

COME SCRIVERE UN RASD

Alessio Bellon:

INDICE:

- L'ingegneria dei requisiti;
- Workflow
- Struttura del RASD
- UML
- esempi;
- riapporto finale

Requirements Engineering

Per lo scopo di definire per cosa il sistema è stato inteso.

→ requisiti specificati dal cliente possono essere divisi:

→ **FUNZIONALI**: i principali obiettivi del SW. Descrivono le interazioni fra il sistema e l'ambiente;

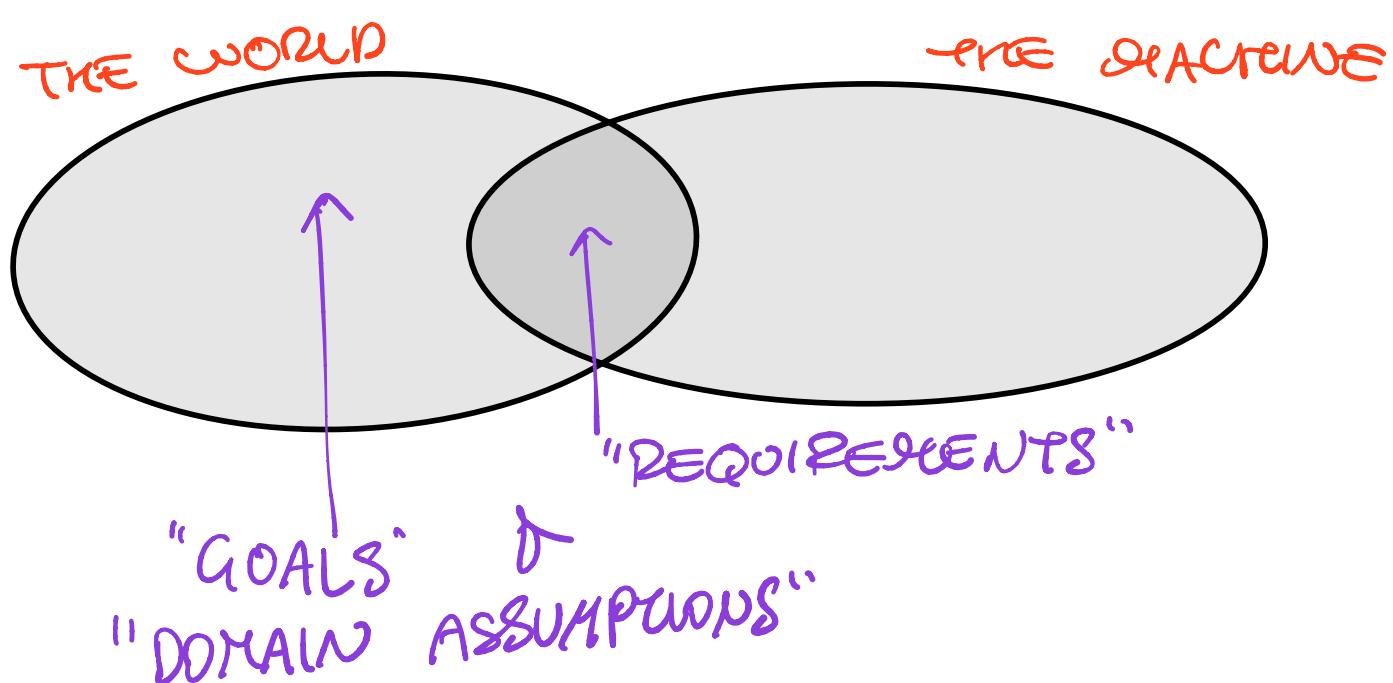
→ **NON FUNZIONALI**: specifiche in dettagli con strettamente legate alle particolarità (es. sicurezza, disponibilità...);

→ **VINCOLI** (o "REQUISITI TECNICI"):
requisiti su come implementare il sistema;

Spesso i requisiti non sono
specificati nello stesso linguaggio,
in particolare **NON** devono:

- essere vaghi o incompleti;
- essere contraddittori;
- man poter essere testati;
- mischiare aspetti differenti;
- riferirsi a responsabilità di terzi
oltre il cliente finale e non
il sistema;
- man essere fattibili.

