- ***<u>Lezione 8 – (04/11/2024)</u>***

→ *<u>Terzo Blocco</u>*

I requisiti catturano quello che sono le aspettative del cliente.

I Requisiti software *(o software requirements)* è la descrizione dei servizi che un sistema software deve fornire, insieme ai vincoli da rispettare sia in fase di sviluppo che durante la fase di operatività del software.

Una definizione più standard che troviamo nel nostro "glossario di termini" usati nell'ambito dell'ingegneria del Software → *Def. IEEE Std 610.12 (1990):*

1. Una condizione o capacità necessaria a un utente per risolvere un problema o raggiungere un obiettivo
2. Una condizione o capacità che deve essere soddisfatta o posseduta da un sistema o componente di sistema per soddisfare un contratto, standard, specifica o altro documento formalmente imposto
3. Una rappresentazione documentata di una condizione o capacità come nella definizione (A) o (B)

I requisiti vengono generati applicando un processo di ingegneria dei requisiti *(requirements engineering)*

Per quanto riguarda l'astrazione dei requisiti*, Davis,* nel 1993 scrisse il seguente estratto

"If a company wishes to let a contract for a large software development project, it must define its needs in a sufficiently abstract way that a solution is not pre-defined. The requirements must be written so that several contractors can bid for the contract, offering, perhaps, different ways of meeting the client organisation's needs. Once a contract has been awarded, the contractor must write a system definition for the client in more detail so that the client understands and can validate what the software will do. Both of these documents may be called the requirements document for the system"

→ Esistono 2 tipologie di Requisiti:

- *Requisiti Utente (user requirements):* rappresentano una descrizione, in linguaggio naturale, con eventuale aggiunta di diagrammi, dei servizi che il sistema deve fornire e dei vincoli operativi. Questi requisiti sono scritti tenendo conto del cliente e spesso vengono sviluppati proprio insieme al cliente.

- *Requisiti di Sistema (system requirements):* sono una descrizione più dettagliata e formale, infatti si specificano mediante la stesura di un documento strutturato che descrive in modo dettagliato i servizi che il sistema software deve fornire. Questo documento rappresenta una sorta di "contratto" tra il cliente e il fornitore del servizio

software: descrive nel dettaglio tutto ciò che il sistema deve fare e diventa un punto di riferimento fondamentale durante la fase di sviluppo.
N.B: quando si parla di *specifica* si parla di requisiti di sistema.

Aggiungiamo dei termini ad altri già citati in precedenza:

- *Fornitore (supplier,contractor):* la persona/organizzazione che produce software per il cliente
- *Utente Finale (end-user):* la persona che interagisce direttamente con il prodotto software. Non corrisponde necessariamente al cliente

Un requisito nasce come forma astratta, generata dalla richiesta del cliente. Successivamente un team di Analisti Software lo trasformano in un requisito di sistema.

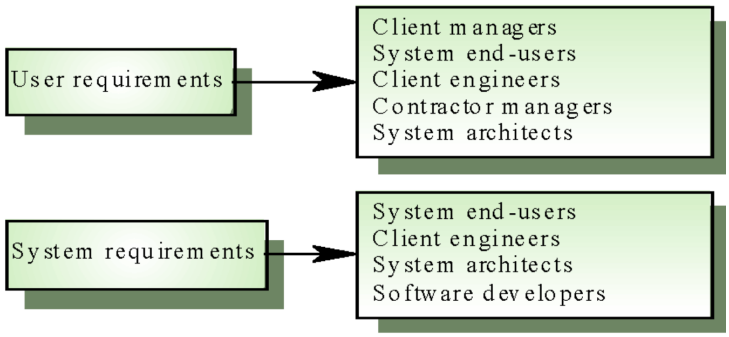
Esempi di Requisiti:

- *Requisito utente:* Il sistema software deve fornire un mezzo per rappresentare e visualizzare file esterni generati da altri tool. Ciò significa che l'utente oltre a desiderare di vedere la realizzazione del tool del proprio progetto, spesso vuole vedere pure tool precedenti.

- *Requisito di sistema:*

    1. L'utente deve avere la possibilità di definire il tipo dei file esterni
    2. Ad ogni tipo di file esterno deve essere associato il tool che lo ha generato
    3. Ogni tipo di file esterno deve essere rappresentato mediante una specifica icona sullo schermo
    4. L'utente deve avere la possibilità di definire l'icona che rappresenta il tipo di file esterno
    5. Quando l'utente seleziona un'icona che rappresenta un file esterno, deve poter essere eseguito il tool in grado di visualizzare il file

A seconda del tipo di requisito, cambia il tipo di lettore. Nella foto sottostante sono rappresentate tutte le figure di coloro che leggono uno dei due requisiti.



