INGEGNERIA DEL SOFTWARE

SPECIFICA DEI REQUISITI

Avvertenza: gli appunti si basano sul corso di Ingegneria del Software tenuto dal prof. Picco della facoltà di Ingegneria del Politecnico di Milano (che ringrazio per aver acconsentito alla pubblicazione). Essendo stati integrati da me con appunti presi a lezione, il suddetto docente non ha alcuna responsabilità su eventuali errori, che vi sarei grato mi segnalaste in modo da poterli correggere.

e-mail: webmaster@morpheusweb.it
web: http://www.morpheusweb.it

SPECIFICA DEI REQUISITI

Il termine "specifica" viene spesso utilizzato per indicare il risultato dell'attività di analisi (e "specifica", appunto) dei requisiti. Tale fase ha come scopo la raccolta, organizzazione e razionalizzazione dei requisiti dell'utente o, in altri termini, l'esplorazione dello spazio del problema. Bisogna capire il contesto, le condizioni d'uso ecc…

DISTINZIONE TRA PROBLEMA E SOLUZIONE

Prima di procedere alla soluzione bisogna capire qual è il problema.

- I **REQUISITI** si occupano del problema (WHAT)
- **DESIGN** ed **IMPLEMENTAZIONE** trattano lo spazio della soluzione (HOW)

REQUISITI

I requisiti per un manufatto sono, in generale, le *proprietà richieste,* ed in particolare, gli effetti che la macchina deve produrre sul dominio del problema.

Il processo di ingegnerizzazione serve a colmare il gap tra i requisiti ed i materiali grezzi collegando tutto ciò che è disponibile (linguaggi, componenti…) ed assemblandoli in modo da soddisfare i requisiti

LO SPAZIO DEL PROBLEMA

Occorre capire qual è il dominio applicativo.

IL PROCESSO DI REQUIREMENTS

C'è il problema di capire cosa deve fare il sistema a quali sono i vincoli al contorno.

Occorre un interazione con:

- Clienti
- Utenti
- Esperti del dominio

Occorre identificare gli ***stakeholders*** (azionisti); sono gli attori fondamentali, le persone rilevanti per le applicazioni.

WHAT VS HOW

La distinzione è imprecisa.

WHAT cattura i requisiti funzionali; non specifico il come
Ci sono anche quelli non-funzionali

REQUISITI NON-FUNZIONALI

- Look and Feel (aspetto)
- Usabilità
- Performace
- Manutenibilità
- Portabilità (vincoli sulla piattaforma)
- Sicurezza
- Legali (ad esempio le norme sulla privacy)
- Culturali e politici

REQUISITI FUNZIONALI

Componenti principali dei requisiti funzionali sono:

- Il dominio applicativo
- Il problema da risolvere
- La macchina da costruire (la soluzione software)

