Model View Controller (MVC)

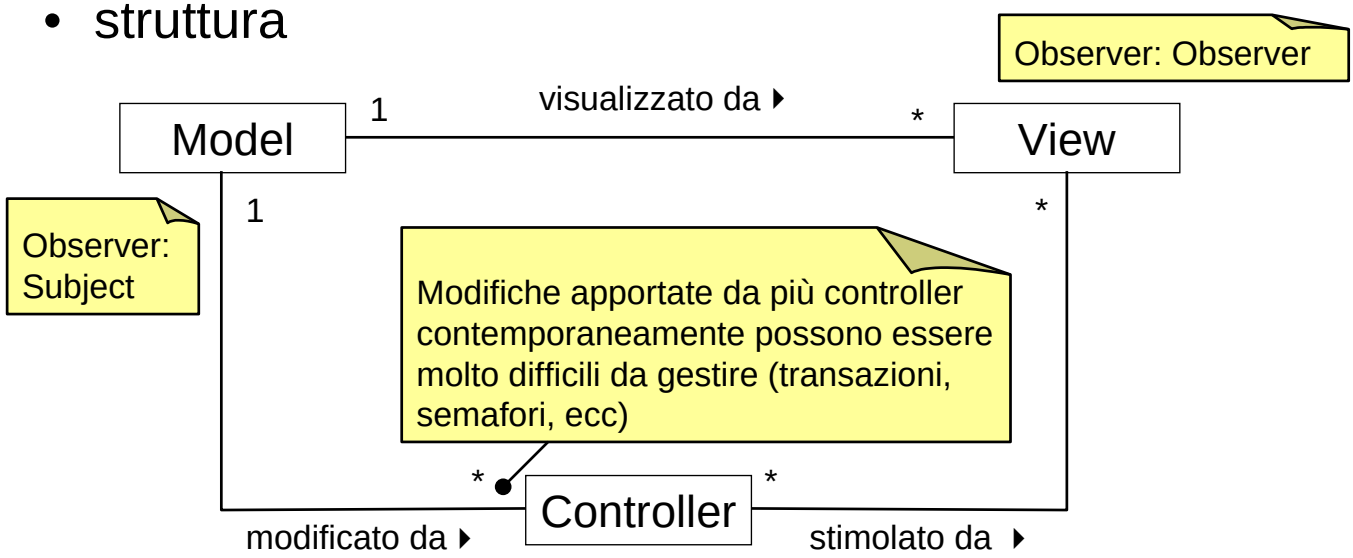
in sintesi

- è una applicazione del pattern Observer alle interfaccie utente (GUI ma non necessariamente)
- lunga tradizione in smalltalk
- non è propriamente un design patter ma un “architectural pattern”
  - perché i vari ruoli possono essere ricoperti da insiemi di classi anziché da singole classi
- intento

  disaccoppiare:
  - rappresentazione del modello di dominio (model)
  - interfaccia utente (view), non necessariamente GUI
  - controllo dell’interazione uomo-macchina (controller)

