# LEZIONE 5
# I LINGUAGGI DI MODELLAZIONE && UML

Ingegneria del Software e Progettazione Web

Università degli Studi di Tor Vergata - Roma

Guglielmo De Angelis
guglielmo.deangelis@isti.cnr.it

# cosa sono ?

... ad ogni modo in un contesto simile ...

# ... quindi, qual è la vostra opinione ora?


cosa sono i “MODELLI”?

come vengono utilizzati i “MODELLI”?

si usano in pratica? e nelle comunità scientifiche? a cosa servono?

# cosa è un modello (di un sistema software) ?

A simplification of a system built with an intended goal in mind. The model should be able to answer questions in place of the actual system [Bézivin et al 2001]

# qualche esempio ...

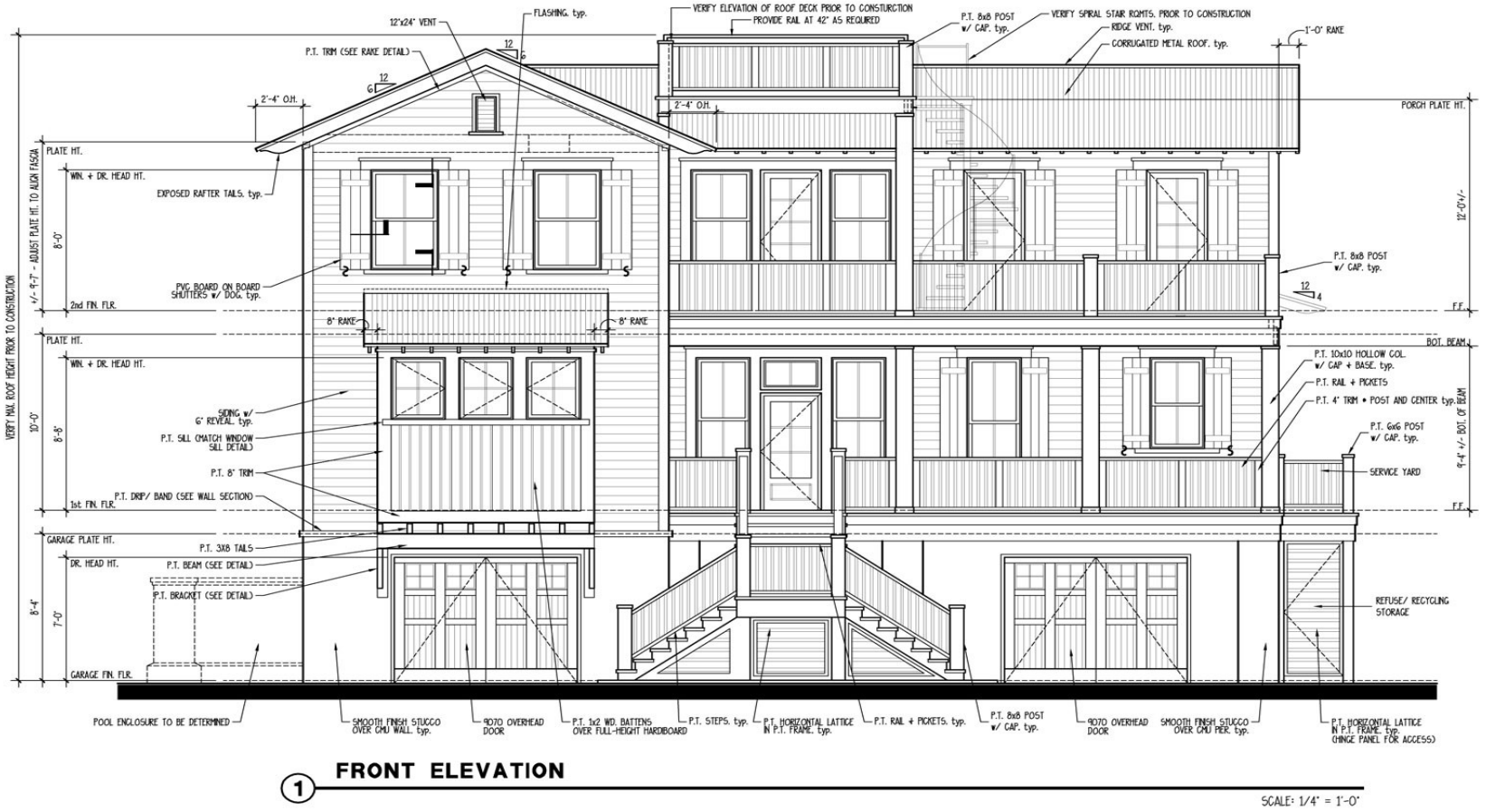
- le formule matematiche sono l'esempio più classico di modelli:

$$\vec{F}_{21} = -G \frac{m_1 \cdot m_2}{\left|\vec{r}_2 - \vec{r}_1\right|^2} \cdot \hat{r}_{21}$$


$$\frac{max\left(min\left(\frac{P_{Comp}-30}{20},2.375\right),0\right)+max\left(min\left(\frac{(P_{Yds}-3)}{4},2.375\right),0\right)+max\left(min\left(P_{TD}\times 20,2.375\right),0\right)+max\left(min\left(2.375-(P_{Intr}\times 25),2.375\right),0\right)}{6}\times 100$$

# qualche esempio ...

- i progetti depositati al catasto sono modelli:



