- ***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.

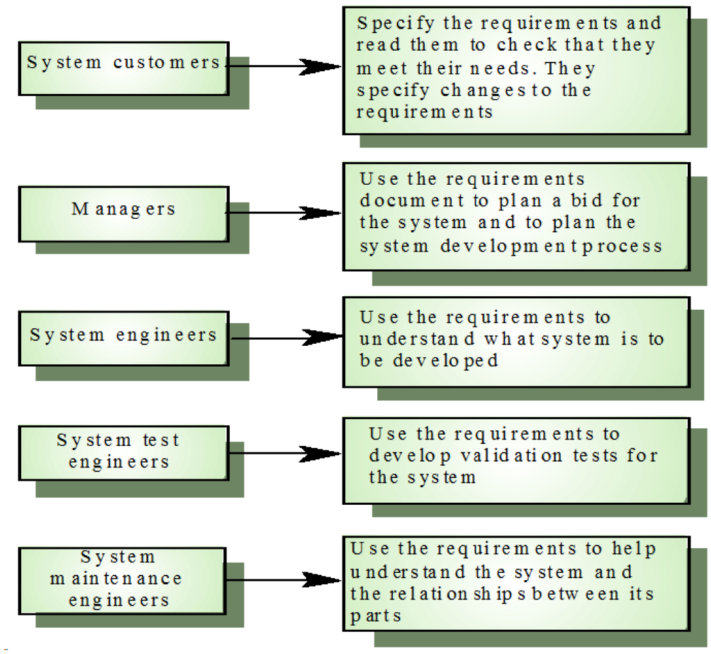
Esempio basato su interfaccia basata su PDL:

```
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.

→ *Quarto Blocco*

Il processo di ingegneria dei requisiti (*requirements engineering*) varia in base al dominio applicativo, alle persone coinvolte ed all'organizzazione che sviluppa il sistema software.

Si può però individuare un insieme di attività generiche comuni a tutti i processi. Seguono in ordine di esecuzione:

- studio di fattibilità (feasibility study)
- identificazione e analisi dei requisiti (requisiti elicitation and analysis)
- specifica dei requisiti (requisiti specification)
- convalida dei requisiti (requisiti validation)
- gestione dei requisiti (requisiti management)

Vediamo nel dettaglio le fasi elencate precedentemente:

- *Studio di fattibilità:* è una fase iniziale (o anche fase che precede i requisiti) del processo di ingegneria dei requisiti. Si basa su una descrizione sommaria del sistema software e delle necessità utente. Le informazioni necessarie per lo studio di fattibilità vengono raccolte da colloqui con:

- client manager
- ingegneri del software con esperienza nello specifico
- dominio applicativo
- esperti delle tecnologie da utilizzare
- utenti finali del sistema

Lo scopo di questi colloqui è rispondere ad una serie di domande, del tipo:

- In che termini il sistema software contribuisce al raggiungimento degli obiettivi strategici del cliente?
- Può il sistema software essere sviluppato usando le tecnologie correnti e rispettando i vincoli di durata e costo complessivo?
- Può il sistema software essere integrato con altri sistemi già in uso?

Lo studio di fattibilità produce come risultato un report/documento che stabilisce l'opportunità o meno di procedere allo sviluppo del sistema software.

- *Identificazione e analisi dei requisiti:* In questa è fondamentale interagire, infatti il team di sviluppo incontra il cliente e gli utenti finali al fine di identificare l'insieme dei requisiti utente, dalla cui analisi si generano i requisiti di sistema (specifiche). L'identificazione dei requisiti può coinvolgere personale che copre vari ruoli sia all'interno dell'organizzazione del cliente che in altre organizzazioni o tra gli utenti finali. A questo proposito viene introdotto il termine *stakeholder*, il quale viene usato

per identificare tutti coloro che hanno un interesse diretto o indiretto sui requisiti del sistema software da sviluppare.

Esistono dei task principali per fare questa identificazione e analisi:

- *Comprensione del dominio:* l'analista deve acquisire conoscenze sul dominio applicativo.

  Esempio: se il sistema software deve supportare il lavoro di un ufficio postale, l'analista deve comprendere il funzionamento di un ufficio postale)

- *Raccolta dei requisiti:* mediante interazione con gli stakeholder si identificano i requisiti utente.