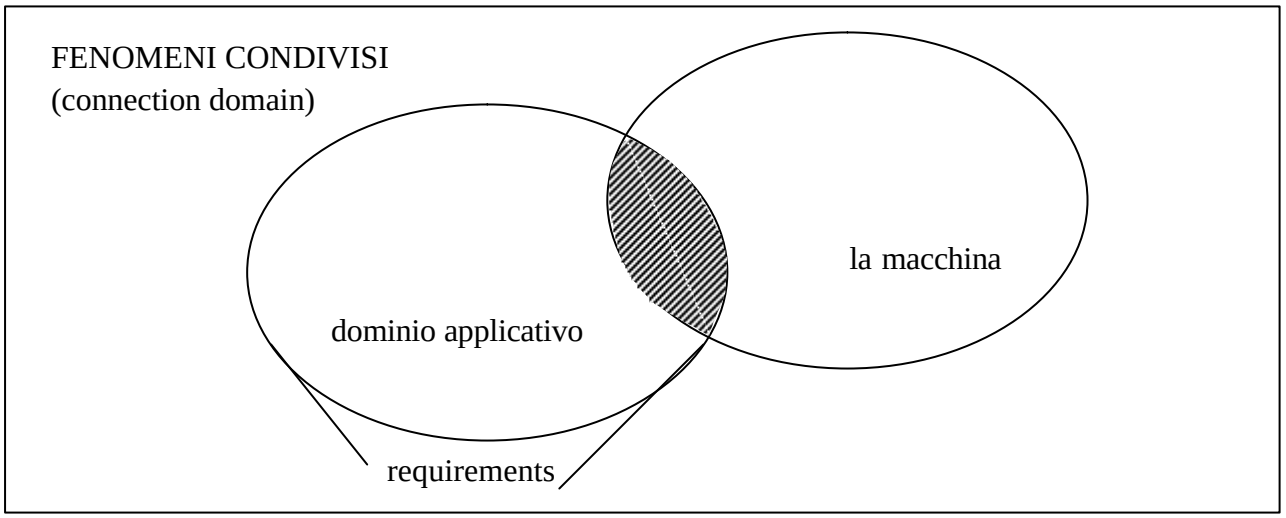
IL DOMINIO APPLICATIVO E LA MACCHINA

DOMINIO APPLICATIVO:
Un set di fenomeni e proprietà osservabili

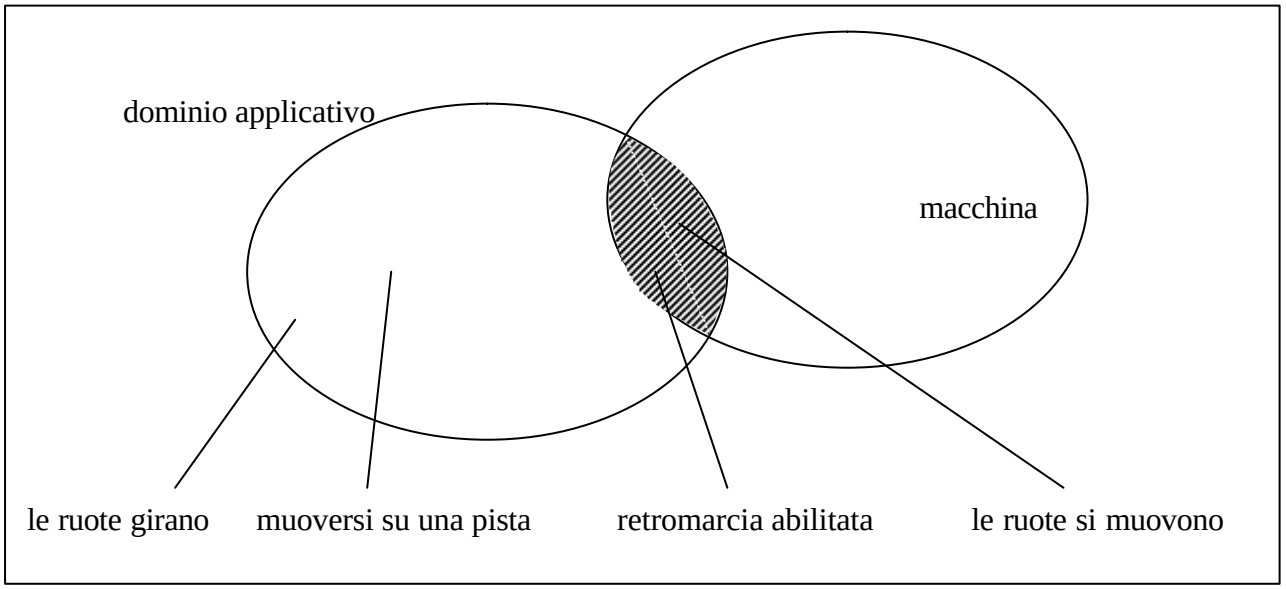
MACCHINA:
set di fenomeni di implementazione e proprietà

alcuni fenomeni sono CONDIVISI, sono l'interfaccia tra la macchina ed il dominio

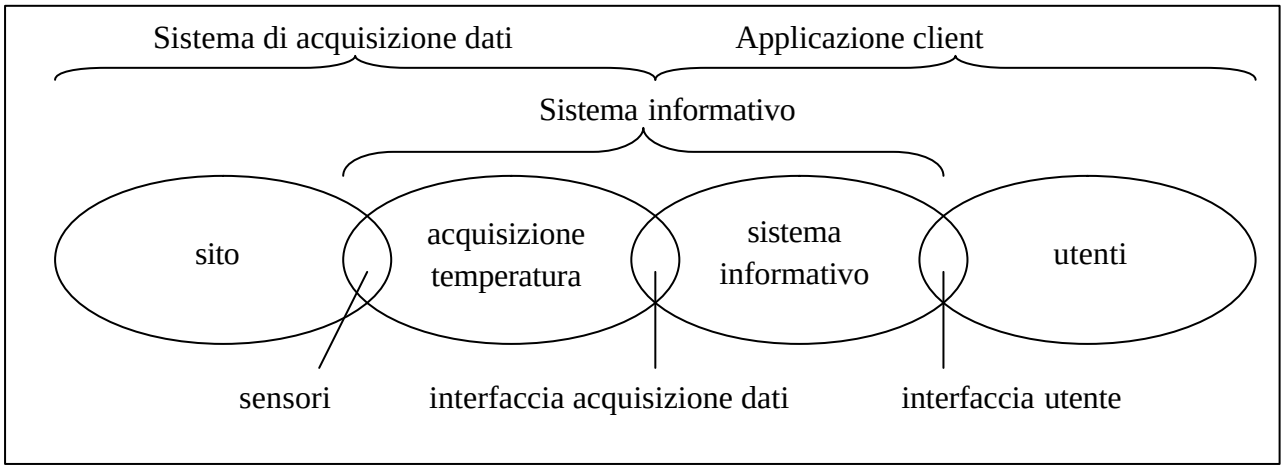
UNA VISIONE ASTRATTA



ESEMPIO



Se il sistema è complesso posso avere più domini di connessione



IL GOAL

Dare alle persone del mondo reale la possibilità di fare qualcosa che non potrebbero fare senza; capire cosa vuole il committente, le caratteristiche della macchina software che deve essere progettata ed implementata.

La macchina interagisce con il mondo reale per raggiungere l'obiettivo (in alcuni casi l'obiettivo della macchina è interagire con una macchina esistente)

CONTEXT DIAGRAM

E' come descriviamo i requisiti, i context diagram cercano di catturare la visione di ciò che deve fare la macchina ad un livello di astrazione abbastanza alto.

- Descrivono la distinzione tra la macchina e l'ambiente.
- Descrivono tutti i domini coinvolti, rilevanti ai requisiti del problema.
- Le connessioni mostrano l'esistenza di fenomeni condivisi

ESEMPIO: MONITORAGGIO DEI PAZIENTI

I pazienti in un padiglione di cura intensiva, sono monitorati da strumenti elettronici collegati ai loro corpi mediante sensori di vario tipo.
I sensori degli strumenti misurano i fattori vitali dei pazienti: le pulsazioni, la pressione, e così via.
E' necessario un programma che legga i valori alla frequenza specificata per ogni paziente e memorizzi i dati su un database.
I fattori letti devono essere comparati con dei livelli di sicurezza specificati per ciascun paziente, e le letture che superano tali valori devono essere segnalate mediante messaggi di allarme che vengono mostrati sullo schermo nella stanza degli infermieri.
Un messaggio di allarme viene anche mostrato se uno qualsiasi dei dispositivi analogici si guasta.

CONTEXT DIAGRAM

In prima analisi il software interagisce con il centro di cura intensiva

Non da molte informazioni

DOMINI DI INTERESSE

Raffiniamo il diagramma. I domini di interesse si evincono analizzando il testo. Ci sono delle fresi che coinvolgono degli attori:

- pazienti in un padiglione di cura intensiva, sono monitorati
- messaggi di allarme che vengono mostrati sullo schermo nella stanza degli infermieri
- allarme viene anche mostrato se uno qualsiasi dei dispositivi analogici si guasta

Da cui comprendiamo i domini di interesse

- I pazienti
- La stanza degli infermieri
- I dispositivi analogici
  perché i requisiti degli utenti sono espressi in termini di dispositivi