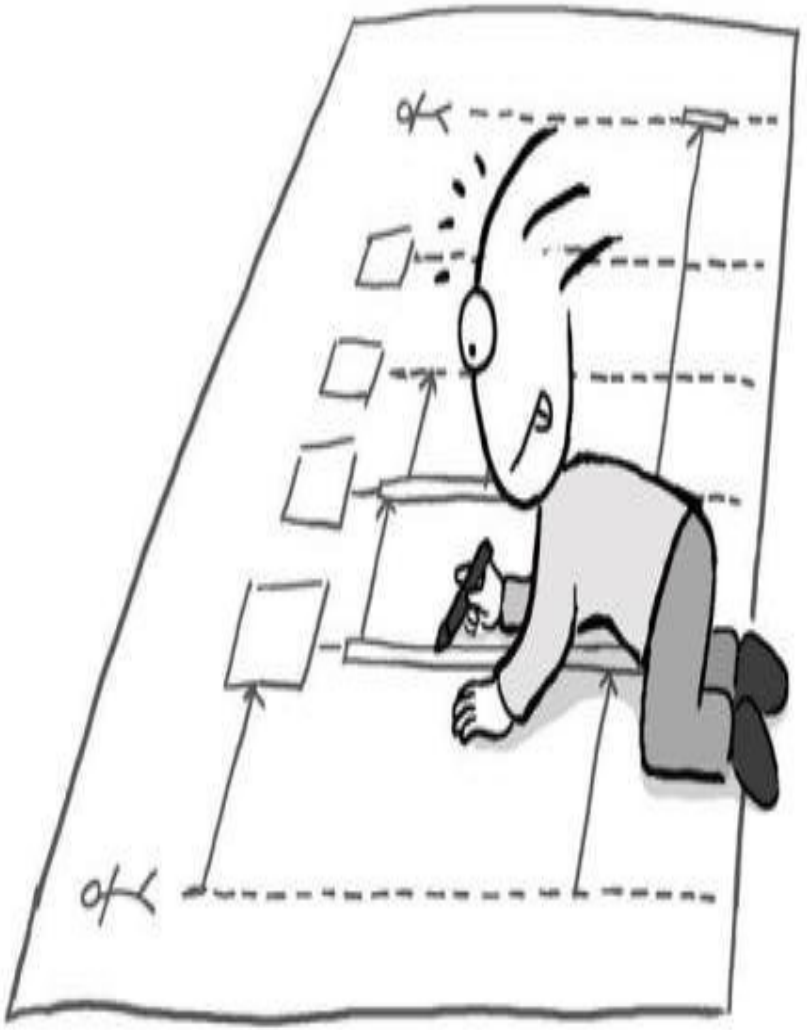

1. COSA C'ENTRA QUESTO CON IL SOFTWARE ?!?!?!?
2. COSA C'ENTRA QUESTO CON L'INGEGNERIA DEL SOFTWARE ?!?!?!

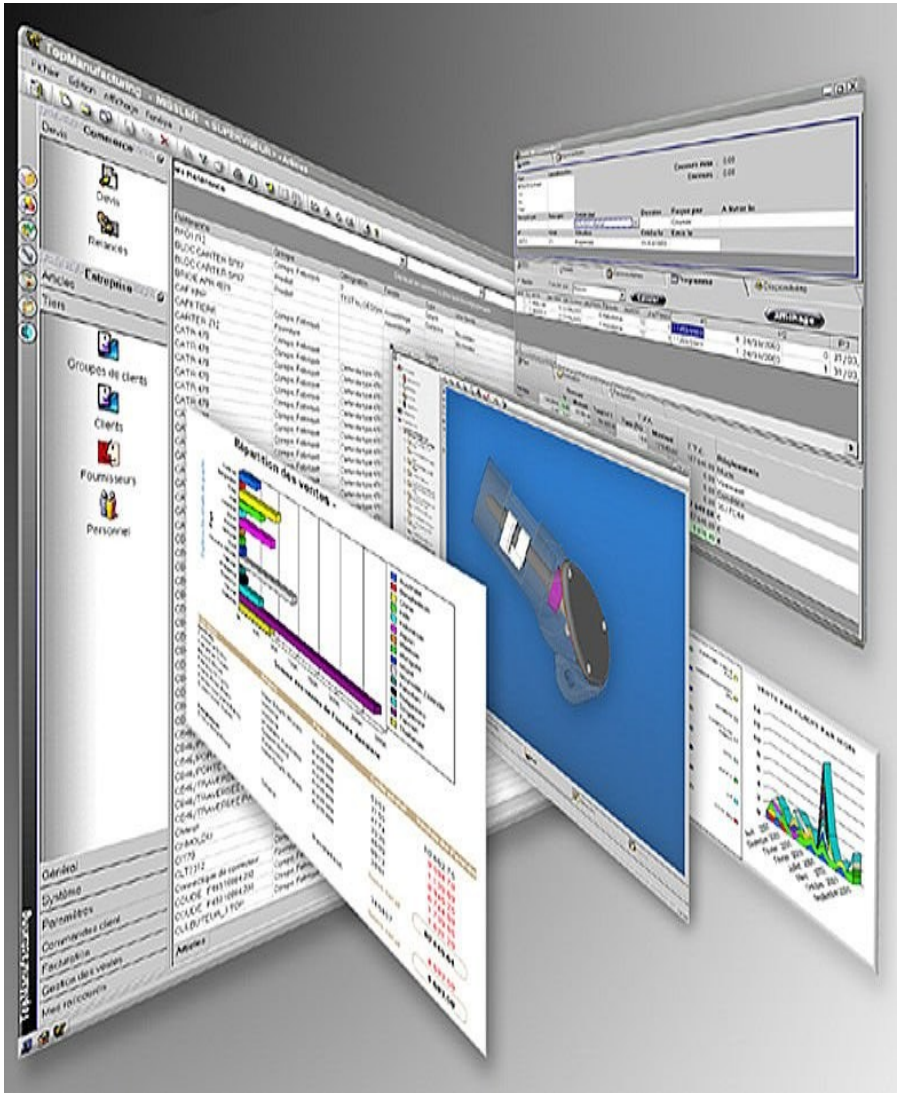
la risposta a queste domande passa per la seguente riflessione :

sviluppare software è :

