avere molti punti di vista).

Spesso è utile usare la **lista delle priorità dei requisiti**:

Requirement Number	Brief Requirement Description	Requirement Source	Requirement Priority*	Requirement Status
1	One-page query must respond in less than 1 second	A major account marketing representative	Priority 1	Accepted for this release
2	Help text must be field sensitive	Large account users	Priority 2	Postponed for next release

*Priority may be 1, 2, 3, or 4, with 1 being the highest.

Molto usato è l'Analytical Hierarchical Process (processo analitico gerarchico), in cui ogni requisito è comparato con gli altri. Per farlo viene assegnato un valore di intensità alla relazione tra due requisiti, i valori di intensità vanno da 1 (stesso valore) a 9 (valore estremamente più alto), i reciproci sono usati per esprimere valori a più bassa intensità.

	Req 1	Req 2	Req 3
Req 1	1	2	3
Req 2	1/2	1	2
Req 3	1/3	1/2	1

Ogni valore è poi diviso per la somma di tutti i valori della sua colonna, infine il valore di ogni requisito è calcolato come la somma di tutti i valori sulla sua riga, diviso per il numero di colonne.

	Req 1	Req 2	Req 3
Req 1	1	2	3
Req 2	1/2	1	2
Req 3	1/3	1/2	1
	1.83	3.5	6.0

	Req 1	Req 2	Req 3
Req 1	0.55	0.57	0.5
Req 2	0.27	0.29	0.33
Req 3	0.18	0.14	0.17

1.62 / 3 = 54%

0.89 / 3 = 30%

0.49 / 3 = 16%

L'Analisi dei requisiti dovrebbe assicurare che i requisiti sono **Completi** (dovrebbero includere descrizioni di tutte le facilities necessitate) e **Coerenti** (non ci dovrebbero essere contraddizioni né conflitti nella descrizione delle facilities del sistema).

> Facilities può significare strutture, mezzi, impianti, attrezzature.

Nella pratica però assicurare Completezza e Coerenza è quasi impossibile a causa della complessità del sistema e dell'ambiente.

REQUIREMENTS DEFINITION

View-Oriented Requirements Definition

VORD (definizione dei requisiti orientata alla vista), si basa sul concetto che i requisiti sono visti in modo diverso da persone divere, bisogna quindi:

- **identificare gli stakeholders** e i loro punti di vista sui requisiti;
- **categorizzare i punti di vista** dei requisiti ed eliminare ogni duplicato;
- **rifinire** i punti di vista sui requisiti identificati;
- **mappare** i punti di vista sui requisiti rispetto al sistema e rispetto ai servizi che il sistema deve fornire.

Definizione dei Requisiti

Questa fase include la scrittura formale dei requisiti, possono essere usati diversi approcci:

1. linguaggio naturale;
2. linguaggio naturale strutturato (con tabelle Input-Process-Output);
3. Diagrammi Dataflow;
4. Entity Relations Diagram;
5. Scenari e User Stories;
6. UML (Diagrammi e Descrizioni Use Case, Class Diagrams e Sequence Diagrams sono i più usati in questa fase).

1. **Linguaggio Naturale**

I requisiti vengono scritti come una lista di frasi, è un buon metodo perché così i requisiti possono essere compresi dagli utenti e dai clienti, inoltre, il linguaggio naturale è universale, intuitivo ed espressivo. Tuttavia, manca di chiarezza ed è difficile essere precisi, inoltre si verifica l'amalgamazione dei requisiti (più requisiti vengono espressi insieme).

> Esempio della pompa di insulina:

> 3.2 The system shall measure the blood sugar and deliver insulin, if required, every 10 minutes. *(Changes in blood sugar are relatively slow so more frequent measurement is unnecessary; less frequent measurement could lead to unnecessarily high sugar levels.)*
>
> 3.6 The system shall run a self-test routine every minute with the conditions to be tested and the associated actions defined in Table 1. *(A self-test routine can discover hardware and software problems and alert the user to the fact the normal operation may be impossible.)*

2. **Linguaggio Naturale Strutturato**

I requisiti vengono scritti in un **form standard**, ogni campo del form fornisce informazioni su uno specifico aspetto dei requisiti.