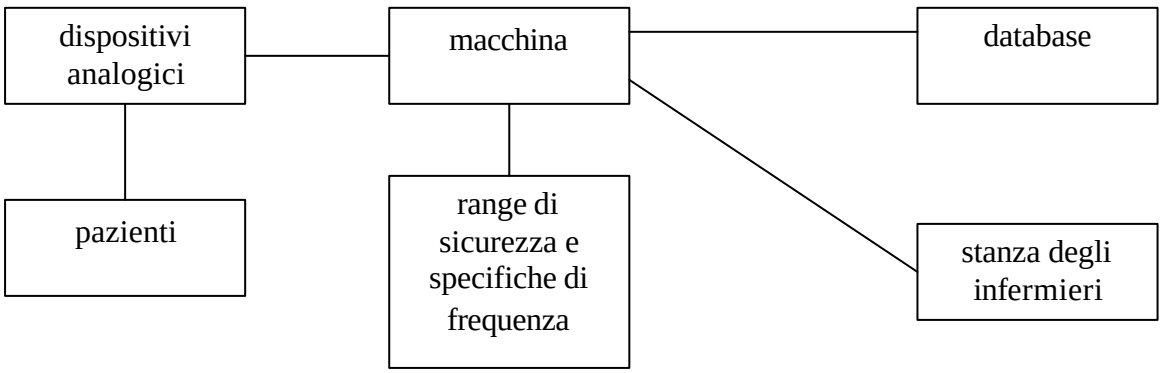
PRINCIPIO

Il principio della rilevanza del dominio dice che: tutto quello che è rilevante per i requisiti deve comparire da qualche parte del dominio applicativo.

DOMINI

- Cosa sono le entità del dominio?
- Che tipo di attributi/proprietà hanno?
- Che tipo di relazioni esistono tra le entità?
- Che tipo di eventi possono accedere?

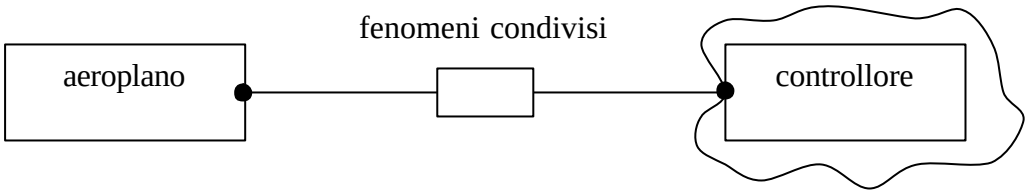
CONTEXT DIAGRAM DETTAGLIATO



I pazienti non hanno collegamenti diretti con la macchina, ma devono essere ineriti per il principio di rilevanza in quanto vincolano il resto del sistema.

CONTEXT DIAGRAM: CONTENIMENTO

Per domini applicativi che non sono disgiunti (ad esempio per fenomeni condivisi)



Il dominio di connessione lo rappresentiamo come un dominio a se. ● indica il fatto che il dominio è contenuto sia nel dominio applicativo (aeroplano) che nella macchina (controllore)

I fenomeni osservabili soddisfano alcune leggi che dipendono dal dominio.

Ad esempio:
ruote_girano ⇔ *ci_si_muove_in_pista*
ruote_si_muovono ⇔ *ruote_girano*

REQUISITI FUNZIONALI

Dopo aver identificato i domini (con il diagramma di contesto), occorre derivare i requisiti funzionali che devono definire le proprietà della macchina mediante predicati che possono coinvolgere sia fenomeni condivisi che non. In pratica:

- Il problema da risolvere va descritto in termini di dominio
- Bisognerà specificare le proprietà che saranno realizzate dal software
- Possono essere formalizzati come predicati su fenomeni osservabili nel dominio applicativo
- I predicati possono coinvolgere sia fenomeni condivisi che non

REQUISITI VS SPECIFICA

Specifica → Rappresenta i vincoli sui fenomeni condivisi tra macchina ed ambiente, come deve comportarsi la macchina. Devono essere soddisfatti dal software. Non dicono come la macchina dovrà ottenere certi risultati, ma ciò che deve essere fatto in termini di segnali gestibili dalla macchina

Requisiti → Proprietà che vogliamo ottenere nel mondo applicativo

Dai requisiti bisogna derivare la specifica.

SPECIFICA DEI REQUISITI

Sono un particolare tipo di requisito che trattano soltanto i fenomeni condivisi

I fenomeni vincolati dalla specifica sono quelli sotto il controllo della macchina da costruire

> ***Esempio***
>
> *la direzione di marcia può essere invertita Û le ruote sono in movimento*
> *(impedisce di andare in retromarcia se l'aereo è in volo).*

L'ATTIVITA DI SPECIFICA DEI REQUISITI

Si compone di tre parti

G → E' il problema da risolvere, le proprietà che voglio ottenere attraverso il software (requisiti)

S → Specifica, descrive l'interfaccia tra dominio applicativo e macchina

D → Dominio applicativo (insieme dei fenomeni che ho nel mondo reale e proprietà)

$\boxed{D \cap S \Rightarrow G}$

La congiunzione delle proprietà del dominio con le specifiche devono implicare i requisiti

TERMINOLOGIA

Non è stata stabilita ed accettata una terminologia, requisiti, specifica e design sono termini usati con significati diversi, spesso perché usati in momenti diversi dello sviluppo.

TERMINOLOGIA COMUNE

- REQUISITO per un "servizio" fornito da X espresso da Y, il quale richiede X
  Proprietà che esprime l'aspettativa di Y sull'abilità di X
- SPECIFICA: contratto tra X che fornisce il servizio ed Y che ne usufruisce
- PROGETTO: la struttura che fornisce il servizio

NEL CONTESTO SPECIFICO

- REQUISITO: gli effetti richiesti da un'applicazione sull'ambiente esterno
- SPECIFICA: vincoli sull'*interfaccia* tra l'ambiente e la macchina
- REQUIREMENTS ENGINEERING: come progettare requisiti e specifiche

ALCUNI CONCETTI

- **Descrizioni del dominio**
  - Titoli (nomi): concetti primitivi, fenomeni di interesse
  - Definizioni: sono delle convenzioni, definiscono termini nuovi usando termini già definiti
  - Proprietà: esprimono la nostra conoscenza della realtà in un modo indicativo
- **Requisiti** e **Specifiche**
  - Esprimono dei *desideri*

Su questi concetti si basa la metodologia di gestione dei requisiti

DESCRIZIONE DEL DOMINIO

Abbiamo tre concetti:

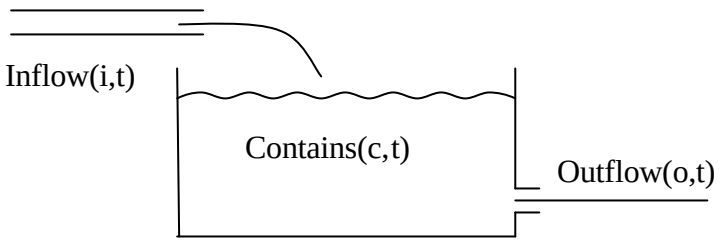
- DESIGNATIONS (designazioni)
- DEFINITIONS (definizioni)
- PROPERTIES (proprietà)