il pattern MVC

- struttura

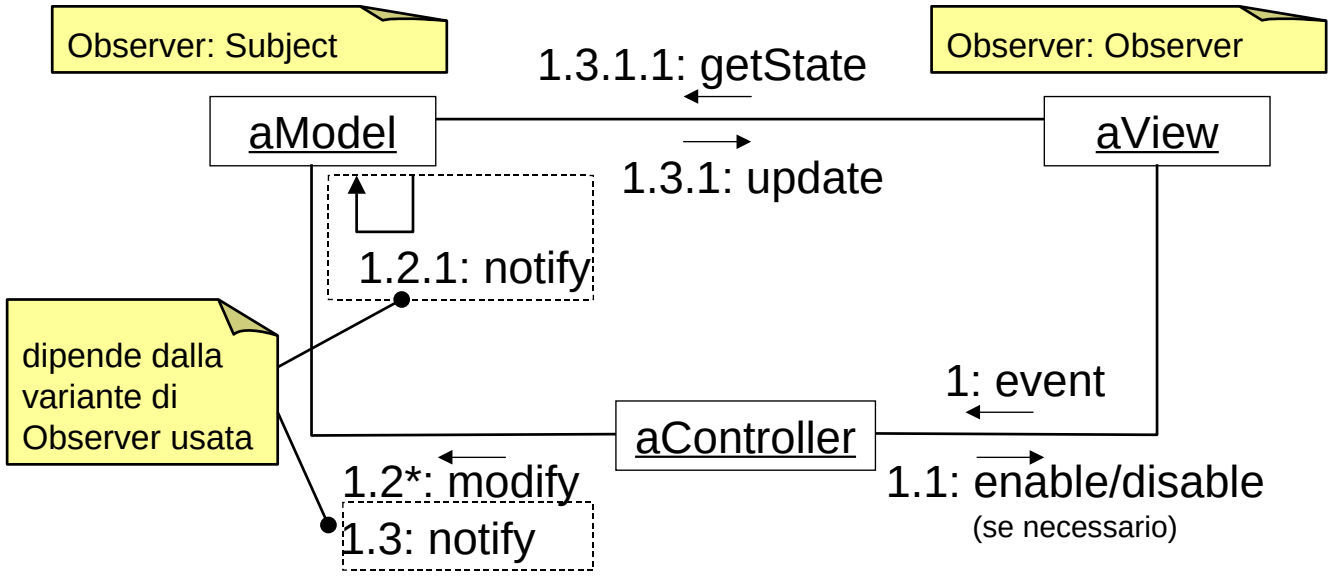


qui, per semplificare, supponiamo che modello vista e controller possano essere implementati (o rappresentati) con un sola classe (o interfaccia)

il pattern MVC

ESERCIZIO
dai il diagramma di sequenza equivalente a questo collaboration diagram per le due varianti di Observer

- interazione



qui, per semplificare, supponiamo che il modello sia rappresentato da un solo oggetto, osservato con una sola view e controllato con un solo controller

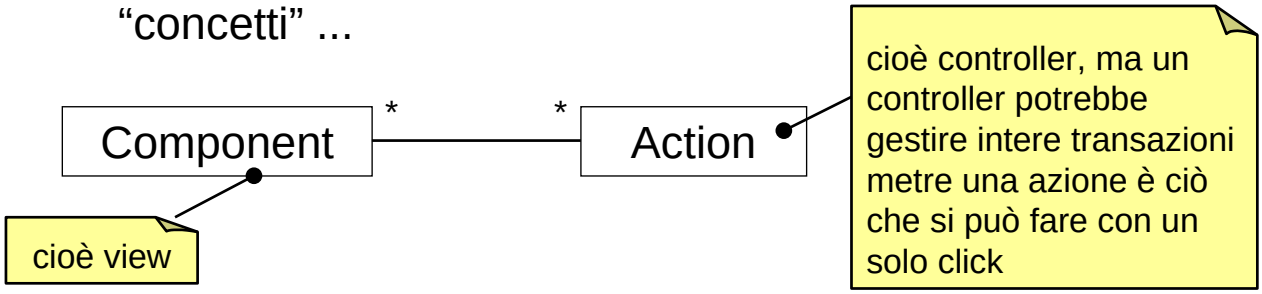
MVC in ambiente grafico

- l'approccio MVC viene spesso applicato a GUI
- nasce il problema di conciliare
  - gli obiettivi di disaccoppiamento di MVC
  - i vantaggio dati dall'utilizzo di toolkit grafici
  - portabilità del codice o di gran parte di esso

- infatti spesso i toolkit grafici...
  - sono dipendenti dalla piattaforma
  - non impongono uno schema progettuale (MVC o altro)
  - se utilizzati in maniera "ingenua" portano a sistemi mal progettati (alto accoppiamento e bassa coesione)

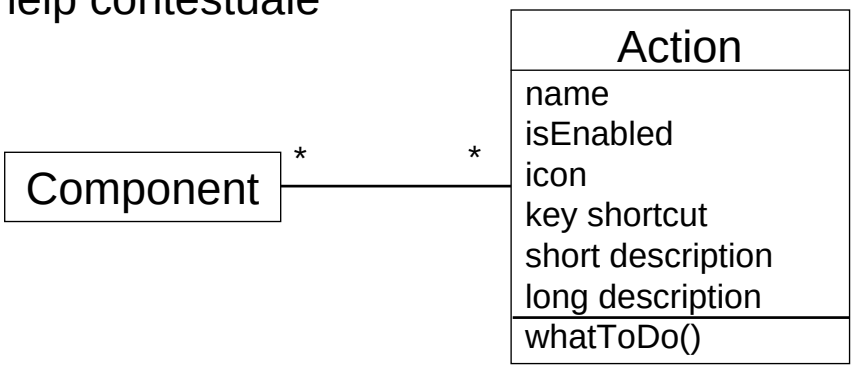
MVC in ambiente grafico

- Il modello è indipendente dall'ambiente grafico: nessun supporto specifico è possibile
- Le view sono formati da **componenti** grafici riutilizzabili (bottoni, scrollbar, liste, ecc.)
  - moltissime librerie di supporto in molti linguaggi
- I controller "aggregano" **azioni** che un utente può effettuare sul modello
  - le moderne interfacce grafiche associano alle azioni altri "concetti" ...