> Esempio della pompa di insulina:

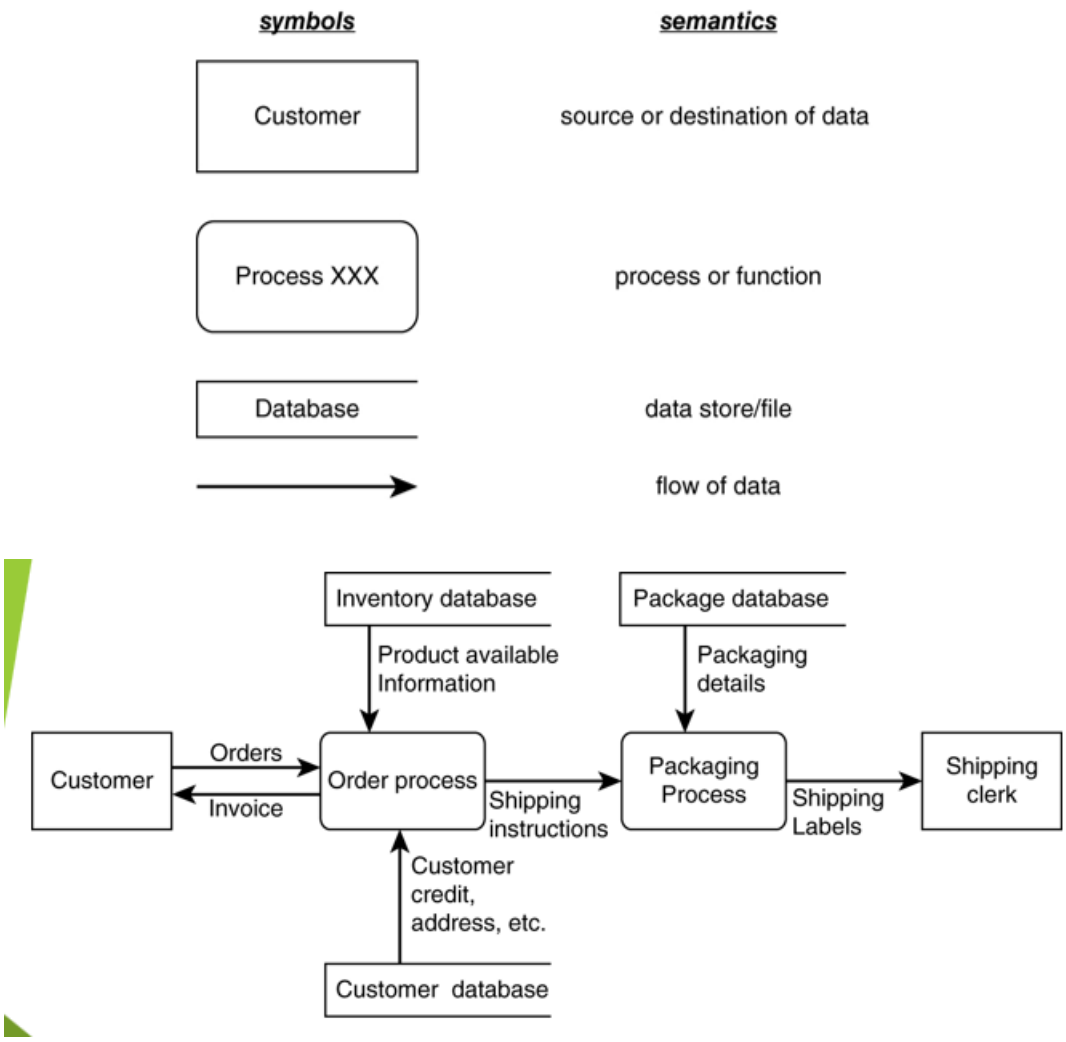
Insulin Pump/Control Software/SRS/3.3.2	
Function	Compute insulin dose: Safe sugar level.
Description	Computes the dose of insulin to be delivered when the current measured sugar level is in the safe zone between 3 and 7 units.
Inputs	Current sugar reading (r2), the previous two readings (r0 and r1).
Source	Current sugar reading from sensor. Other readings from memory.
Outputs	CompDose—the dose in insulin to be delivered.
Destination	Main control loop.
Action:	CompDose is zero if the sugar level is stable or falling or if the level is increasing but the rate of increase is decreasing. If the level is increasing and the rate of increase is increasing, then CompDose is computed by dividing the difference between the current sugar level and the previous level by 4 and rounding the result. If the result, is rounded to zero then CompDose is set to the minimum dose that can be delivered. (see Figure 4.14)
Requires	Two previous readings so that the rate of change of sugar level can be computed.
Precondition	The insulin reservoir contains at least the maximum allowed single dose of insulin.
Postcondition	r0 is replaced by r1 then r1 is replaced by r2.
Side effects	None.