DESIGNATIONS: dobbiamo identificare i concetti primitivi, i fenomeni di interesse. Non vanno confusi con de definizioni.
Esempi: le_ruote_girano, uomo(x)…

DEFINITIONS: sono una comodità per definire nuovi termini a partire da quelli designati
Esempio: $brother(x,y) ::= male(x) \wedge \exists f \mid father(f,x) \wedge father(f,y)$

PROPERTIES: sono un modo per dare una definizione del dominio. Esprimono in modo indicativo la conoscenza della realtà (a prescindere da ciò che si vorrebbe, rappresentato dalle proprietà optative). Sono ciò che esiste, non ne abbiamo il controllo dal punto di vista della macchina.
Sono *confutabili*, se stendiamo durante la fase di analisi e si possono anche commettere errori.
Esempio: $\forall x, y(human(x) \wedge mother(x,y)) \rightarrow (female(x) \wedge human(y))$

ESEMPIO 1

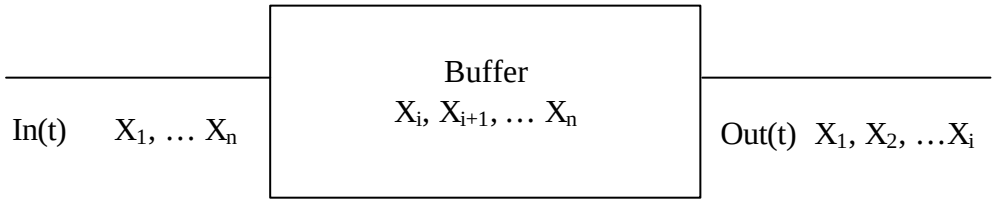


$\forall i,o,c,t Inflow(i,t) \wedge Outflow(o,t) \wedge Contains(c,t) \rightarrow i = c + o$
(su questa proprietà non abbiamo controllo)

Le specifiche utilizzano solo fenomeni condivisi, danno proprietà di tipo optativo poiché rappresentano i nostri desideri. Si possono riconoscere dal documento testuale perché esprimono degli **obblighi**. (E' necessario un programma che legga i valori alla frequenza specificata per ogni paziente e memorizzi i dati su un database. etc…)

ESEMPIO 2

Consideriamo un buffer:



$In(t) = Out(t) catBuffer(t)$

E' una specifica.

CORRETTEZZA DELLA SPECIFICA DEI REQUISITI

Possiamo aver sbagliato a modellare gli obiettivi (sbagliando a definire G)

La descrizione del dominio D può non riflettere la realtà (non abbiamo tenuto conto di fenomeni o lo abbiamo fatto in maniera sbagliata

In questi casi non è più vero che possiamo derivare G da D ed S

Esempio:

D
ruote_girano ⇔ *ci_si_muove_in_pista*
ruote_vibrano ⇔ *ruote_girano*

S
retromarcia_inserita ⇔ *ruote_girano*

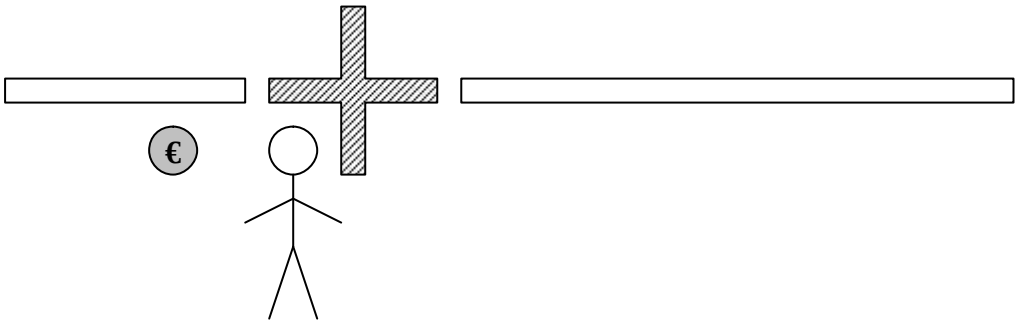
G
retromarcia_inserita⇔ ci_si_muove_in_pista

Il problema è nell'analisi del dominio" ruote_girano ⇔ ci_si_muove_in_pista",
non è corretto, infatti può accadere che: "ci_si_muove_in_pista ∧¬ ruote_girano"

E' l'acquaplanning; un caso di incidente in cui un aereo plana sull'acqua

ESEMPIO DI ANALISI DEI REQUISITI

Consideriamo un sistema per il controllo di un tornello per l'ingresso ad uno zoo.



La persona inserisce una moneta che abilita il tornello.
Appena la persona è entrata, il tornello è bloccato fino all'inserimento di un'altra persona.
Vogliamo progettare il controllore software che blocca e sblocca il tornello a seconda degli eventi esterni.

ANALISI DEL DOMINIO

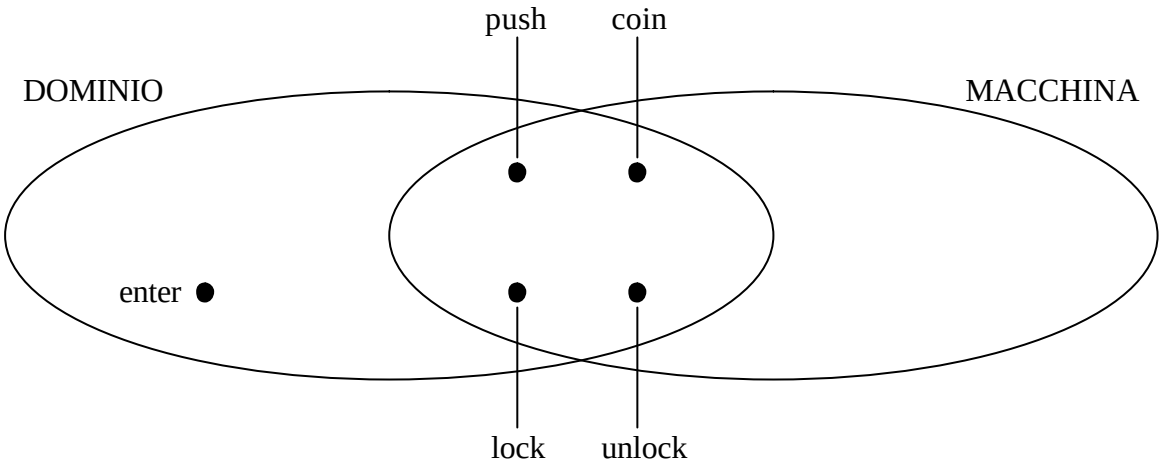
Il primo passo è analizzare i fenomeni rilevanti (rispetto all'obiettivo)