Esiste un'altra dimensione che possiamo usare utilizzare per identificare differenti categorie di requisiti. Ne abbiamo effettivamente 3, ma possiamo racchiuderle principalmente in 2.

- *Requisiti funzionali*

Questi 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.

Esempio1: Il sistema software deve fornire un appropriato visualizzatore per i documenti memorizzati

Esempio2: L'utente deve essere in grado di effettuare ricerche sia sull'intero insieme di basi di dati che su un loro sottoinsieme

Esempio3: Ad ogni nuovo ordine deve essere associato un identificatore unico *(Order_ID)*

- *Requisiti non funzionali*

Questi descrivono le proprietà del sistema software in relazione a determinati servizi o funzioni e possono anche essere relativi al processo:

- Caratteristiche di efficienza, affidabilità, safety, portabilità ecc. (tutte quelle caratteristiche del prodotto software che insieme definiscono il concetto di qualità del software)
- Caratteristiche del processo di sviluppo (standard di processo, uso di ambienti CASE, linguaggi di programmazione, metodi di sviluppo, ecc.)
- Caratteristiche esterne (intero per abilità con sistemi di altre organizzazioni, vincoli legislativi, ecc.)

Esempio1: Il tempo di risposta del sistema all'inserimento della password utente deve essere inferiore a 10 sec (requisito non funzionale, caratteristica di efficienza)

Esempio2: I documenti di progetto (deliverable) devono essere conformi allo standard XYZ-ABC-12345

Esempio3: Il sistema software non deve rilasciare ai suoi operatori nessuna informazione personale relativa ai clienti, tranne nominativo e identificatore

- *Requisiti di dominio (non è una caratteristica aggiuntiva)*

Questi requisiti derivati dal dominio applicativo del sistema software piuttosto che da necessità dettate dagli utenti. Questo può essere:

- *requisiti funzionali*, nuovi o adattatati, relativi al particolare dominio applicativo
- *requisiti non funzionali*, nuovi o adattati, relativi a standard esistenti o a procedure e regolamenti da applicare

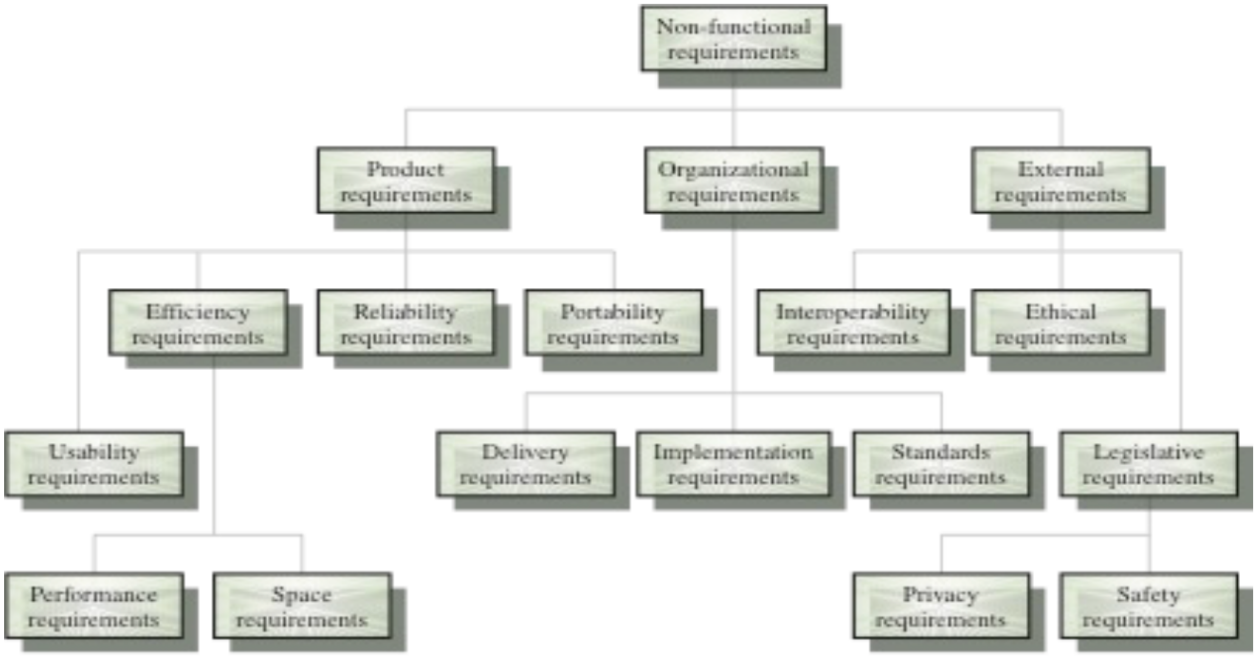
Esempio1: I documenti di rendiconto contabile, secondo la normativa XYZ.03, devono essere stampati alla ricezione e cancellati immediatamente