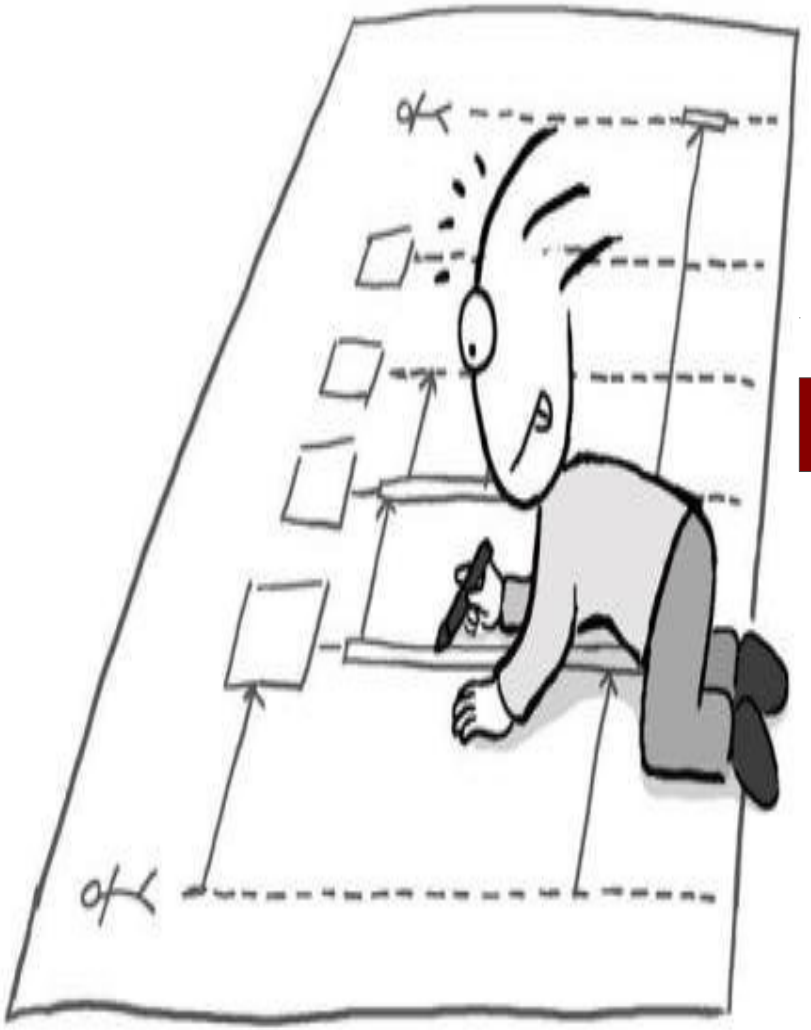
1) una forma di artigianato?
   - forma creativa basata su abilità personali e attitudini derivanti dallo studio e dall'esperienza
2) un processo ingegnerizzato “alla buona”?
   - forma estesa di artigianato che cerca di formalizzare alcune fasi del processo produttivo per svincolarlo dalle influenza di fattori umani
3) parte di un processo di produzione industriale ?
   - automatizzato
   - controllabile
   - ripetibile