Vogliamo identificare le DESIGNAZIONI

- coin(e) → la persona deve inserire la moneta
- push(e) → evento di pressione della persona contro il tornello
- enter(e) → il tornello viene spinto del tutto
- lock(e) → blocco del tornello
- unlock(e) → sblocco del tornello

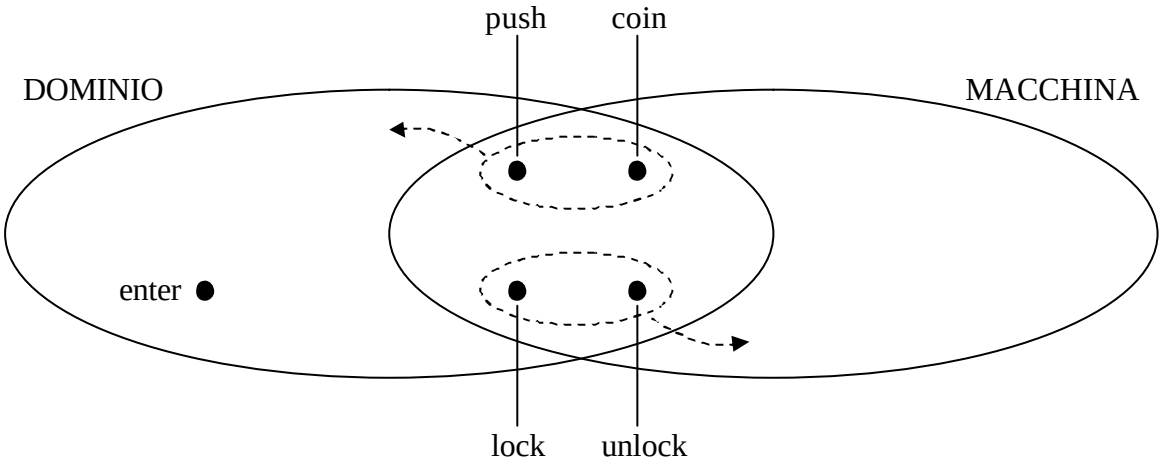
è utile anche capire qual è la natura di queste designazioni (se sono fenomeni condivisi, dell'ambiente o solo della macchina, questi ultimi non ci interessano in questa fase)

CLASSIFICHIAMO I FENOMENI



push e coin sono condivisi (nascono nell'ambiente reale ma sono gestiti dal controllore) come pure lock ed unlock (permettono alla macchina di modificare il comportamento dell'ambiente).
enter non è invece modificabile direttamente dalla macchina.

Occorre anche identificare dove sta il controllo (chi può controllare gli eventi).
push e coin sono controllati dall'ambiente, lock ed unlock dalla macchina.



Assumiamo che gli eventi siano di tipo istantaneo. Se ci fosse stato un ritardo avremmo dovuto modellarlo)

DESCRIZIONI INDICATIVE ED OPTATIVE

Indicative → fenomeni che esistono nell'ambiente indipendentemente dalla macchina (possono anche essere sbagliate

Optative → desideri, ciò che vogliamo ottenere dalla macchina

DESCRIZIONI INDICATIVE

Abbiamo due proprietà:

IND1 → push ed enter si alternano (rende conto del fatto che non è possibile passare senza spingere il tornello e un'altra persona non può spingere finché una non è entrata)

IND2 → se gli eventi di lock ed unlock si alternano, e in particolare, unlock è il primo ⇒ l'evento di pressione del tornello può avvenire solo dopo l'unlock e prima del lock

Sono proprietà del mondo reale; sono D

DESCRIZIONI OPTATIVE (requisiti G**)**

OPT1 → in qualsiasi istante il numero di ingressi non deve essere maggiore del numero dei pagamenti effettuati fino a quel momento, assumendo per semplicità che serva una sola moneta per un ingresso. (è un requisito debole, permette gli ingressi di gruppo, posso ad esempio mettere 10 monete e poi far passare 10 persone)

OPT2 → quelli che hanno pagato non sono ostacolati ad entrare (dalla macchina). Di fatto presa alla lettera non è applicabile (può accadere qualcosa per cui chi ha pagato può non entrare, ad esempio se ha un malore o altro…)

DERIVARE LE SPECIFICHE DAI REQUISITI

Abbiamo D e G dobbiamo derivare **S** (le specifiche) in modo da avere solo i fenomeni condivisi che possono essere usati per implementare la macchina.

R1 → può essere reso vero controllando gli ingressi o le monete. Possiamo vincolare in modo indiretto l'ingresso delle persone bloccando e sbloccando il tornello.

IND3 → possiamo ricavarla da IND1 per raffinamento
in t se abbiamo e ingressi e p push $\Rightarrow p\text{-}1 \leq e \leq p$

OPT1$_a$ → derivabile da IND3 ed OPT1:
$\forall t$ se ho p push e c coin $\Rightarrow p \leq c$

Come ci assicuriamo che $p \leq c$? Se $p=c$, non devono essere permesse ulteriori spinte nel tornello (bloccandolo). Occorre agire su lock ed unlock; vediamo in che modo:

4) imponiamo lock ed unlock alternati

5) Raffiniamo la OPT1$_a$
se bloccato e $p=c$ non deve sbloccarsi
se sbloccato e $p=c$ deve bloccarsi in tempo per impedire un ulteriore evento di push

R2 → non ostacolare quelli che hanno pagato

6) raffiniamo la OPT2
se il tornello è sbloccato e $p<c$ la macchina non lo blocca
se è bloccato, lo sblocca

La specifica è data da {4, 5, 6}

Occorre fare in modo che S = {4, 5, 6} implichino i requisiti

RIASSUNTO

Dove troviamo i requisiti?