- *Classificazione:* l'insieme dei requisiti raccolti viene suddiviso in sottoinsiemi coerenti di requisiti

- *Risoluzione dei conflitti:* eventuali contraddizioni e/o conflitti tra requisiti vanno identificati e risolti

- *Assegnazione delle priorità:* mediante interazione con gli stakeholder, ad ogni requisito o sottoinsiemi di requisiti va assegnata una classe di priorità

- *Verifica dei requisiti:* i requisiti vengono controllati per verificarne completezza e consistenza, in accordo a quanto richiesto dagli stakeholder

Per quanto riguarda le tecniche di identificazione dei requisiti sono 3:

- Ethnography, basata sull'osservazione: quando l'utente non è in grado o non può farlo, quello che fa l'analista software è osservare e descrivere i requisiti
- Casi d'uso o basati su scenari: definire dei possibili scenari e a partire da questi l'analista software dovrebbe capire quali sono i requisiti.
- Prototipazione (già vista precedentemente)

Esistono pure tecniche di analisi (e specifica) dei requisiti, ovvero:

- Tecniche semi-formali, basate su modelli del sistema e usate dai metodi di analisi strutturata o analisi orientata agli oggetti [vedremo in dettaglio più avanti]
- Tecniche formali, basate su Petri Net, FSM, Z, etc.

- *Convalida dei requisiti*

  È finalizzata ad accertare se il documento dei requisiti, ottenuto come risultato della fase di analisi, descrive realmente il sistema software che il cliente si aspetta. La scoperta di errori in questa fase è fondamentale per evitare costosi rework in fasi più avanzate del ciclo di vita. I controlli da effettuare includono:

  - Validità
  - Consistenza
  - Completezza
  - Realizzabilità
  - Verificabilità

  Per la convalida dei requisiti esistono delle tecniche di convalida. Di seguito elencate dal più informale al più formale:

  - revisioni informali
  - revisioni formali: possono essere di vario tipo:
    - *walkthrough:* letteralmente "camminare dentro" il documento. Durante un meeting organizzato, i partecipanti ricevono in anticipo il documento da esaminare. Nel corso della riunione, una persona legge il documento ad alta voce e, in caso si riscontrino problemi, la lettura viene interrotta per discuterne insieme.
    - *ispezioni:* richiedono ruoli specifici tra cui un moderatore, uno sviluppatore, un progettista, un analista, un tester e uno "scribe" (segretario). Il processo segue passaggi formali: si inizia con un'analisi preliminare (overview), poi con la fase di preparazione durante la quale il documento viene distribuito ai partecipanti. Successivamente si svolge l'ispezione vera e propria, durante la quale si redige un report che raccoglie eventuali errori. Dopo aver apportato le correzioni, si procede con un "rewalk" per verificare la soluzione. È importante che la riunione di ispezione non superi le 3 ore, per evitare cali di concentrazione. Sebbene questa tecnica sia molto efficace, comporta costi elevati.
  - Prototipazione
  - generazione dei test-case
  - analisi di consistenza automatizzata (per requisiti formali)

Nel task di identificazione dei requisiti ad ogni requisito si associa un etichetta, in base alla probabilità che un requisito possa essere modificato sia in fase di sviluppo che durante la fase di uso e manutenzione. Infatti avremo:

- *requisiti stabili*, con probabilità minima di essere modificati nel tempo
- requisiti volatili, con probabilità elevata di essere modificati nel tempo. Possiamo elencare dei sottotipi che ti specificano quale può essere l'origine della modifica:

• mutabili: modifiche legate a cambiamenti nell'ambiente operativo

• emergenti: modifiche causate da una migliore comprensione del sistema software

• consequenziali: modifiche legate all'introduzione di sistemi informatici nel flusso di lavoro

• di compatibilità: modifiche legate a cambiamenti nei sistemi e nei processi aziendali

Ogni modica dei requisiti deve essere opportunamente modificata, attraverso:

- identificazione univoca dei requisiti
- gestione delle modifiche
    - analisi dei costi
    - analisi dell'impatto
    - analisi della realizzazione
- politiche di tracciabilità (relazioni tra requisiti e tra requisiti e progetto del sistema software)
- uso di tool CASE per il supporto alle modifiche

Qui riportiamo un diagramma che ci descrive il costo di specifiche formali e informali