> Esempio di tabella Input-Process-Output:

Requirement Number	Input	Process	Output
12: Customer order	• Items by type and quantity • Submit request	• Accept the items and respective quantities	• Display acceptance message • Ask for confirmation message

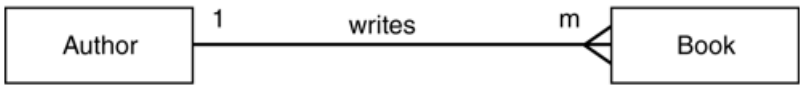
3. **Data Flow Diagrams**

Servono a catturare il Business Flow e il Data Flow delle funzionalità, di seguito Simboli, Semantica ed un Esempio.

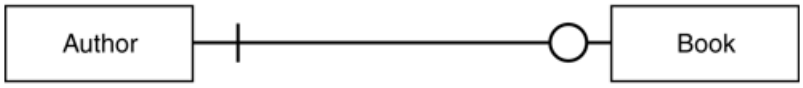


4. **Entity Relations Diagrams**

Identificano le entità del sistema principale e catturano le relazioni tra esse.

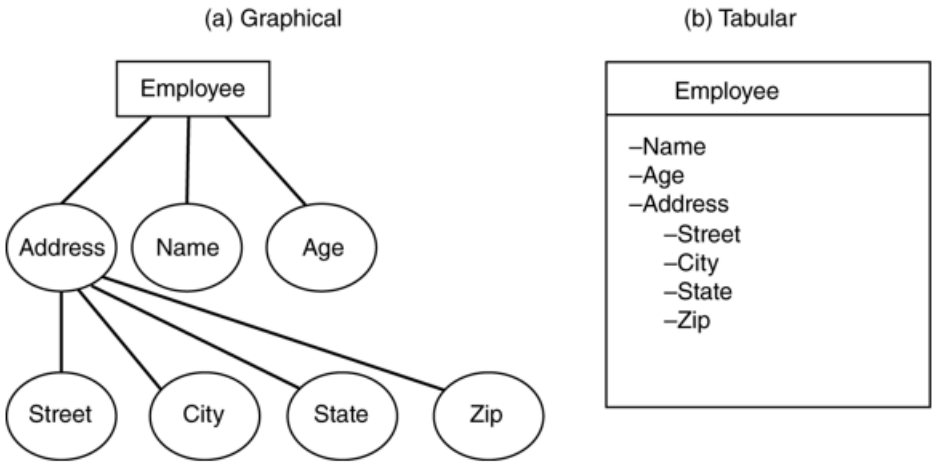


Cardinality: Specifies the number of occurrences of entities



Modality: Specifies the necessities of relationship to exist

Possono anche specificare gli attributi delle entità.