Esempio2: L'interfaccia utente per l'accesso al database magazzino deve essere conforme allo standard ZX.01

I requisiti non funzionali sono molto più articolati dei requisiti funzionali. Infatti esistono dei veri e propri modelli che permettono di classificare in modo più preciso i vari sottotipi dei requisiti non funzionali.

[N.B: nella seconda parte del corso ne vedremo uno in particolare]

In seguito vedremo una mappa gerarchica che classifica i requisiti non funzionali.



Di solito si adotta un approccio misto, poiché i requisiti non funzionali derivano dai requisiti funzionali. Ad esempio, se un requisito funzionale prevede che "il software debba offrire

una funzionalità di autenticazione", possiamo associargli un requisito non funzionale legato all'efficienza, come "il tempo di risposta del sistema all'inserimento della password da parte dell'utente deve essere inferiore a 10 secondi.

Quando si parla di definizione di un requisito si intente un requisito utente. Invece quando si parla di specifica di un requisito si intende un requisito di sistema.

Incorriamo a tre tipi di problema:

- l'utilizzo del linguaggio naturale, che può essere ambiguo, un requisito deve essere monostabile. Quindi qui si sfocia ad un problema di ambiguità, ovvero requisiti interpretabili in modo differente.

  Esempio1: specificare un tempo senza fornire il riferimento al fuso orario (in un'applicazione che gestisce chiamate intercontinentali)

  Esempio2: significato di "appropriato visualizzatore"

  Interpretazione utente: visualizzatore specifico per ogni tipo di documento

  Interpretazione sviluppatore: generico visualizzatore di testo che mostri il contenuto del documento

- Incompletezza: i requisiti non includono la descrizione di tutte le caratteristiche richieste

- Inconsistenza: conflitti o contraddizioni nella descrizione delle caratteristiche del sistema

  Esempio:

  - *Requisito1:* ogni form di input non deve contenere più di 5 campi editabili dall'utente *(requisito non funzionale)*
  - *Requisito2:* nella form di input relativa all'inserimento dei dati anagrafici l'utente deve introdurre i seguenti dati: nome, cognome, anno di nascita, luogo di nascita, indirizzo, telefono, fax, e-mail *(requisito funzionale)*

  Se soddisfo requisito 1 non posso soddisfare requisito 2, e viceversa, poiché vanno in contraddizione.

  I requisiti non funzionali espressi in modo generico dall'utente (es. il sistema software deve essere easy-to-use) possono risultare non quantificabili e difficili da verificare. Quindi è necessario esprimere i requisiti non funzionali usando una misura determinata che permetta di verificare quantitativamente se il requisito verrà soddisfatto dal sistema software.

  Seguono degli esempi di misure per requisiti non funzionali

Property	Measure
Speed	Processed transactions/second User/Event response time Screen refresh time
Size	K Bytes Number of RAM chips
Ease of use	Training time Number of help frames
Reliability	Mean time to failure Probability of unavailability Rate of failure occurrence Availability
Robustness	Time to restart after failure Percentage of events causing failure Probability of data corruption on failure
Portability	Percentage of target dependent statements Number of target systems

Come scrivere questi requisiti?

Per i requisiti utente, al di la che siano requisiti funzionali e non funzionali, devono essere espressi in modo da risultare comprensibili agli utenti del sistema sprovvisti di conoscenze tecniche, quindi scritti in linguaggio naturale.

Bisogna usare un formato standard per tutti i requisiti, usare il linguaggio naturale in modo consistente (es. uso di "deve" per requisiti necessari e "dovrebbe" per requisiti desiderabili), evidenziare le parti fondamentali di un requisito, evitare l'uso di termini tecnici.