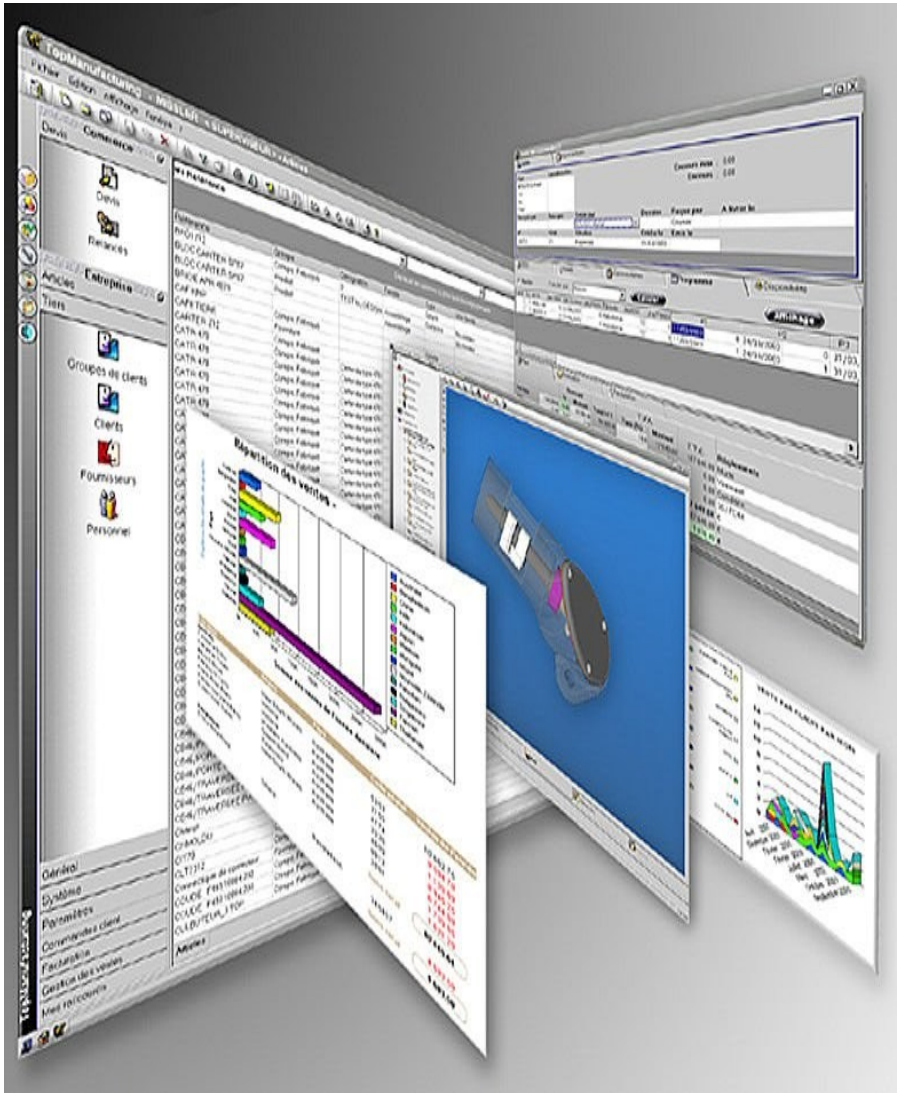
# IdS && modelli: rappresentare



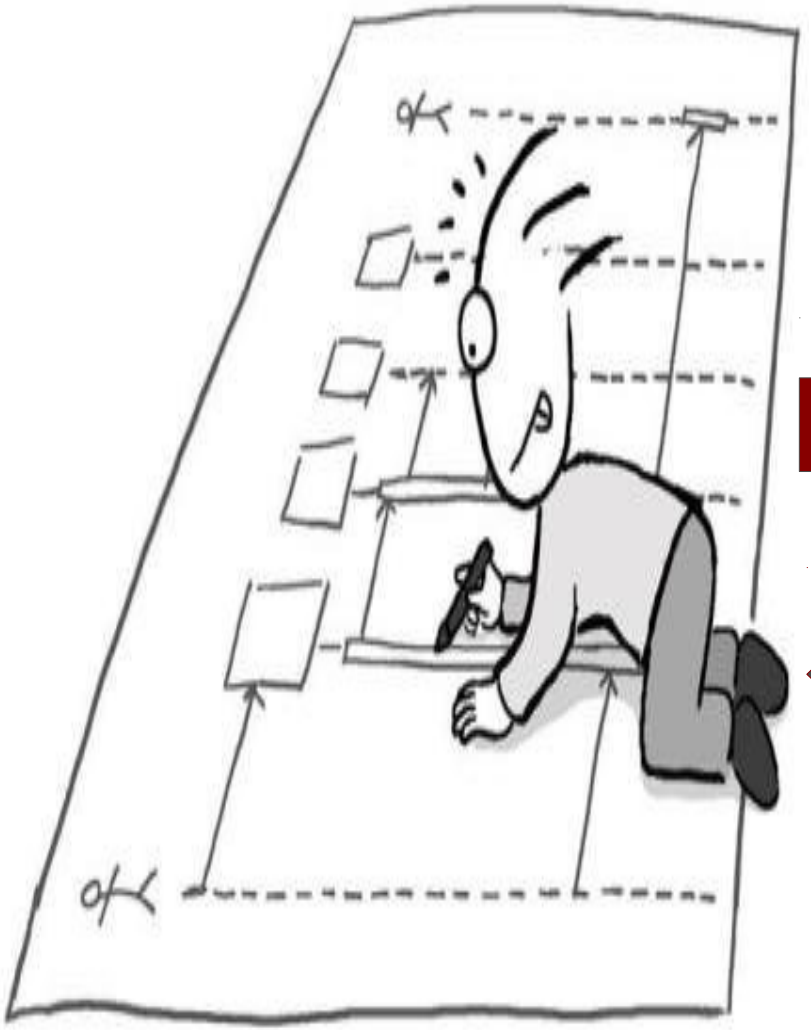


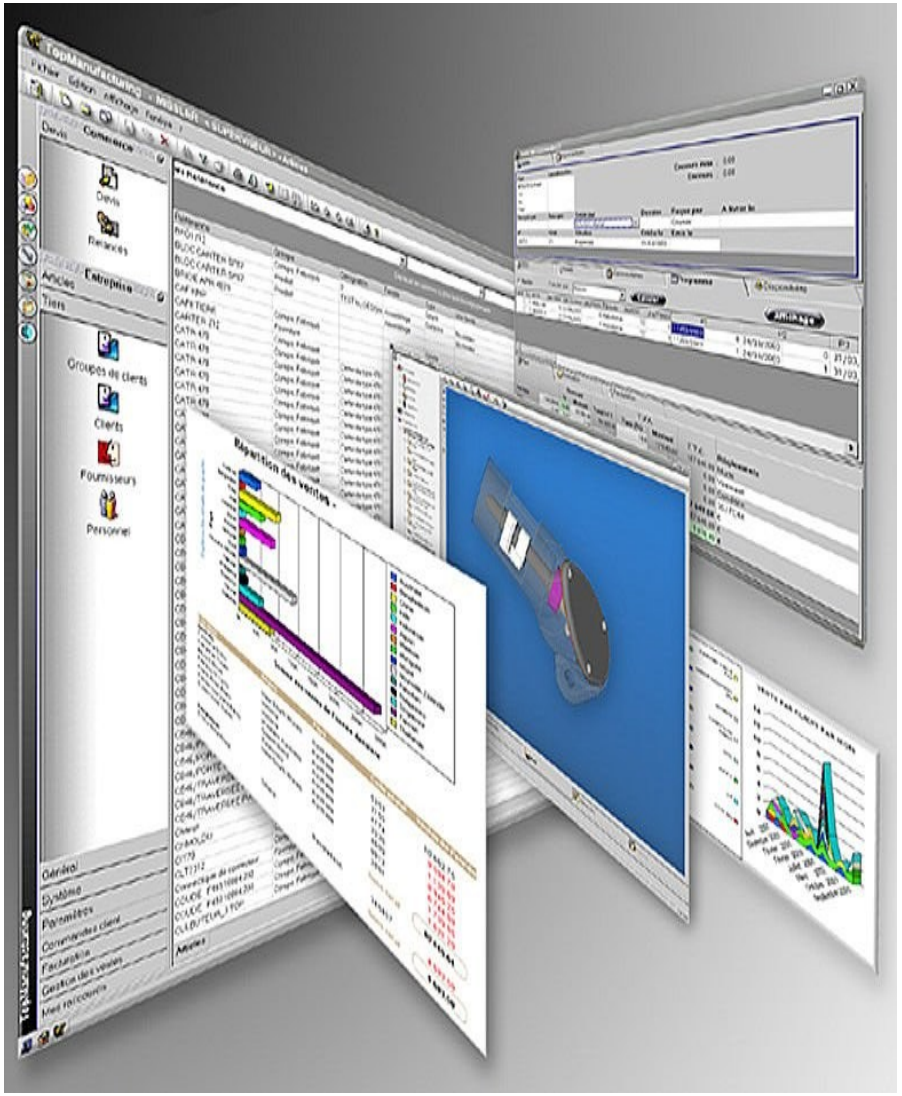
# IdS && modelli: sintetizzare



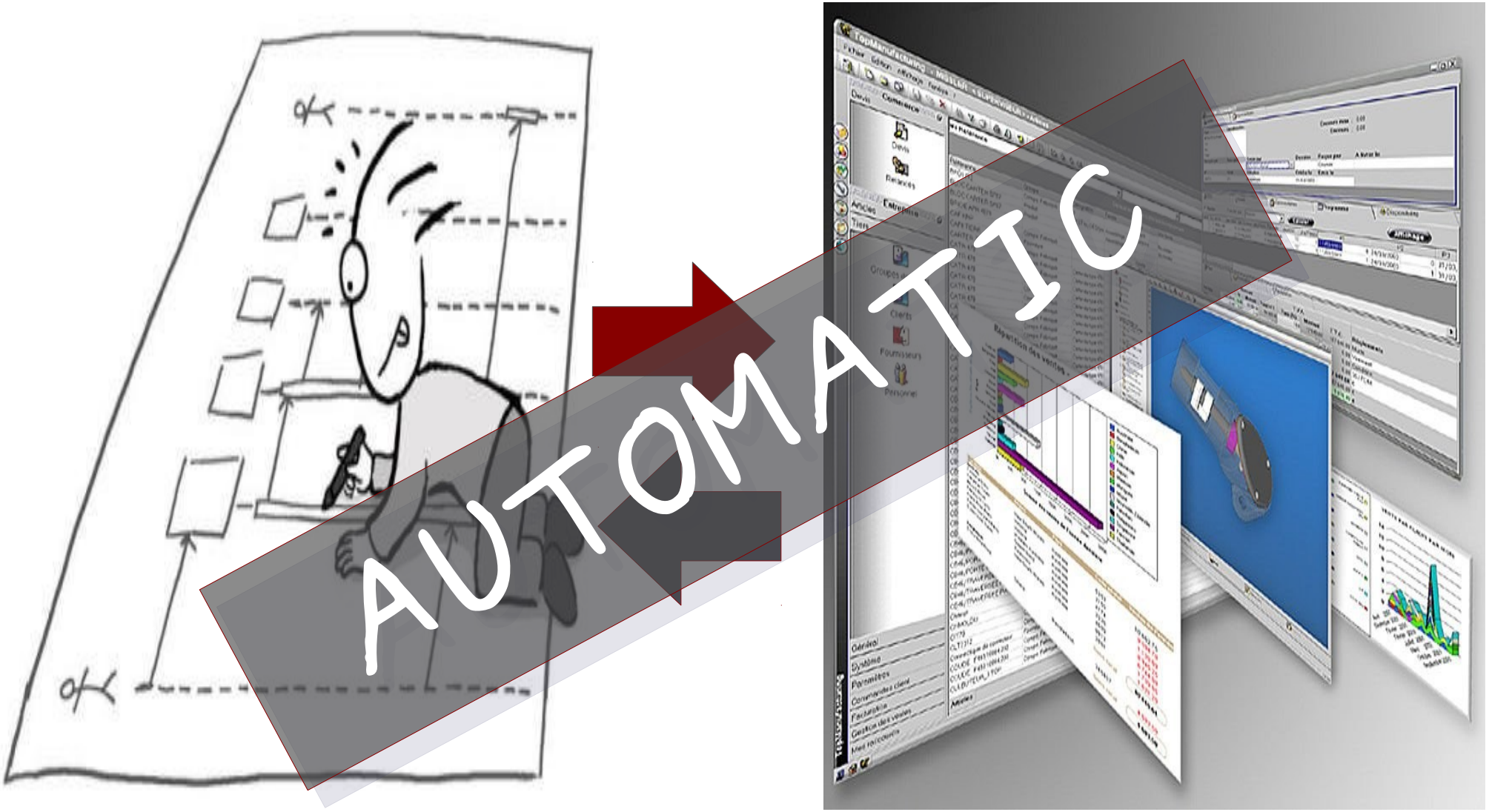


# IdS && modelli: analizzare





# IdS && modelli: automazione

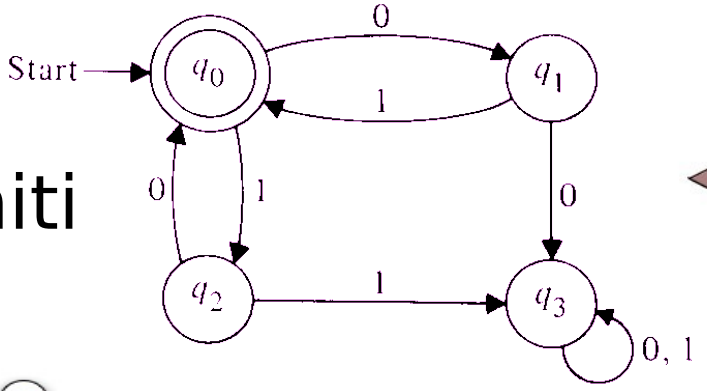


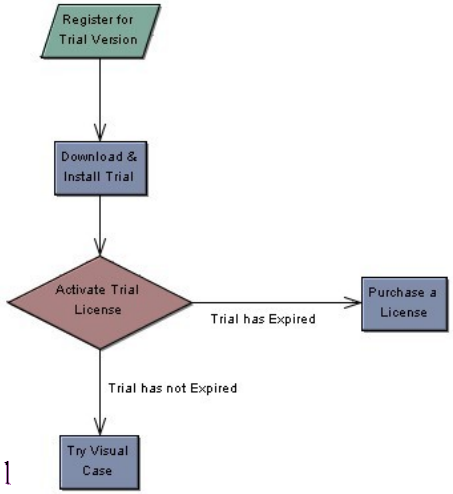
# importanza dei modelli software

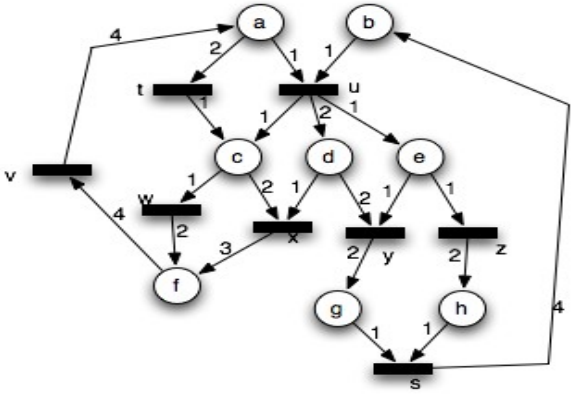
- il codice sorgente (fortunatamente) non è l'unico modo di pensare al software
- in organizzazioni umane complesse esiste il problema della conoscenza, trasmissione, e progettazione del software
- il software non è fatto soltanto di “linee di codice”, ma anche di
  - formati di dati
  - regole di accesso ai dati
  - ruoli ed operazioni associate
  - database
  - processi di business