5. **User Stories and Scenarios**

Esempi veri di come dovrebbe essere usato un sistema per svolgere un task.

Le User Stories sono testuali, gli stakeholders possono immedesimarcisi e commentare sulla propria situazione rispetto alla storia.

Uno Scenario è una storia strutturata che include:

- una descrizione della **situazione di partenza**;
- una descrizione del **normale flusso** di eventi;
- una descrizione di cosa può **andare storto**;
- informazioni su **altre attività** concorrenti;
- una descrizione dello **stato quando lo scenario termina**.

▼ Esempio di User Story (tradotta da Bing AI)

> Jack è un insegnante di scuola primaria a Ullapool. Ha deciso che un progetto di classe dovrebbe essere incentrato sull'industria della pesca nella zona, esaminando la storia, lo sviluppo e l'impatto economico della pesca. Come parte di questo, agli studenti viene chiesto di raccogliere e condividere ricordi dei parenti, utilizzare archivi di giornali e raccogliere vecchie fotografie relative alla pesca e alle comunità di pescatori nella zona. Gli studenti utilizzano un wiki iLearn per raccogliere storie di pesca e SCRAN (un sito di risorse storiche) per accedere agli archivi dei giornali e alle fotografie. Tuttavia, Jack ha anche bisogno di un sito di condivisione di foto perché vuole che gli studenti scattino e commentino le foto degli altri e caricano le scansioni di vecchie fotografie che potrebbero avere nelle loro famiglie.

> Jack invia un'email a un gruppo di insegnanti di scuola primaria, del quale è membro, per vedere se qualcuno può consigliare un sistema appropriato. Due insegnanti rispondono e entrambi suggeriscono che utilizzi KidsTakePics, un sito di condivisione di foto che consente agli insegnanti di controllare e moderare i contenuti. Poiché KidsTakePics non è integrato con il servizio di autenticazione iLearn, crea un account docente e un account classe. Utilizza il servizio di configurazione iLearn per aggiungere KidsTakePics ai servizi visti dagli studenti della sua classe in modo che quando si collegano, possono immediatamente utilizzare il sistema per caricare foto dai loro dispositivi mobili e dai computer della classe.

▼ Esempio di Scenario (tradotto da Bing AI)

- Assunzione Iniziale:

Un utente o un gruppo di utenti hanno una o più fotografie digitali da caricare sul sito di condivisione delle immagini. Queste sono salvate su un tablet o un computer portatile. Sono riusciti a effettuare l'accesso a KidsTakePics.

- Normale:

L'utente sceglie di caricare le foto e viene invitato a selezionare le foto da caricare sul proprio computer e a selezionare il nome del progetto sotto il quale verranno archiviate le foto. Dovrebbe anche essere data l'opzione di inserire parole chiave che dovrebbero essere associate a ciascuna foto caricata. Le foto caricate vengono nominate creando una congiunzione del nome utente con il nome del file della foto sul computer locale.

Al termine del caricamento, il sistema invia automaticamente un'email al moderatore del progetto chiedendo loro di controllare i nuovi contenuti e genera un messaggio a schermo per l'utente che questo è stato fatto.

- Cosa può andare storto:

Non è associato alcun moderatore al progetto selezionato. Viene generata automaticamente un'email all'amministratore della scuola chiedendo loro di nominare un moderatore del progetto. Gli utenti dovrebbero essere informati che potrebbe esserci un ritardo nel rendere visibili le loro foto.

Le foto con lo stesso nome sono già state caricate dallo stesso utente. All'utente dovrebbe essere chiesto se desidera ricaricare le foto con lo stesso nome, rinominare le foto o annullare il caricamento. Se scelgono di ricaricare le foto, gli originali vengono sovrascritti. Se scelgono di rinominare le foto, viene generato automaticamente un nuovo nome aggiungendo un numero al nome del file esistente.

- Altre attività:

Il moderatore potrebbe essere collegato al sistema e potrebbe approvare le foto man mano che vengono caricate.