Nel dominio applicativo

Come troviamo i requisiti?

Dal testo, interagendo con gli stakeholders e comprendendo il dominio e le sue proprietà.

Dai requisiti alle specifiche

E' poi necessario passare alle specifiche, espresse in termini della porzione del dominio su cui la macchina può influire.

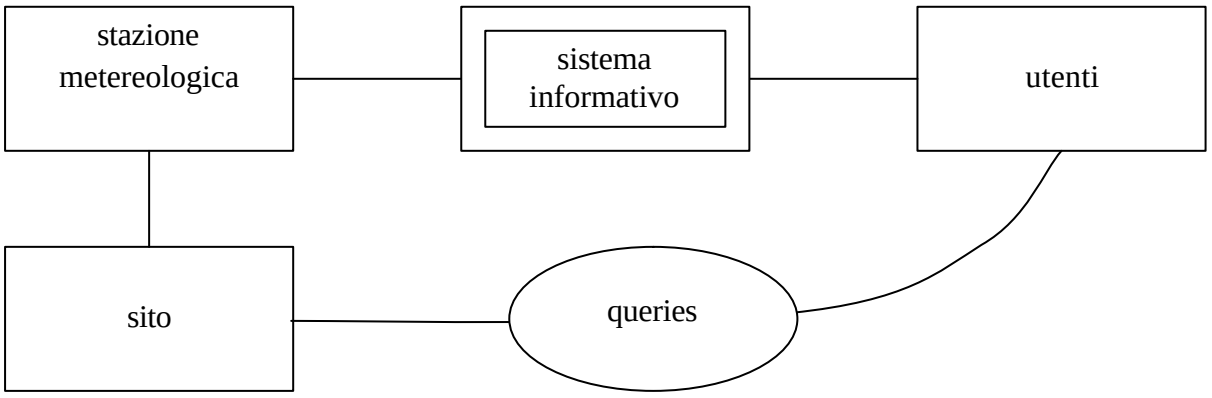
I requisiti vengono espressi in termini di fenomeni condivisi, e deve accadere che vengano verificati

FRAME DIAGRAMS

Una variante dei context diagrams viene realizzata aggiungendo ai domini un ovale che rappresenta le relazioni. Nascono così i frame diagrams che sono usati subito dopo i diagrammi di contesto.

ESEMPI

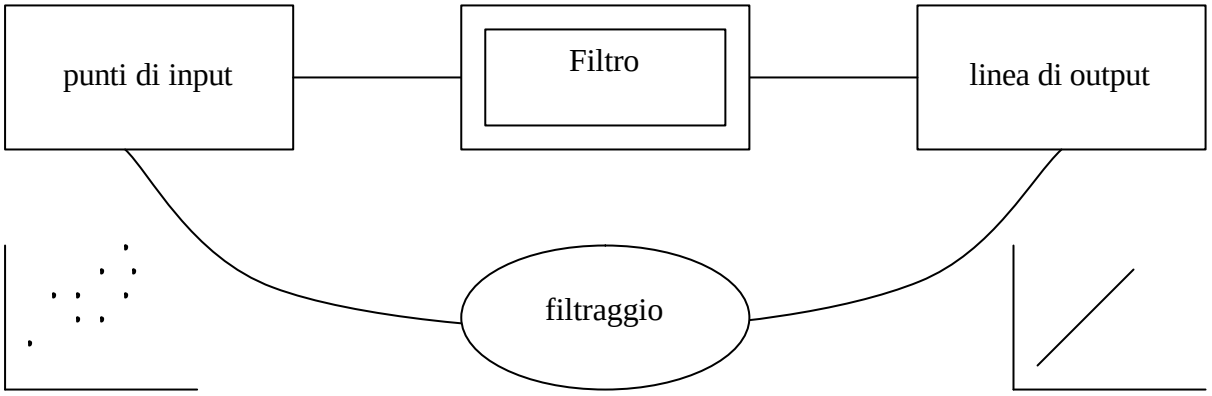
Si vuole consentire agli utenti di formulare interrogazioni sulle temperature in siti diversi nel mondo.



Il requisito fondamentale è permettere agli utenti di fare interrogazioni sui siti.

Sistema di filtraggio di immagini:

Input: immagine come insieme di punti Output: interpolazione dei punti

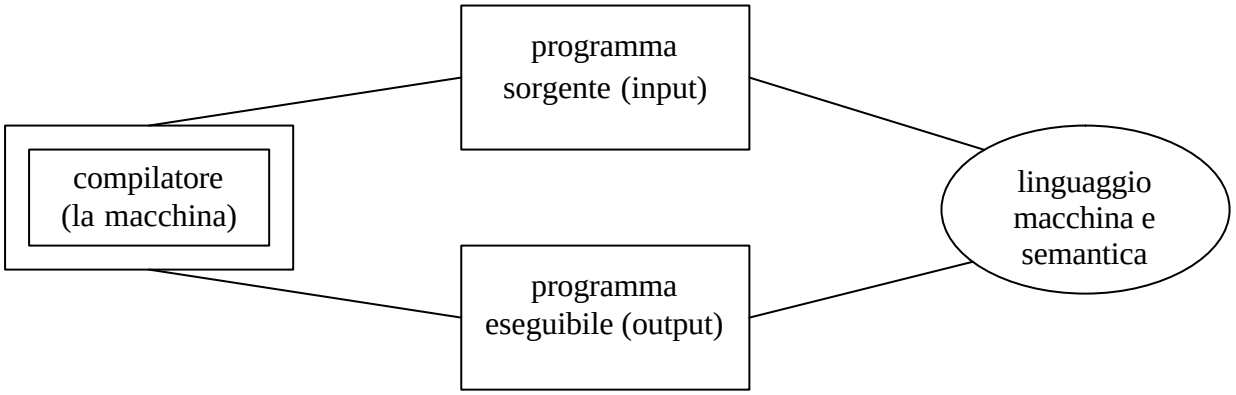


SPECIFICA IN MODO SISTEMATICO

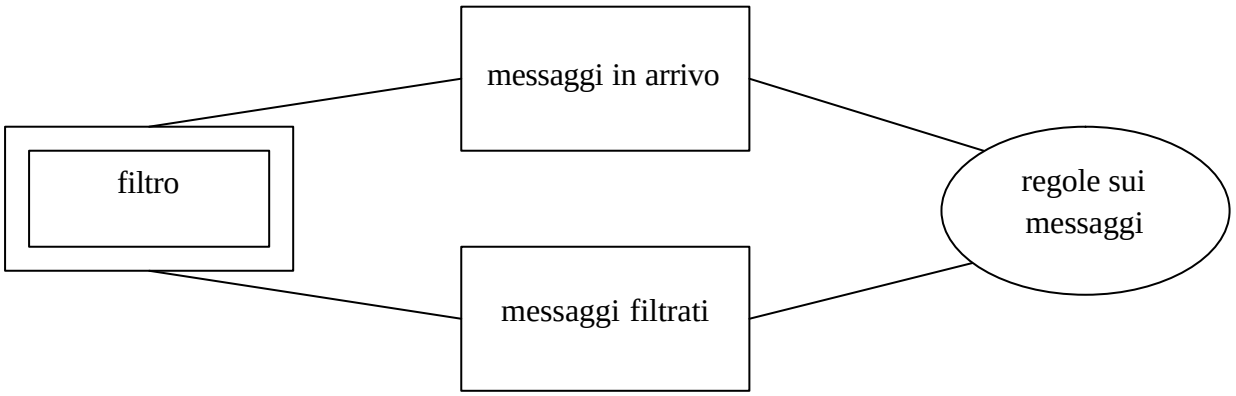
I frame diagram sono più espressivi dei context diagram, permettono inoltre di identificare degli schemi ricorrenti di problemi in cui la struttura è la stessa. In tal modo lo stesso schema è riusabile in contesti diversi.

COMPILER FRAME

Prende in input un programma sorgente, da in output un eseguibile, la trasformazione è fatta con delle regole derivate dal linguaggio.

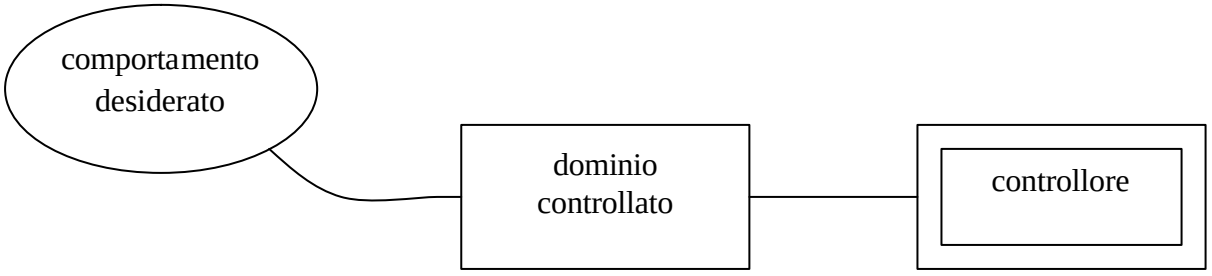


ESEMPIO: filtraggio posta elettronica

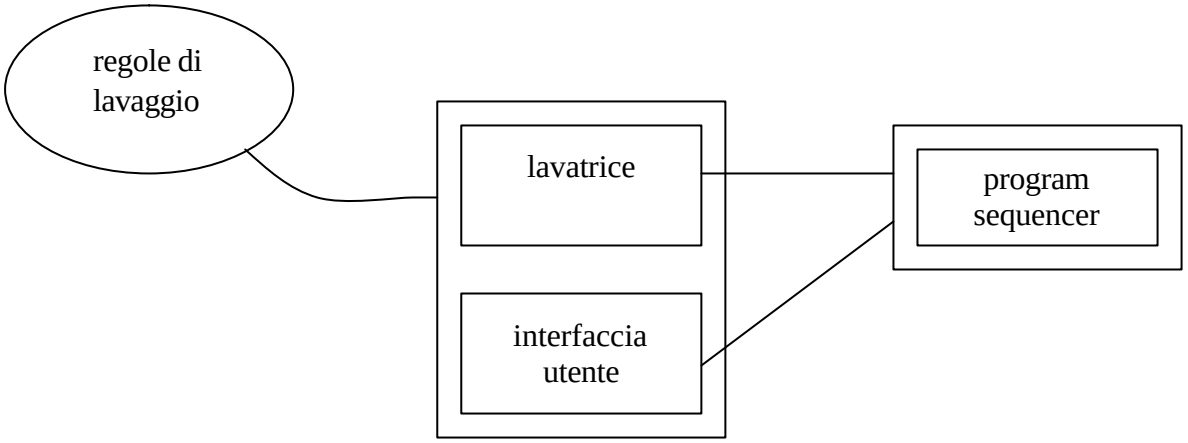


COTROL FRAME

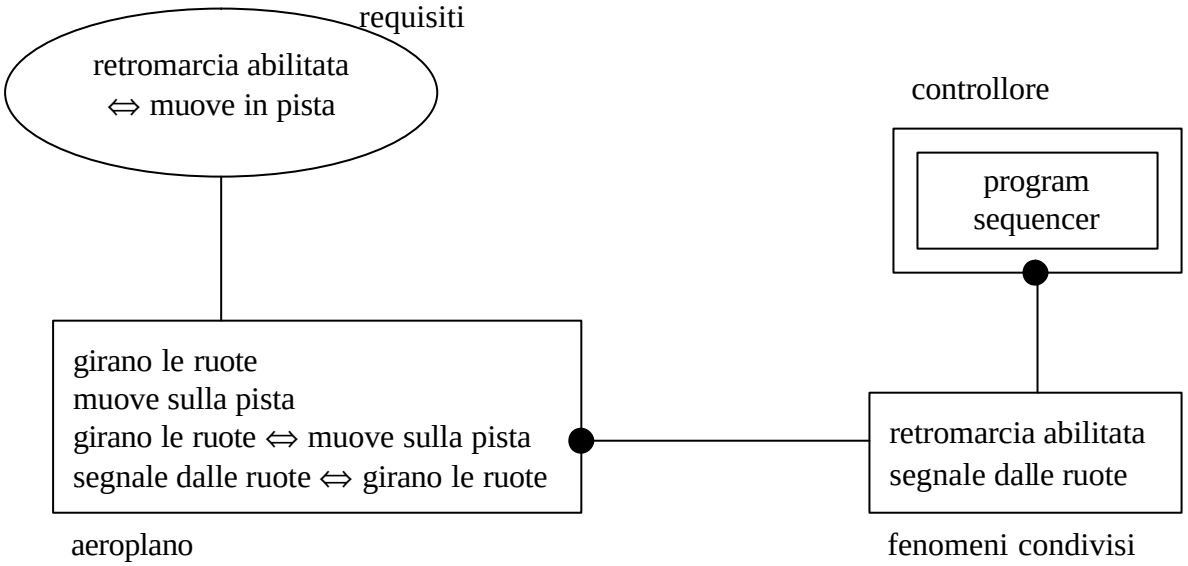
Non c'è notazione di trasformazioni I/O, c'è il dominio controllato, la macchina è il controllore e la relazione è solo sul dominio controllato (vogliamo far si che si comporti in un certo modo)