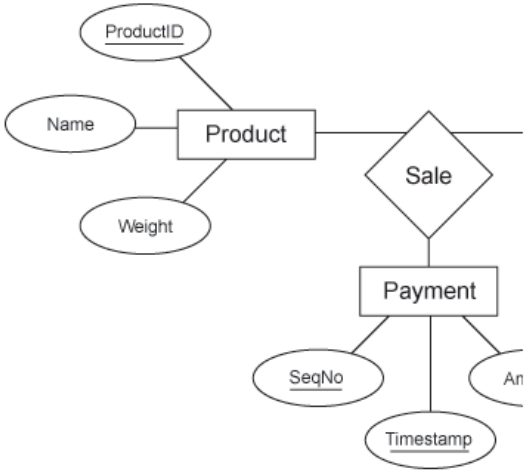
# i modelli ed il software – 1

- approcci alla modellazione del software:
  - diagrammi di flusso
  - macchine a stati finiti
  - petri net
  - diagrammi ER



# i modelli ed il software – 2

- la scelta del modello da usare dipende
  - dal problema che si vuole risolvere
  - da cosa è richiesto essere messo in evidenza per raggiungere una soluzione accettabile
- modelli esprimibili a differenti livelli di precisione
  - connessione col mondo reale
  - vari piccoli modelli correlati (o quasi...)

solo per confondere ulteriormente le idee

linguaggi di programmazione quali:

Java

SQL

C++

PHP

Perl

che conoscete ed usate sono a tutti gli effetti dei modelli imperativi/dichiarativi con i quali “modellate” sistemi software (i.e. “programmate”)

# “i modelli” && “questo corso”

- i modelli UML come strumento a supporto:
  - all'analisi di sistemi software
  - alla progettazione di sistemi software
  - alla programmazione
  - al controllo dell'evoluzione del software

- altre motivazioni
  - ridefinire/ristrutturare modelli
  - reverse engineering
  - supporto alla progettazione  dei casi di test
  - supporto alla implementazione dei casi di test
  - generazione di nuove viste

## cosa è UML?

- lo Unified Modelling Language è un linguaggio (dalla metà 90)
- riunisce costrutti/concetti di approcci esistenti
  - Booch: progettazione object-oriented
  - Harel: state machines
  - Jacobson: objectory vision
- inoltre
  - meccanismi di estensibilità
  - astrae dai linguaggi di programmazione
  - rappresentazione di concetti

# cosa è UML?

- lo Unified Modelling Language è un linguaggio (dalla metà 90)

*"In short, the Unified Modeling Language (UML) provides industry standard mechanisms for visualizing, specifying, constructing, and documenting software systems."*

# un po' di storia ... – 1

- inizialmente i disegni UML erano pensati per lo più come forma di documentazione
  - UML era utilizzato nello sviluppo di progetti software reali:
    - documentazione dei requisiti e dei casi di uso
    - documentazione delle soluzioni progettuali
  - i tool UML erano usati per disegnare diagrammi:
    - di supporto alla pianificare le attività
    - documentare per le fasi di sviluppo e progettazione
    - supporto grafico alla creazione di report

# un po' di storia ... – 2

- inizialmente i disegni UML erano pensati per lo più come forma di documentazione

- successivamente i modelli UML sono stati utilizzati per la progettazione, lo sviluppo, il testing e la documentazione di progetti software reali
  - i prodotti delle varie attività sono modelli e non disegni
  - i modelli delle varie fasi/soluzioni si referenziano rendendo le soluzioni adottate tracciabili

- i modelli UML anche come strumento per:
  - analisi dei requisiti e progettazione
  - pianificazione dei test e manutenzione
- i modelli UML devono anche essere la base:
  - generare  codice
    - da skeleton o pattern di esecuzione
  - risalire a modelli da pattern di codice
- il vero contributo di UML è nella visione di supportare i processi di sviluppo del software attraverso fasi (possibilmente) AUTOMATICHE/AUTOMATIZZABILI