THE WORLD AND THE DESIGN



GOALS: cosa il sistema deve fare in
termi dei fenomeni che
caratterizzano il mondo;

DOMAIN ASSUMPTION: cosa siamo per
scritto sul mondo

REQUIREMENTS: cosa il nostro sistema
deve permettere in
termi dei fenomeni
"conosciuti" tra il
mondo e il sistema

LA SPECIFICA DI QUESTE COSE
È L'UNICA DIFFERENZA DEL RASD

CARATTERIZZAZIONE DEI REQUISITI:

$$R \wedge D \vdash G$$

"Se qualcosa non soddisfa "G"
allora non deve soddisfare
neanche " $R \wedge D$ ".

RASD workflow

1. analisi dello specifico, divisione in requisiti funzionali, non funzionali e tecnici;
2. identificazione ^(*) degli scenario ^{e completi} dei fenomeni; ^(**) di sequenze di sequenze e attività di sequenze
3. da scenario a use-case e activity diagrams
4. modellizzazione di GOALS, REQUIREMENTS e DOMAIN ASSUMPTIONS sulla base di quanto delineato negli use-cases;
5. definizione dei requisiti non funzionali;
6. verifica e validazione dei requisiti con il log.

Possibilità:

1. definizione di fenomeni e scenario
 2. definizione di DOMAIN ASS.
 3. definizione di GOAL
 4. definizione di REQUIREMENTS
- class diagrams e state diagrams;
- use-case, sequence e activity diagrams.

STRATEGIA DEL RASD:

e c'è un
sistema e
esterno e
utile

OVERALL DESCRIPTION

- A. Product perspective: here we include scenarios and further details on the shared phenomena and a domain model (class diagrams and state diagrams)
- B. Product functions: here we include the most important requirements
- C. User characteristics: here we include anything that is relevant to clarify their needs
- D. Assumptions, dependencies and constraints: here we include domain assumptions

SPECIFIC REQUIREMENTS:

Here we include more details on all aspects in Section 2 if they can be useful for the development team.

A. External Interface Requirements

- A.1 User Interfaces
- A.2 Hardware Interfaces
- A.3 Software Interfaces
- A.4 Communication Interfaces

anche le
limitazioni: telecamere, di sicurezza,
category, qual:
functionalità a caso

(*)

B. Functional Requirements: Definition of use case diagrams, use cases and associated sequence/activity diagrams, and mapping on requirements

C. Performance Requirements

D. Design Constraints

- D.1 Standards compliance
- D.2 Hardware limitations
- D.3 Any other constraint

va bene anche
mettere delle
interfaccie
grafiche
("semplici")

E. Software System Attributes

- E.1 Reliability
- E.2 Availability
- E.3 Security
- E.4 Maintainability
- E.5 Portability

FORMAL ANALYSIS USING ALLOY:

This section should include a brief presentation of the main objectives driving the formal modeling activity, as well as a description of the model itself, what can be proved with it, and why what is proved is important given the problem at hand. To show the soundness and correctness of the model, this section can show some worlds obtained by running it, and/or the results of the checks performed on meaningful assertions.

STAFFE STRATEGICHE

→ USE-CASE DIAGRAMS

MODELLO IL
mondo

→ WORLD CLASS DIAGRAMS

IL MODELLO
DI ALCUNE STANZE

→ STATE DIAGRAMS

MODELLO CE INTEGRATO
IN UNO USE-CASE

→ SEQUENCE DIAGRAMS

DESCRIVE
(SENTA)

I PROCESSI
ASSEGNAME (nuo)

→ ACTIVITY DIAGRAMS

DYNAMIC MODELLING

IL RASD...