Esempio di requisito utente:

3.5.1 Adding nodes to a design

3.5.1.1 **The editor shall provide a facility for users to add nodes of a specified type to their design.**

3.5.1.2 The sequence of actions to add a node should be as follows:

1. The user should select the type of node to be added.
2. The user should move the cursor to the approximate node position in the diagram and indicate that the node symbol should be added at that point.
3. The user should then drag the node symbol to its final position.

Rationale: The user is the best person to decide where to position a node on the diagram. This approach gives the user direct control over node type selection and positioning.

Specification: ECLIPSE/WS/Tools/DE/FS/Section 3.5.1

Esempio Tradotto:

3.5.1 Aggiunta di nodi a un progetto

3.5.1.1 L'editor deve fornire una funzione per consentire agli utenti di aggiungere nodi di un tipo specificato al proprio progetto.

3.5.1.2 La sequenza di azioni per aggiungere un nodo dovrebbe essere la seguente:

1. L'utente deve selezionare il tipo di nodo da aggiungere.
2. L'utente deve spostare il cursore nella posizione approssimativa in cui il nodo deve essere aggiunto nel diagramma e indicare che il simbolo del nodo deve essere aggiunto in quel punto.
3. L'utente deve quindi trascinare il simbolo del nodo nella posizione finale.

Razionale: L'utente è la persona più adatta a decidere dove posizionare un nodo nel diagramma. Questo approccio offre all'utente un controllo diretto sulla selezione e il posizionamento dei tipi di nodo.

Specifiche: ECLIPSE/WS/Tools/DE/FS/Sezione 3.5.1 (una specie di percorso)

Invece, per i requisiti di sistema sono descritti attraverso specifiche più dettagliate dei requisiti utente. Sono usati come base per il progetto software e possono essere espressi facendo uso di notazioni differenti.

Esistono 2 notazioni *semi-formali*:

- *PDL,* ovvero *Program description languages*, noto a noi come pseudocodice
- *Graphical notations*, ovvero tramite una serie di diagrammi

Fino ad arrivare a quelli più *formali* che sono le *specifiche matematiche*, in questo caso si utilizzano vere notazioni algebriche che richiedono grandi conoscenze ma hanno un vantaggio enorme in termini di verifica.

Esempio di requisito di sistema basato su form in linguaggio naturale:

ECLIPSE/Workstation/Tools/DE/FS/3.5.1

Function	Add node
Description	Adds a node to an existing design. The user selects the type of node, and its position. When added to the design, the node becomes the current selection. The user chooses the node position by moving the cursor to the area where the node is added.
Inputs	Node type, Node position, Design identifier.
Source	Node type and Node position are input by the user, Design identifier from the database.
Outputs	Design identifier.
Destination	The design database. The design is committed to the database on completion of the operation.
Requires	Design graph rooted at input design identifier.
Pre-condition	The design is open and displayed on the user's screen.
Post-condition	The design is unchanged apart from the addition of a node of the specified type at the given position.
Side-effects	None
Definition:	ECLIPSE/Workstation/Tools/DE/RD/3.5.1

[*N.B:* Un esempio su specifiche matematiche lo vedremo successivamente]

- ***Lezione 9 – (07/11/2024)***

Esempio basato su PDL (Java-like):

```
// DEFINIAMO LA CLASSE BANCOMAT

class ATM {

    // declarations here
    public static void main (String args[]) throws InvalidCard {
       try {

// LEGGI LA CARTA INSERITA → PUO LANCIARE UN ECCEZIONE INVALIDCARD
          thisCard.read () ;

// ATTRAVERSO L'OGGETTO KEYPAD INSERISCI IL PIN, E SI IMPOSTA LA
VARIABILE TENTATIVI = 1
          pin = KeyPad.readPin () ; attempts = 1 ;

// AVVIO UN CICLO WHILE CHE MI DETERMINA IL NUMERO DI TENTATIVI
DELL'INSERIMENTO DELLA PASSWORD
          while ( !thisCard.pin.equals (pin) & attempts < 4 )

          { pin = KeyPad.readPin () ; attempts = attempts + 1 ; }

// SE IL VALORE è DIVERSO DALLA CARTA PIN ALLORA LANCIA
UN'ECCEZIONE

          if (!thisCard.pin.equals (pin))

             throw new InvalidCard ("Bad PIN");

// MEMORIZZO ALL'INTERNO DI QUESTA VARIABILE IL SALDO DELLA CARTA
INSERITA
         thisBalance = thisCard.getBalance () ;

// AVVIO UN CICLO DO WHILE DOVE SI VISUALIZZA UN PROMPT POSTO
ALL'UTENTE CHE GLI CHIEDE DI SELEZIONARE I SERVIZI ELENCATI

          do {  Screen.prompt (" Please select a service ") ;
             service = Screen.touchKey () ;
             switch (service) {
               case Services.withdrawalWithReceipt:
                             receiptRequired = true ;
```

Abbiamo specificato attraverso un linguaggio pseudocodice, in modo compatto e preciso, il funzionamento della macchina.
A questo punto si ha il rischio di avvicinarsi all'implementazione, ma a noi serve specificare in dettaglio cosa deve fare il prodotto software, non come farlo. Questo è un problema che si osserva nei documenti di specifica.