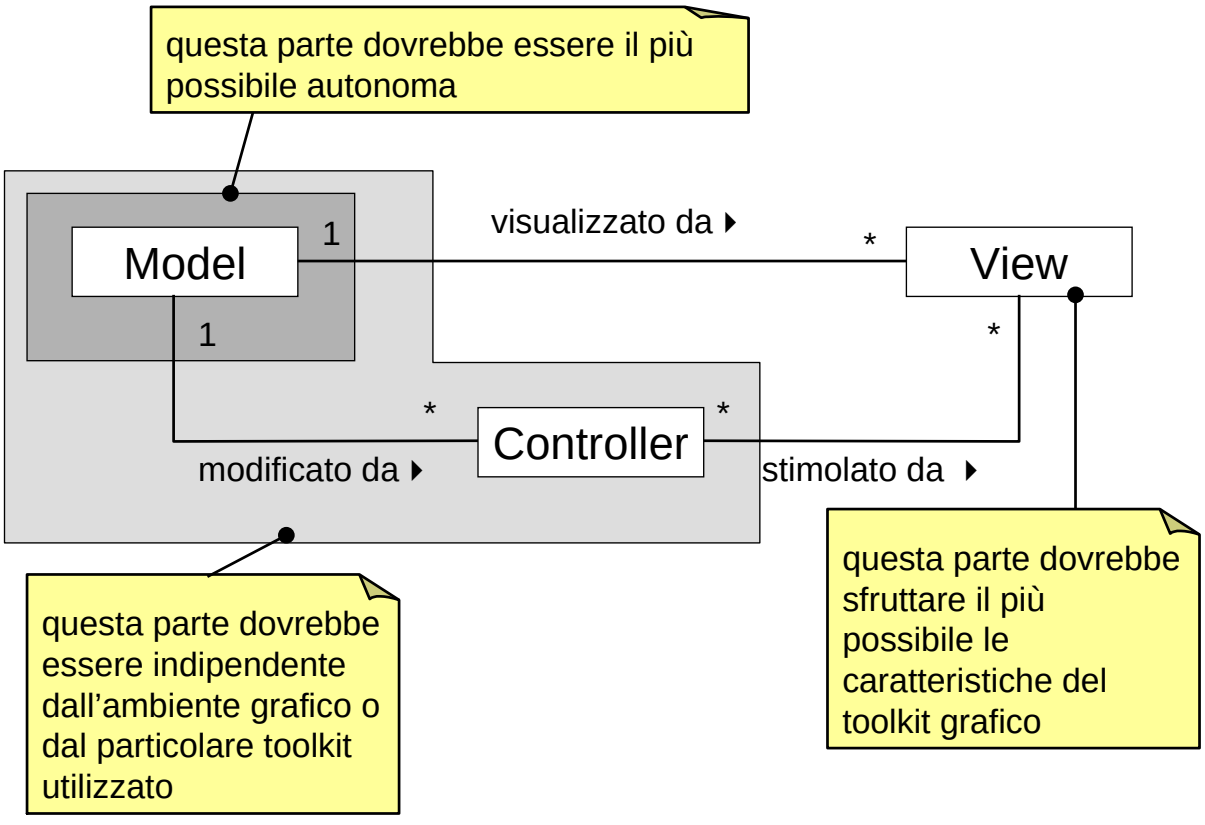


le azioni dell'utente in ambiente grafico

- non sempre un'azione è ammessa
  - quando una azione non è ammessa tutti i componenti grafici che permettono quell'azione devono essere disabilitati
- una azione è rappresentata da
  - icona, nome
- associata ad una azione ci sono una o più descrizioni
  - tooltip, help contestuale