ESEMPIO: lavatrice

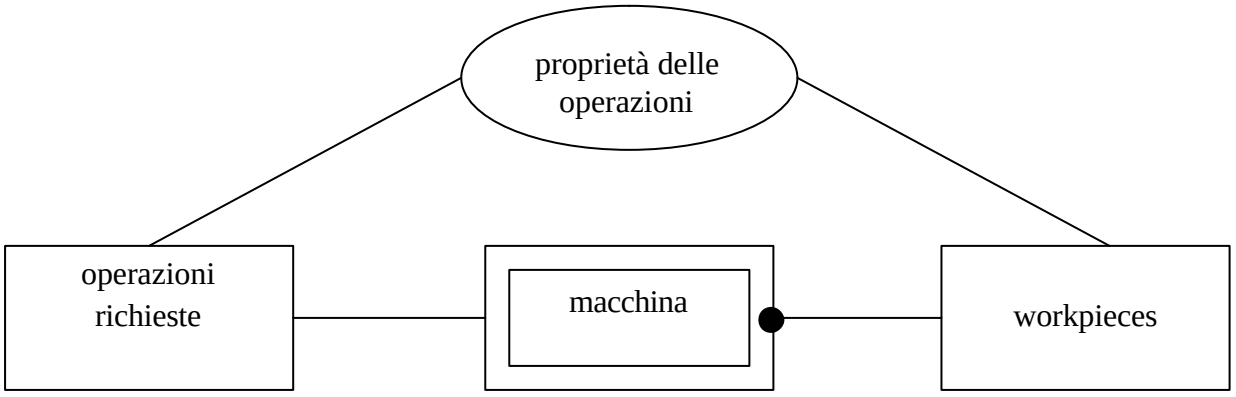


ESEMPIO: aeroplano



WORKPIECES FRAME

Detto workpieces perché la caratteristica è che uno dei domini al livello applicativo è contenuto nella macchina.

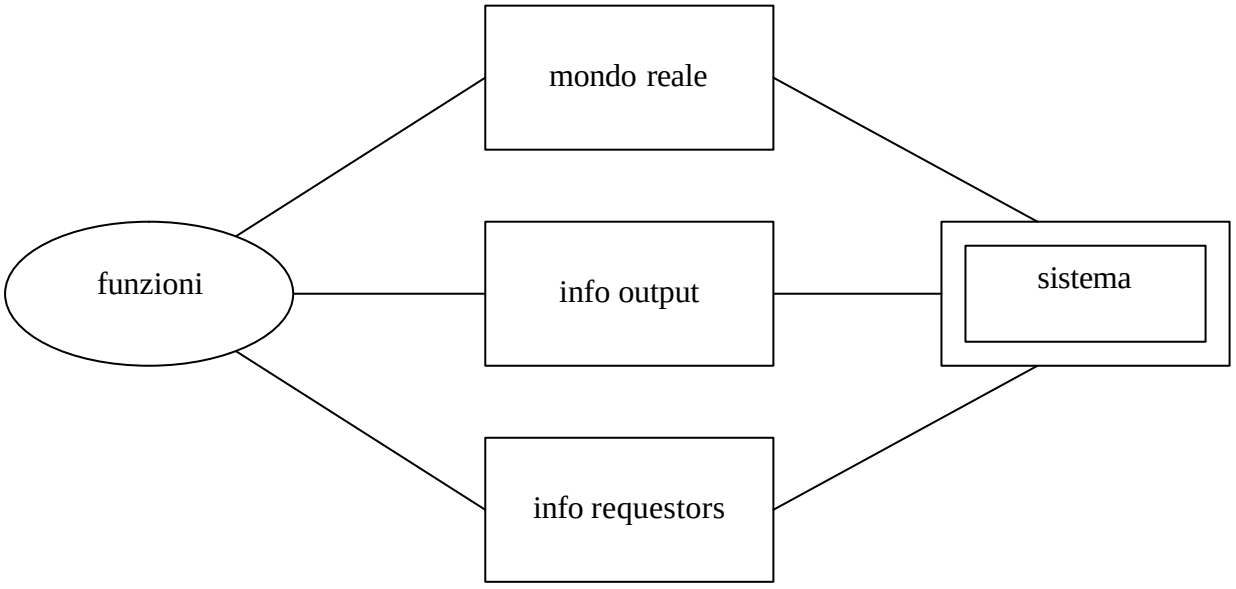


ESEMPIO: editor di testo

L'editor opera su un dominio interno alla macchina (sono gli oggetti gestiti dall'editor). Voglio distinguere la macchina dagli oggetti su cui opera, ma di fatto la parte risiede fisicamente sulla macchina.

INFORMATION SYSTEM FRAME

Rappresentato come una macchina che ha come dominio il mondo reale, ottiene richieste dagli utenti e produce informazioni in base a qualche funzione



PROBLEMI MULTIFRAME

I frame diagram sono utili perché se mi accorgo per tempo che posso modellare il problema con uno di questi problem frame noti, ho una linea di guida. In genere i problemi sono più complessi, usiamo dei multiframe che combinano insieme problem frame elementari.

FATTORI DI MERITO DEI REQUISITI

- verificabilità (voglio poter verificare che siano validi
- consistenza (non voglio avere conflitti)
- completezza (voglio aver incluso tutte le funzioni richieste dall'utente)
- realismo (i requisiti posso effettivamente implementarli, in termini di costi, tempi, tecnologia…?)
- verificabilità (devono essere testabili)
- comprensibilità
- tracciabilità (è una metrica interna, ci chiediamo se dato un particolare frammento possiamo risalire al perché è stato inserito li)
- adattabilità (qual è l'impatto di un determinato requisito sul sistema)