- deve essere un specifico che il sistema soddisfa i req. e spett. le cui regole;
- il linguaggio deve essere chiaro e chi facilmente comprensibile per gli stakeholders:
 - Clienti
 - Utenti
 - analisti
 - programatori
 - manager

E' importante specificare come il SW commuiderà con l'ambiente interno e SW esterno.

Sarebbe meglio anche aggiungere la bibliografia.

(x) POSSIBILE ORGANIZZAZIONE:

- user class 1
- requirement 1
- sub requirement 1.1.1.

Comunque il RASD deve essere:

- preciso
- pertinente: tutti i req. devono soddisfare un goal, nessuno scelto che non sia definito in un requirement (a quel l'impl. no).

- non obbligare mai vocaboli, nelle responsabilità, nelle procedure per verificare un requisito;
- tutto quello che si era già definito prima di usarlo;
- ogni requirement e assumption deve avere il suo CDAI

Bisogna anche evitare ciò che è verificato formalmente e cosa no.

E' anche utile numerare qualunque cosa, req., goals, use-cases ecc.
Sì an.

Si deve vedere l'associazione use-case - requirements (in qualunque verso sia)

Compilare anche una matrice gli tracciabilità (vedere su slide gli esempi).

Il RASD non è un
documento che mostra "la
salvezza" non è il "contratto
col cliente". La salvezza è
affrontata dal design.

UML

USE CASE DIAGRAM:



Sono le gerarchie in UML degli use-case. Alla fine tutti gli use-case saranno abbinati alle specifiche del sistema.

(Si vedano le slide per chi dettagliate)

WORLD CLASS DIAGRAM:

Alla fine è un class diagram, però non modelli le operazioni (se non ci interessano di più). Dalla specifica:

- i NOMI diventano entità
- i VENNI diventano associazioni

Alla fine è un CLASS-DIAGRAM di alto livello.

UML PER LA MODELLOTAZIONE DINAMICA

SEQUENCE DIAG.

STATE DIAG.

ACTIVITY DIAG.

Non è tattica

descritto dagli stessi / uno
secondo tipo oggetto
del diagramma

: interazioni tra
oggetti

: mod. stat. di un singolo
oggetto

: processi, flussi

a "granularity level"

+ analisi per i protocolli
rispetto all'attività

PERÒ, QUESTA È MODELLOTAZIONE
DEI REQUISITI! Quindi DOVREBBE
LIMITARSI SOLO A QUELLO CHE "DEVE"
ESSERE SICURAMENTE. È LA
DOCUMENTAZIONE DEL DOMINIO, NON
DELL'APPPLICATIVO!

Altro fine, bisogna controllare di
essere giunto a esaltare,
personi, esse-cole, geog., goals, DE.

1. COORDINAMENTO DI AMBULANZE

un'ambulanza dovrebbe arrivare entro 14 minuti dalla chiamata, la posizione di un'ambulanza è tracciata via GPS. Grazie a smartphone il personale di un ambulanza ha disponibilità.

PROBLEMI:

→ non si capisce quali possono essere i confini del sistema!

ELENCHIAMO I POSSIBILI FENOMENI:

→ WORLD PHENOMENA:

- si verifica un incidente,
- l'incidente viene notificato tramite leva di urgenza,
- le urgenze vengono codificate nel sistema di gestimento,
- viene disposta un'ambulanza,
- l'ambulanza arriva sul luogo dell'incidente.

→ MACHINE PHENOMENA:

- un "incident" object is created
- an "incident" object is updated
- the DB entry is updated
- il percorso + tempo è calcolato

anche se le ambulanze fanno parte del sistema è in sans solo, loro non sono "la macchina"!