MVC e riuso



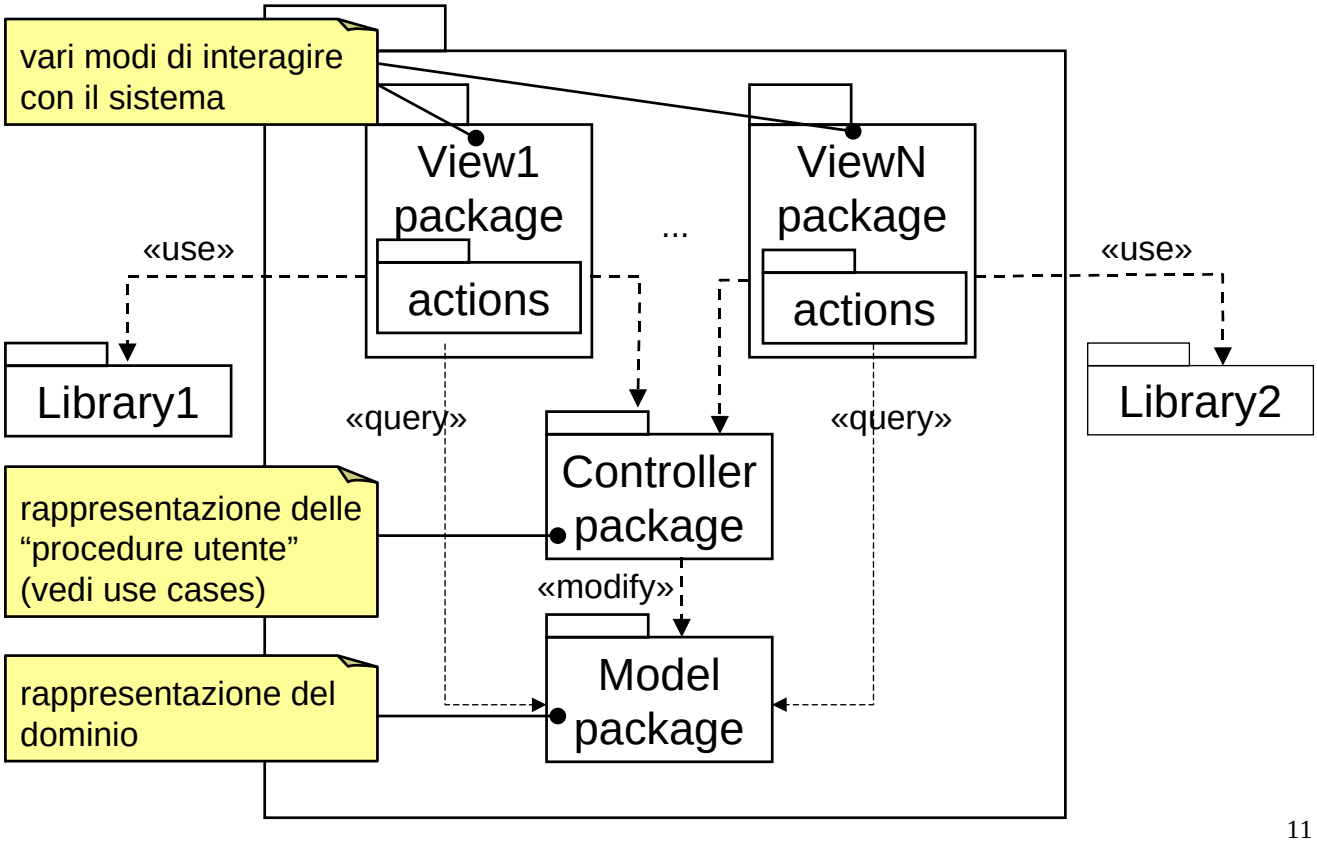
controller: vari punti di vista

possiamo individuare due punti di vista rispetto al ruolo de controller

- punto di vista vicino a model
  - il controller dovrebbe raggruppare le azioni/transazioni “compesse” che possono essere effettuate sul modello
  - è del tutto indipendente da aspetti tecnologici legati all’interfaccia grafica
- punto di vista vicino alla view
  - il controller gestisce eventi (mouse clicks)
  - deve fornisce una interfaccia che sia “compatibile” con il toolkit grafico utilizzato
  - può rappresentare il concetto di azione “elementare” effettuabile tramite l’interfaccia grafica
    - abilitazione/disabilitazione, nomi, tooltip, ecc.