L'uso del pseudocodice è consigliato quando un'operazione è specificata come una sequenza di azioni semplici. L'utilizzo più consigliato è specificare un'interfaccia come quella che segue.

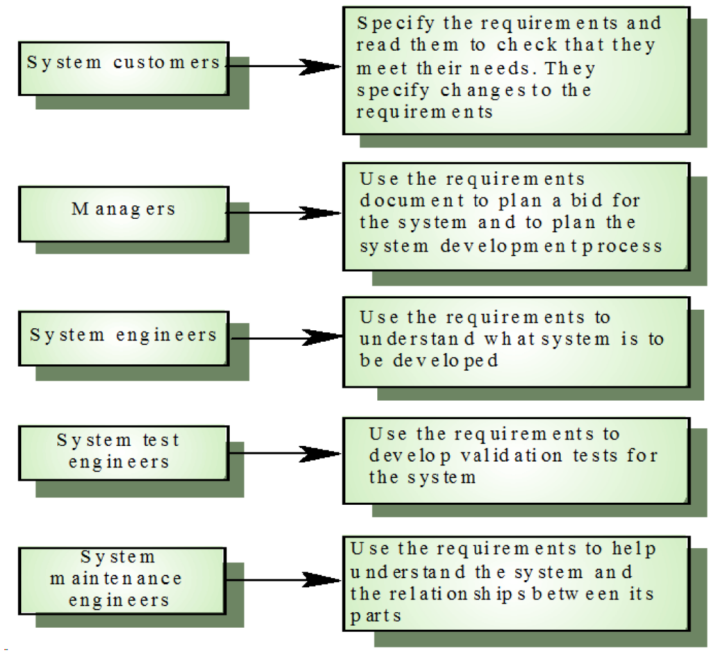
Esempio basato su interfaccia basata su PDL:

```
interface PrintServer {

// Definisce un server di stampa astratto

// Richiede: interfaccia Stampante, interfaccia PrintDoc

// Fornisce: inizializzare, stampare, visualizzarePrintQueue,
annullarePrintJob, switchPrinter

void initialize ( Printer p ) ;
void print ( Printer p, PrintDoc d ) ;
void displayPrintQueue ( Printer p ) ;
void cancelPrintJob (Printer p, PrintDoc d) ;
void switchPrinter (Printer p1, Printer p2, PrintDoc d) ;

} //PrintServer
```

Qui sto solo dichiarando delle funzioni, non sto definendo delle funzioni.

Tutto ciò si trova all'interno del documento di specifica. Questo è un 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).

Nella stesura di questo documento sono coinvolte i seguenti utenti con i loro rispettivi compiti:



Useremo dei template per scrivere il documento di specifica. Il seguente template è stato standardizzato dallo standard IEEE 830-1998 nel documento *IEEE Recommended Practice for Software Requirements Specifications.*

- *Prefazione:* lettura prevista, cronologia delle versioni, riepilogo delle modifiche
- *Scopo dell'introduzione:* breve descrizione del sistema, interazione con altri sistemi, ambito all'interno del contesto aziendale
- *Glossario:* definizione dei termini tecnici utilizzati nel documento
- *Definizione dei requisiti dell'utente:* requisiti utente funzionali e non funzionali
- *Architettura del sistema:* Panoramica di alto livello dei componenti del sistema
- *Specifiche dei requisiti di sistema:* requisiti di sistema funzionali e non funzionali
- *Modelli di sistema:* descrizione delle relazioni tra i componenti del sistema e il sistema e il suo ambiente
- *Ipotesi di evoluzione:* del sistema su cui si basa il sistema e modifiche previste (evoluzione dell'hardware, modifiche alle esigenze dell'utente, ecc.)
- *Appendici:* informazioni specifiche relative all'applicazione in fase di sviluppo (es. Descrizioni HW e DB)
- *Indici:* indice alfabetico, elenco di diagrammi, ecc.