: SHARED
PHENOMENA

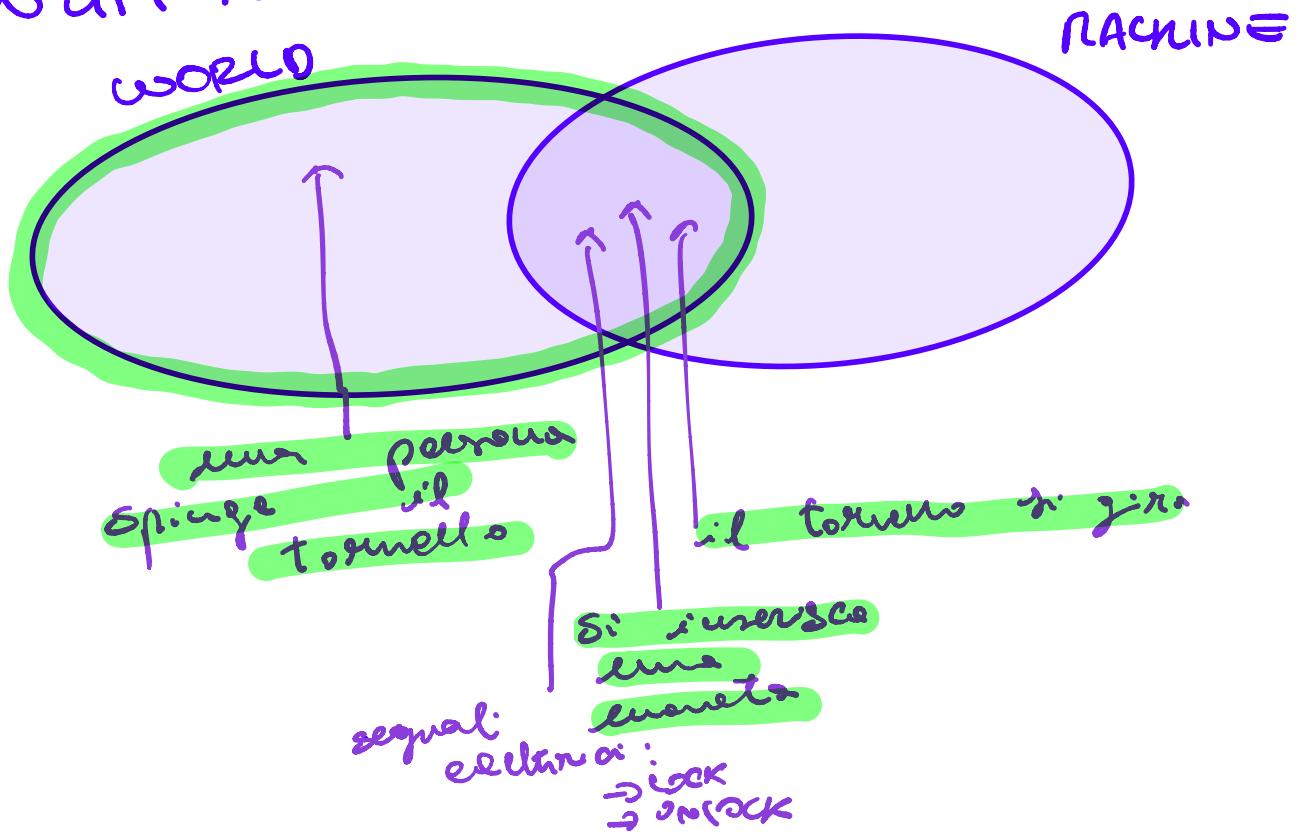
A questo punto, il "GOAL" sono ormai formule cui hanno dei "WORLD PHENOMENA", i requirements in termini dei "SHARED PHENOMENA".

Ad esempio:

- GOAL: per ogni incidente, un'ambulanza dovrà essere disponibile entro un certo istante;
- REQUIREMENTS: quando una ambulanza viene estesa, il software dovrà inviare un segnale secondo le informazioni disponibili sull'aggiornamento sulla posizione e la estetica di disponibilità.