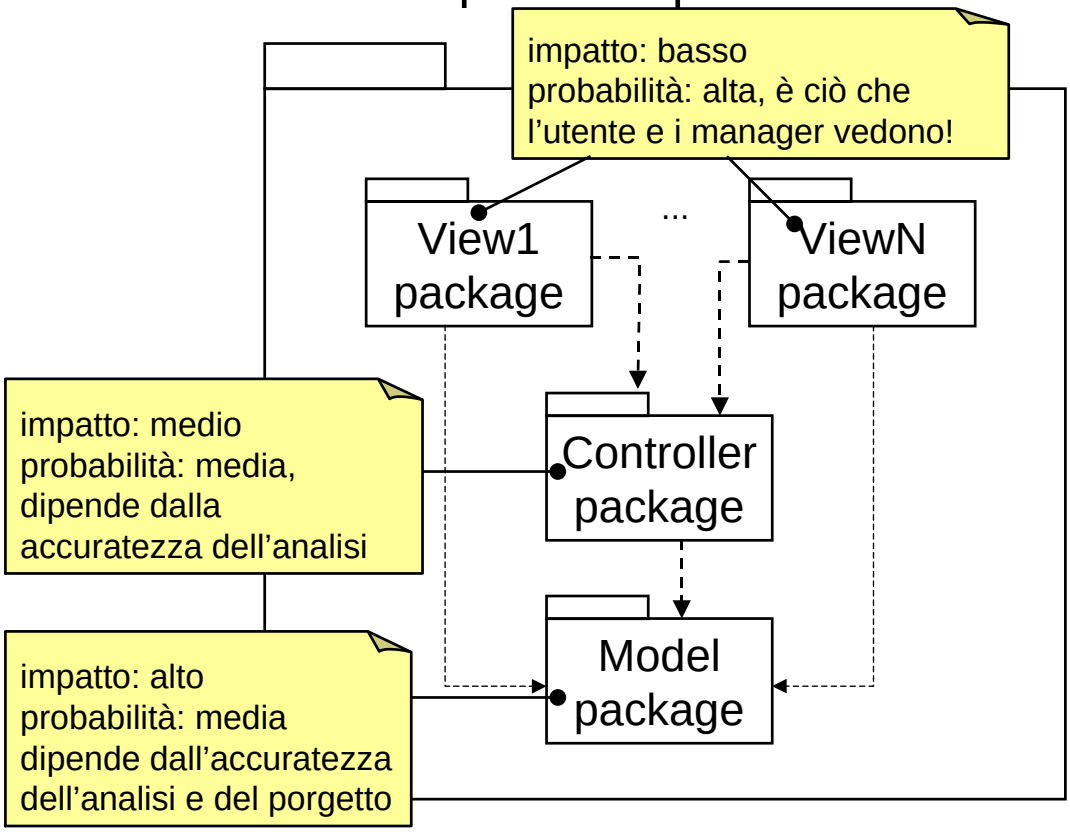
- entrambe le esigenze possono essere soddisfatte
  - dividiamo le responsabilità in due categorie di oggetti distinti
  - ciascuna categoria soddisfa una delle esigenze
  - categorie:
    - controller (propriamente detti)
      - tutto ciò che non è legato al toolkit grafico
    - azioni
      - tutto ciò che è legato a toolkit grafico
      - sono spesso classi molto piccole che delegano la maggior parte delle attivitè al controller
      - possono essere considerate degli adapter verso il controller