# diagrammi per descrivere :

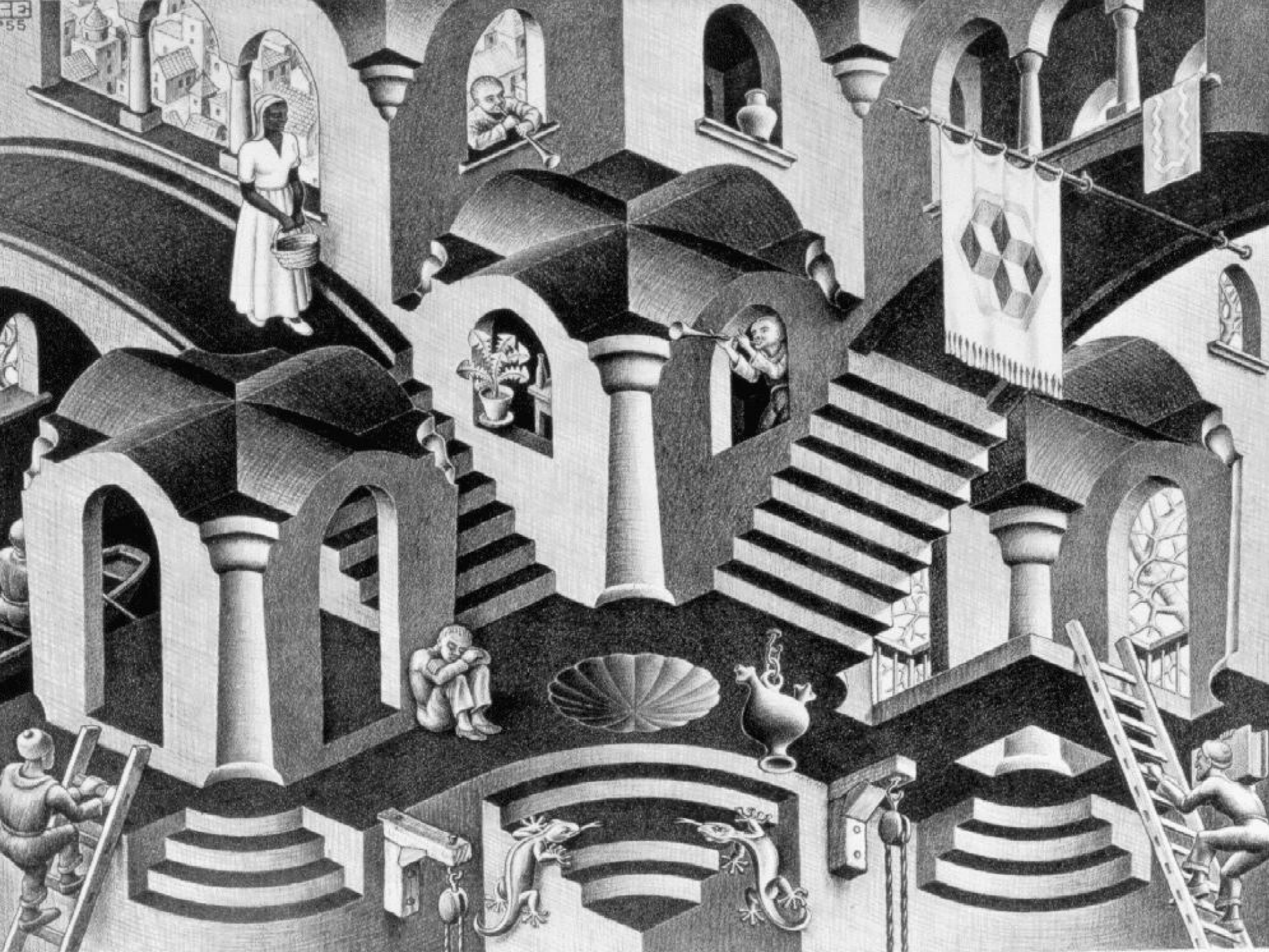
- struttura statica del sistema
  - structure diagrams
- comportamento del sistema
  - behavior diagrams
  - interazioni tra gli oggetti
    - interaction diagrams

# structure diagrams

- class diagrams
- object diagrams
- component diagrams
- deployment diagrams
- composite structure diagrams
- package diagrams

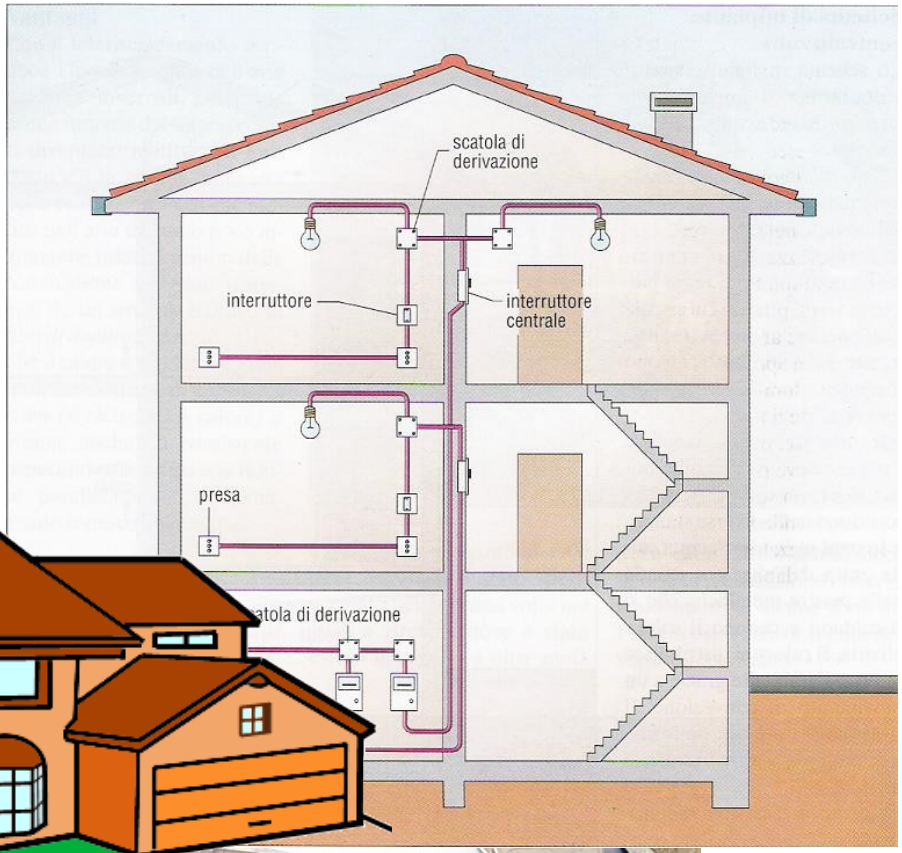
# behavior diagrams

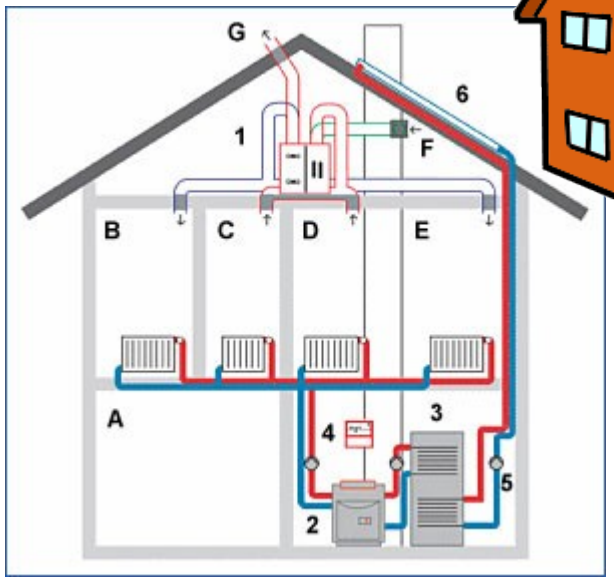
- state machine diagrams
- activity diagrams
- use case diagrams
- interaction diagrams
  - sequence diagrams
  - communication diagrams
  - interaction overview diagrams
  - timing diagrams




è una questione di punti di vista!!