- Stato del sistema dopo il completamento:

L'utente è collegato. Le foto selezionate sono state caricate e assegnate allo stato 'in attesa di moderazione'. Le foto sono visibili al moderatore e all'utente che le ha caricate.

6. **UML**

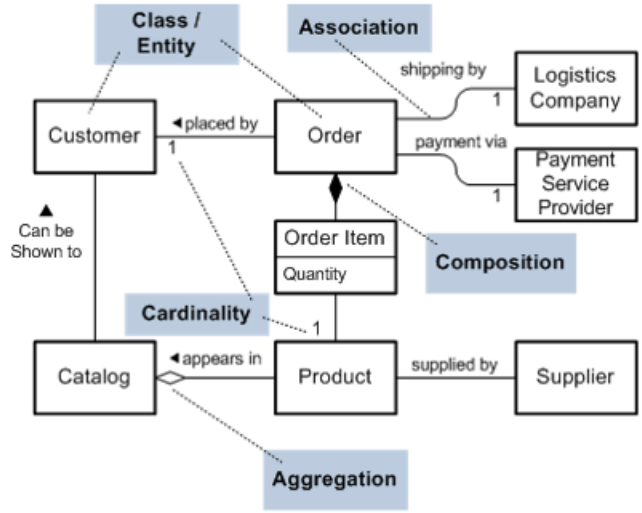
Le descrizioni dei casi d'uso sono usate in questa fase per identificare:

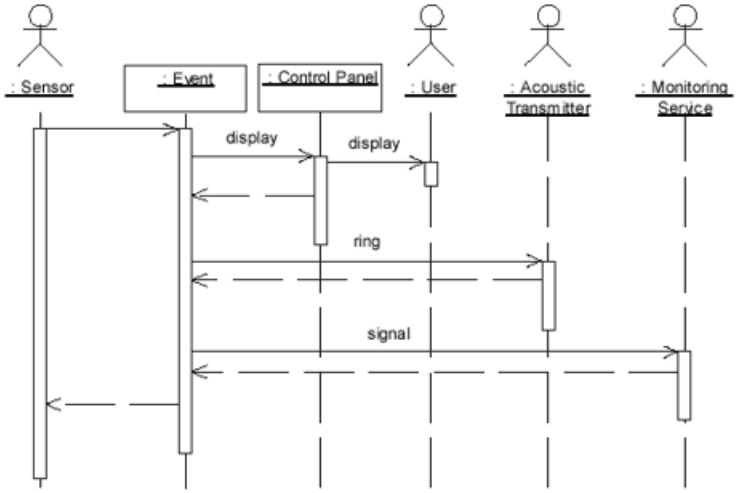
- funzionalità principali;
- precondizioni e postcondizioni delle funzionalità;
- flusso degli eventi (scenari);
- condizioni di errore e flussi alternativi.

Gli Use Case Diagrams (definiti nella fase di Analisi dei Requisiti) sono aggiornati in accordo con quanto appena svolto.

UC-0015	**Register Book Loan**	
Description	The system shall behave as described in the following use case when *a library user requests a loan of one or more books*.	
Precondition	*The library user has been identified by means of his or her identity card, has picked up the books to loan from the shelves, has not reached the maximum number of simultaneous loans and has no penalty.*	
Ordinary Sequence	**Step**	**Action**
	1	Actor *librarian requests the system for starting the book loan registering process.*
	2	The system *requests for the identification of the library user requesting a loan.*
	3	Actor *librarian provides identification data of the library user to the system.*
	4	The system *requests for the identification of the books to be loaned.*
	5	Actor *librarian provides identification data of the books to be loan to the system.*
	6	The system *displays the return date for each of the books to be loan and requests loan confirmation for each of them.*
	7	Actor *library user confirms the librarian which books he or she wants to loan after knowing return dates.*
	8	Actor *librarian re–confirms the book loans confirmed by the library user to the system.*
	9	The system *informs that the book loans have been successfully registered.*
Postcondition	*The library user can take the loaned books away and the system has registered the book loans.*	
Exceptions	**Step**	**Action**
	3	If *the library user has already reached the maximum number of simultaneous loans or has a penalty*, the system *informs of the situation*, then this use case *is cancelled.*

In UML si usano anche i Diagrammi di Classe per identificare le entità del sistema e le loro relazioni, si usano invece Sequence Diagrams e Communication Diagrams per modellare il flusso di informazioni tra entità del sistema.