2. IL CORNEO

—
je tomberai sur le secrétaire
pour penser de personnes <->
américaines et européennes
RENDEZ-VOUS | **FENOGREFF!**



DOMAIN ASSUMPTIONS:

- Push => Turns => Enter
 - Enter ≠> Turns
 - Turn => UNLOCK
 - la pressione "push" si esibisce leggermente dopo l'elisione "Turn"

solo =,
solo con
fuspare

GOAL: G_1

- GOAL: G1

 - 5) le entrate non devono essere superiori alle entrate ricevute
 - 6) il corretto non deve bloccare chi lavora

G2

REQUIREMENTS:

→ su "G1":

→ solo se ci sono eventi iniziali di "push" il touchlo deve dare l'unlock

→ se ci sono tante iniziative quindi push ad un solo touchlo deve bloccare la lock

→ finalmente il touchlo è lock

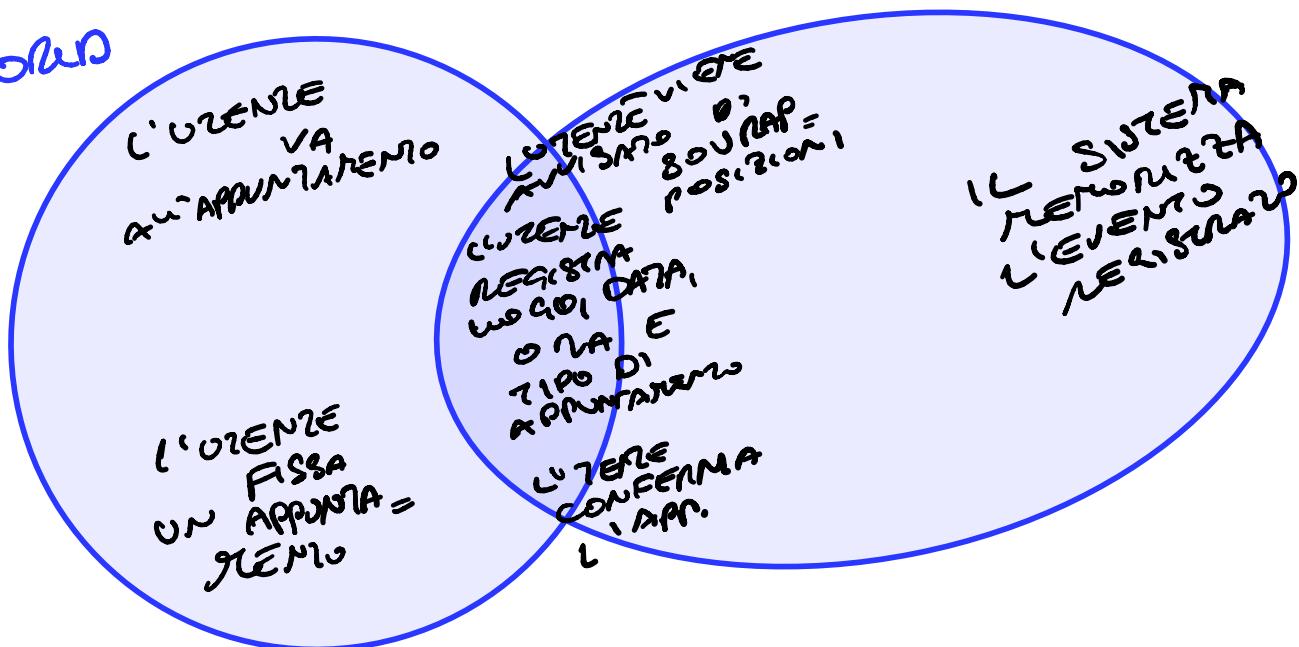
→ su "G2":

→ "lock" va mandato solo se ci sono push con qualche lock

→ SE PUSH COIN E C'È lock, IL sistema deve emettere unlock

3. APPUNTAMENTI

WORKS



alani:

G1: gli utenti tengono traccia degli eventi:

G2: gli utenti: non possono avere sovrapposizioni

D1: l'utente inserisce correttamente le info.