# torniamo all'esempio della casa ...


# i molteplici aspetti di un sistema

- un sistema (software) espone sempre molteplici aspetti che devono essere trattati appositamente:
  - funzionale
    - legati a requisiti utente
    - legati a requisiti implementativi
    - legati a norme o leggi
  - extra-funzionale
    - legati ad aspetti di Qualità del Servizio offerto
    - legati a vincoli sulle risorse a disposizione
  - aspetti organizzativi
    - legati ad aspetti tecnologici e non

# i molteplici aspetti di un sistema

- non è realistico pensare che il processo di sviluppo di un software sia sempre lineare
- non è realistico pensare che il processo di sviluppo di un software sia legato ad un unico livello di astrazione
- non è realistico pensare che un singolo modello catturi tutte le informazioni necessarie per la descrizione di un software

# come UML “vede” il software – 1

UML consente di scomporre un processo di sviluppo software secondo (almeno) 5 prospetti

- use case view
- logical view
- implementation view
- process view
- deployment view

ogni vista enfatizza la descrizione di aspetti specifici del sistema in diverse fasi dello sviluppo

# use case view

- modello funzionale come percepito dagli attori esterni (utenti o altri sistemi)
- modellazione/analisi dei requisiti utente
- obiettivo
  - struttura esterna del sistema (black-box)
  - il cosa (e non il come)
- individuare tutti gli attori, casi d'uso e loro relazioni

# use case view

- modello funzionale come percepito dagli attori esterni (utenti o altri sistemi)
- modellazione/analisi dei requisiti utente
- obiettivo
  - struttura esterna del sistema (black-box)
  - il cosa (e non il come)
- individuare tutti gli attori, casi d'uso e loro relazioni
- TARGET: clienti, progettisti, sviluppatori, tester

## logical view

- descrive come sono progettate le funzionalità del sistema
  - e.g. organizzazione a oggetti del sistema
- progettazione della struttura del sistema (white-box)
- come le funzionalità devono essere
- realizzate
  - in teoria dovrebbe essere già chiaro il cosa

# logical view

- descrive come sono progettate le funzionalità del sistema
  - e.g. organizzazione a oggetti del sistema
- progettazione della struttura del sistema (white-box)
- come le funzionalità devono essere
- realizzate
  - in teoria dovrebbe essere già chiaro il cosa
- TARGET: progettisti e sviluppatori

# implementation view

- organizzazione del codice del sistema in moduli e loro interdipendenze
- organizzazione degli eseguibili
- modelli specifici per l'ambiente di esecuzione

# implementation view

- organizzazione del codice del sistema in moduli e loro interdipendenze
- organizzazione degli eseguibili
- modelli specifici per l'ambiente di esecuzione
- TARGET: sviluppatori, progettisti, tester

## process view

- comprende modelli che descrivono
  - dei processi da eseguire
  - delle entità che eseguono i processi
- si usa per
  - un utilizzo efficace delle risorse
  - stabilire l’esecuzione parallela degli oggetti
  - gestione di eventi asincroni (esterni al sistema)
- fortemente basato su modelli che descrivono la dinamica del sistema

- comprende modelli che descrivono
  - dei processi da eseguire
  - delle entità che eseguono i processi
- si usa per
  - un utilizzo efficace delle risorse
  - stabilire l'esecuzione parallela degli oggetti
  - gestione di eventi asincroni (esterni al sistema)
- fortemente basato su modelli che descrivono la dinamica del sistema
- TARGET: sviluppatori ed integratori di sistema

comprende :

- modelli che descrivono la topologia e l'organizzazione delle macchine fisiche
  - e.g. computer, device mobili, connessioni fisiche tra i nodi
- modelli che descrivono come le parti del sistema software sono mappate sull'architettura fisica

comprende :

- modelli che descrivono la topologia e l'organizzazione delle macchine fisiche
  - e.g. computer, device mobili, connessioni fisiche tra i nodi
- modelli che descrivono come le parti del sistema software sono mappate sull'architettura fisica
- TARGET: sviluppatori, integratori di sistema, tester

# come UML “vede” il software – 2

UML consente di scomporre un processo di sviluppo software secondo (almeno) 5 prospetti

- use case view
- logical view
- implementation view
- process view
- deployment view

MA, COME SI METTONO IN RELAZIONE QUESTE VISTE?!? QUALE PROCESSO DI SVILUPPO SI ADOTTA QUANDO SI USA UML ?!?

# UML ed il processo di sviluppo

- UML non prescrive nessun processo di sviluppo per il software
- UML è “semplicemente” un linguaggio
- è possibile usare il linguaggio UML con il processo di sviluppo (o metodologie) che si ritiene più opportuno:
  - Waterfall, Iterative,
  - Metodologie Agile (±), Metodologie Model-Based, Product Family ...tra i tanti  c'è anche RUP
- in generale un processo di sviluppo non è vincolante per l'uso di UML

attenzione!!!

SOLO PER COMPLETEZZA RICORDO CHE:

- UML non è l'unica scelta possibile alla modellazione
- non è sempre fondamentale
- esistono molti ambienti che non hanno bisogno/non vogliono UML
- UML è una soluzione che al momento è supportata da un notevole interesse economico
- è importante conoscere UML perché molto richiesto dal mondo “industriale”