Tracciabilità dei requisiti

L'abilità di ripercorrere i propri passi dopo lo sviluppo e verificare che tutti i requisiti siano stati sviluppati, testati, packaged (messi in package) e consegnati (delivered).

La tracciabilità permette di assicurare che il prodotto rispecchia a pieno i requisiti e anche se non è parte delle attività del processo dei requisiti bisogna iniziare a farlo nelle fasi iniziali.

I requisiti vanno tracciati dalla fonte (persone e documenti) fino agli ultimi passi (design, implementazione, test e release).

Ogni requisito deve essere reso univocamente verificabile e collegato al documento dal quale proviene o alla persona che l'ha creato.

Ogni design e artefatto di implementazione deve essere collegato al requisito che l'ha generato.

È possibile usare la cosiddetta tabella di tracciabilità:

Requirement	Design	Code	Test	Related Requirements
1	Component X	Module 3	Test Case 32	2, 3
2	Component Y	Module 5	Test Case 16	1
3	Component X	Module 2	Test Case 27	1
4	...	...	...	...

PROTOTIPIZZAZIONE E REVISIONE DEI REQUISITI

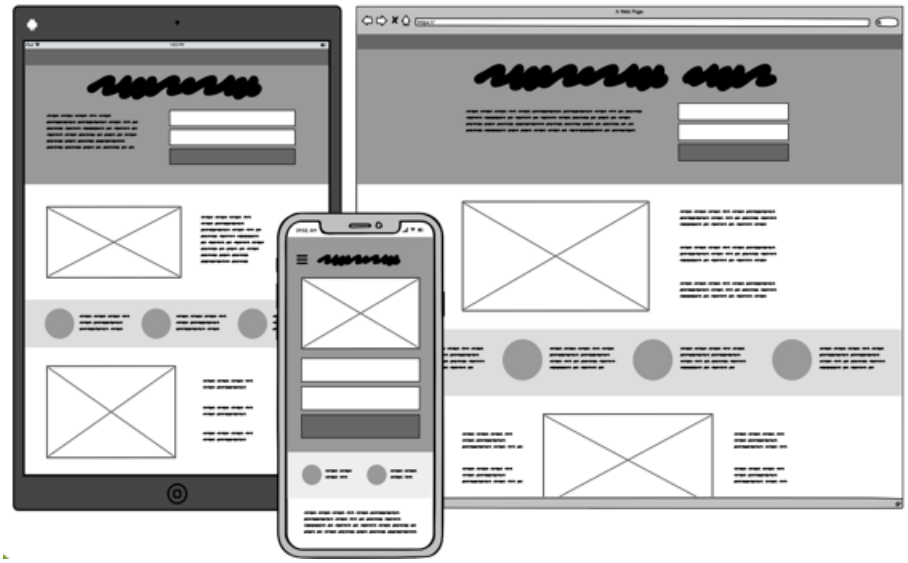
Prototipizzazione

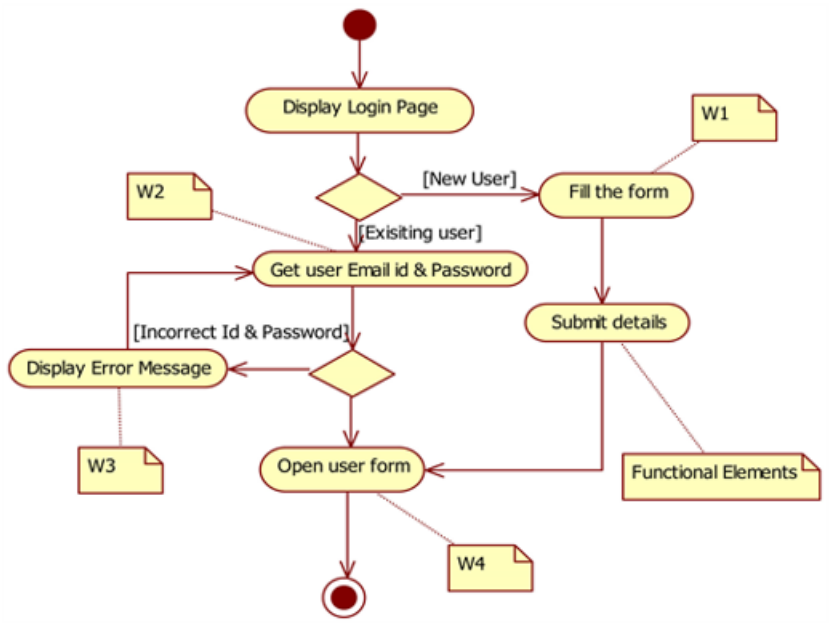
La prototipizzazione affronta i requisiti della User Interface in termini di Visual Look (dimensione, forma, posizione, colore) e di Interaction Flow (controllo e logica del flusso, profondità del flusso).

La prototipizzazione può essere fatta a

- **Low fidelity** (bassa fedeltà), usando degli sketch e descrivendo il flusso di interazione con dei diagrammi (Activity Diagrams),
- **High fidelity** (alta fedeltà), creando un'interfaccia preliminare usando strumenti di GUI design presenti in molte IDE come Visual Studio, Xcode ecc.

In basso esempi di Visual Look Prototyping e Interaction Flow Prototyping.





Revisione

Reviews regolari andrebbero tenute mentre la definizione dei requisiti sta essendo formulata.