R1: il sistema permette di creare eventi specificando ...

R2: quando l'utente conferma la propria disp., l'oggetto va aggiornato entro 1 seconda

4. CESTIONE DI ADESSIONI

Bisogna sviluppare un SW per la gestione delle ammissioni a una Scuola superiore della propria provincia. Un genitore registra i dati del proprio figlio nell'applicazione e possono inserire riferite di ammissione per scuole diverse. Lo stato delle richieste può essere "OK", "KO", "NOT YET" e il genitore lo può consultare, e possono anche essere testificate in automatico. Volevano possono anche cancellare le applicazioni inviate. Dall'altra parte, le scuole possono accettare/rifiutare le occorrenti e possono condizionare la cancellazione che più scuole, possono consentire che riferite si esente e anche loro possono essere testificate ma mai se non ha una sua applicazione.

GOALS:

mejor
"gestiscono"

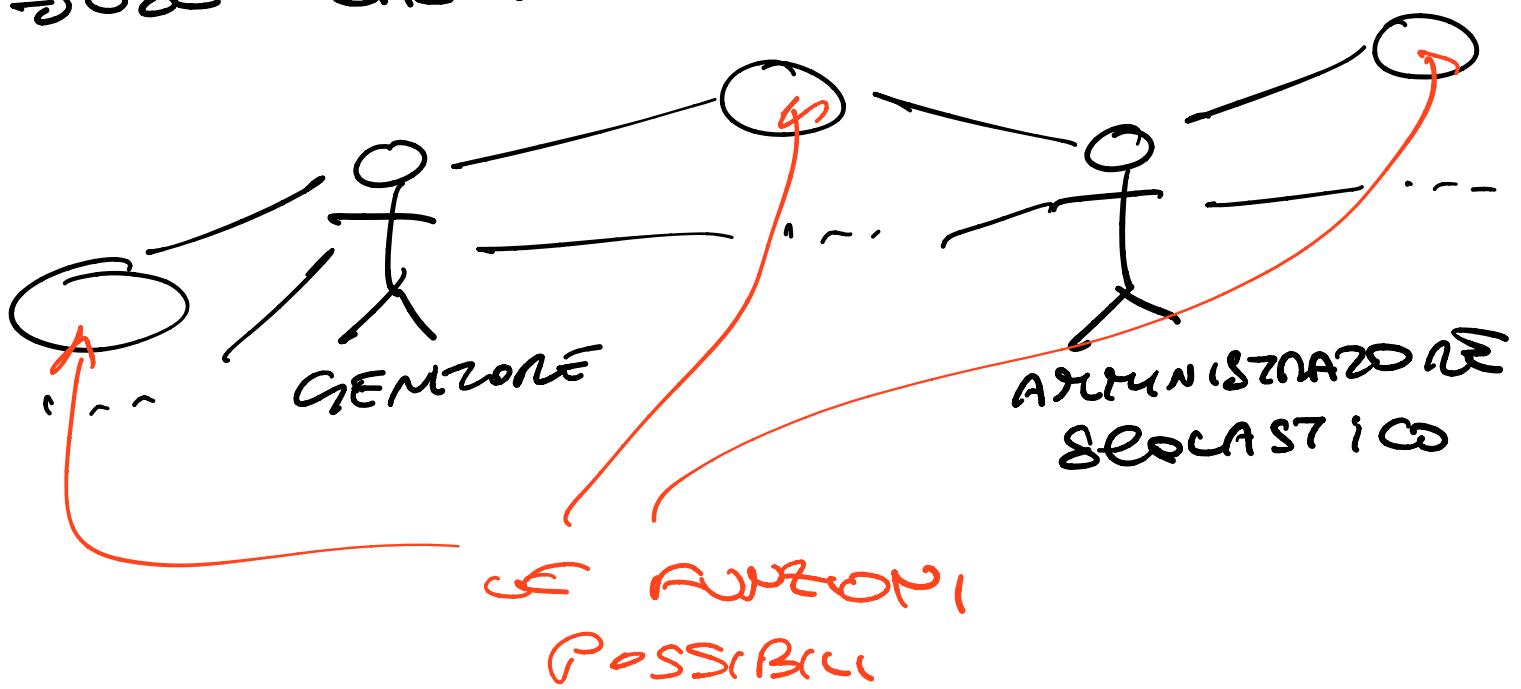
- G1: i genitori inviano le richieste di ammissione alle scuole
- G2: le scuole possono gestire le richieste di ammissione che ricevono
- G3: le scuole possono considerare le loro liste di richieste ricevute
 questo può essere una setta speciale
 di "gestire"

REQ. AND D. A. :

| GOALS | RICHIESTA | DECISO DI RE: |
|-------|--|--|
| G1 | <p>avere il fatto REQ. che uno possa avere e ricevere le proprie richieste di ammissione di scuole al NORFICO</p> | <p>DAT "IL SISTEMA PERMETTE DI SCEGLIERE L'INIZIO DI NORFICO"</p> |
| G2 | <p>avere il sistema che dene risposte deno gli istituti di cui si tratta e avere le liste dei risultati</p> | <p>IL SISTEMA CORRISPONDE PRESSO L'ANAGRAFE CONNESSIONE SEULE INFO. REL. AL FIGLIO</p> |
| | <p>IL SISTEMA INVIA LA RICHIESTA A TUTTE LE SCUOLE INTERESSATE</p> | <p>IL GENITORE INVIA... DATI VENUTI PER I VOTI ✓</p> |
| | <p>UN STATO INIZIALE DI UNA RICHIESTA È "NOT YET" UNA RICHIESTA NON PUÒ AVERE + STATI ✓</p> | |
| | <p>I VALORI ASSEGNAVABILI A UNA RICHIESTA SONO "NOT YET", "KO", "OK"</p> | <p>È POSSIBILE NEAMBIARE UN STATO DI UNA RICHIESTA</p> |
| | <p>IL CAMBIAMENTO DI UNA RICHIESTA VA NOTIFICATO SE Necessario</p> | |
| | <p>L'INVIO DI UNA RICHIESTA VA NOTIF. ALLA SCUOLA SE necessario</p> | |

DESIGN UML:

→ use CASE:



RESONCZO

Tendenzialmente si parla di fenomeni e della distinzione tra "world" e "MACHINE", dopo che si evitano di definire scavi ormai o GOAL/REQ.
DA ricordando che:

- R \wedge D \models G
- i GOAL sono in funzione del world
- i PHENOMENA
- i REQ. sono in funzione degli STATE
- i PHENOMENA

Lo scopo di un "GOAL" è capire cosa il nostro sistema deve garantire nel mondo, quello che se il nostro sistema non c'è fosse, cosa potrebbe essere garantito, da una persona che però non c'è nel nostro SW, ness persona che sia: "che cosa non va nel mondo?".

E poi dopo i requirement sono quelli che il nostro sistema deve fare al fine di garantire al mondo i GOAL, quelli che dimostra gli utenti: devono fare è una DOMAIN ASSUMPTION.

Dopo elisti si forse altra tutto
questo con CML e/o
Akog.

La compl. del req. è questo,
" $R \wedge D \neq G$ " significa:

- se il sist. rispetta i req.
- Se il mondo for da sua parte
- allora il mondo esterno ^{esterno} avrà
quel valore aggiunto che è
prefissato nei COAL