Sia il cliente che lo staff dell'appaltatore dovrebbero essere coinvolti in questo processo, la buona comunicazione tra sviluppatori, clienti e utenti può permettere di risolvere i problemi nelle fasi iniziali del progetto.

Le reviews possono essere formali (con documentazioni complete) o informali.

I controlli più importanti da fare durante le reviews sono:

- verificabilità (il requisito è realisticamente testabile);
- comprensibilità (il requisito è appropriatamente compreso?);
- tracciabilità (l'origine del requisito è chiaramente espressa?);
- adattabilità (il requisito può essere cambiato senza un grande impatto sugli altri requisiti?).

REQUIREMENTS SPECIFICATION AND ACCEPTANCE

Una volta che i requisiti sono definiti, prototipizzati e revisionati, dovrebbero essere posti in un documento SRS (Software Requirements Specification).

La quantità di dettagli di questo documento dipendono da:

- **dimensioni e complessità del progetto;**
- conoscenza ed **esperienza degli sviluppatori** nell'area tematica;
- **releases successive** che sono state pianificate;
- numero atteso di attività di **customer support**.

ISO/IEC/IEEE 29148:2018

The ISO/IEC/IEEE 29148:2018 International standard "Systems and software engineering — Life cycle processes — Requirements engineering" specify that the **SRS document should includes the following information**

SRS overview
Purpose
Scope
Product perspective
Product functions
User characteristics
Limitations
Assumptions and dependencies
Apportioning of requirements
Specified requirements
External interfaces
Functions
Usability requirements
Performance requirements
Logical database requirements
Design constraints
Standards compliance
Software system attributes
Verification
Supporting information

Il documento SRS può essere “firmato” formalmente, firmando un contratto, o informalmente, comunicando un accordo.

Questa attività chiude la fase dei requisiti, fornisce una linea guida formale per le specifiche dei requisiti e fa da linea guida rispetto alla quale monitorare e controllare futuri cambiamenti.