MVC: architettura target

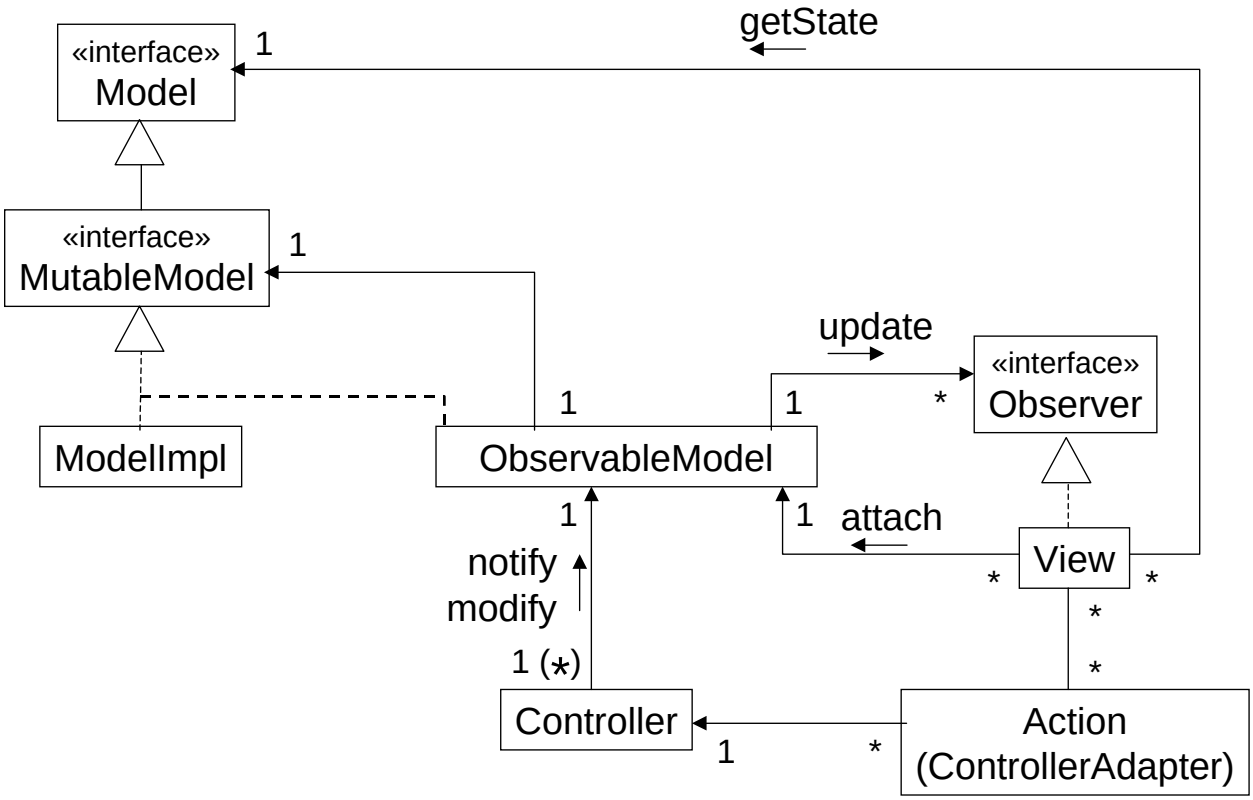


MVC: cambiamenti al sistema

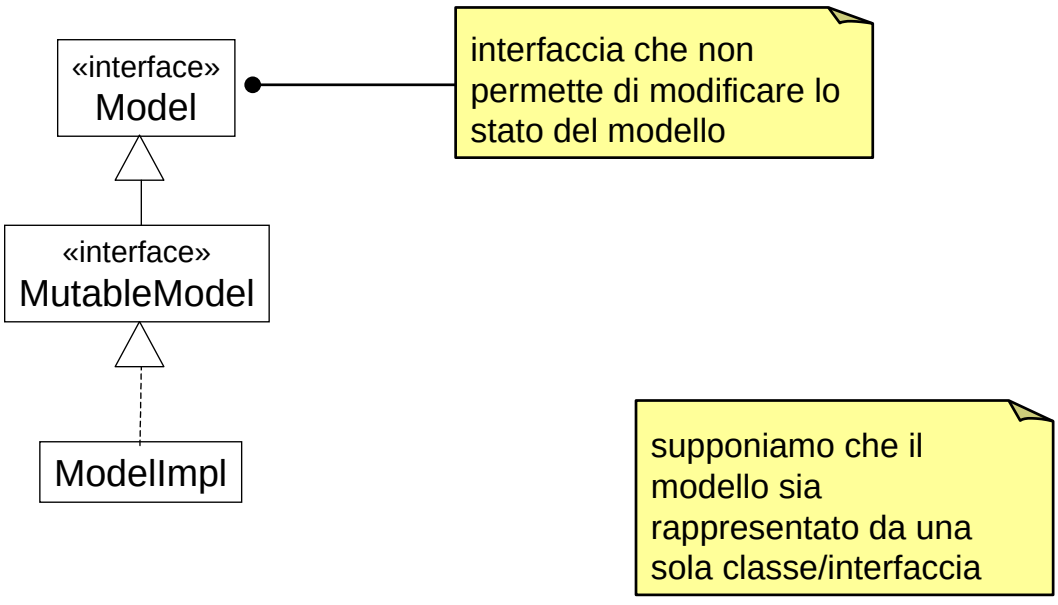
impatto vs. probabilità



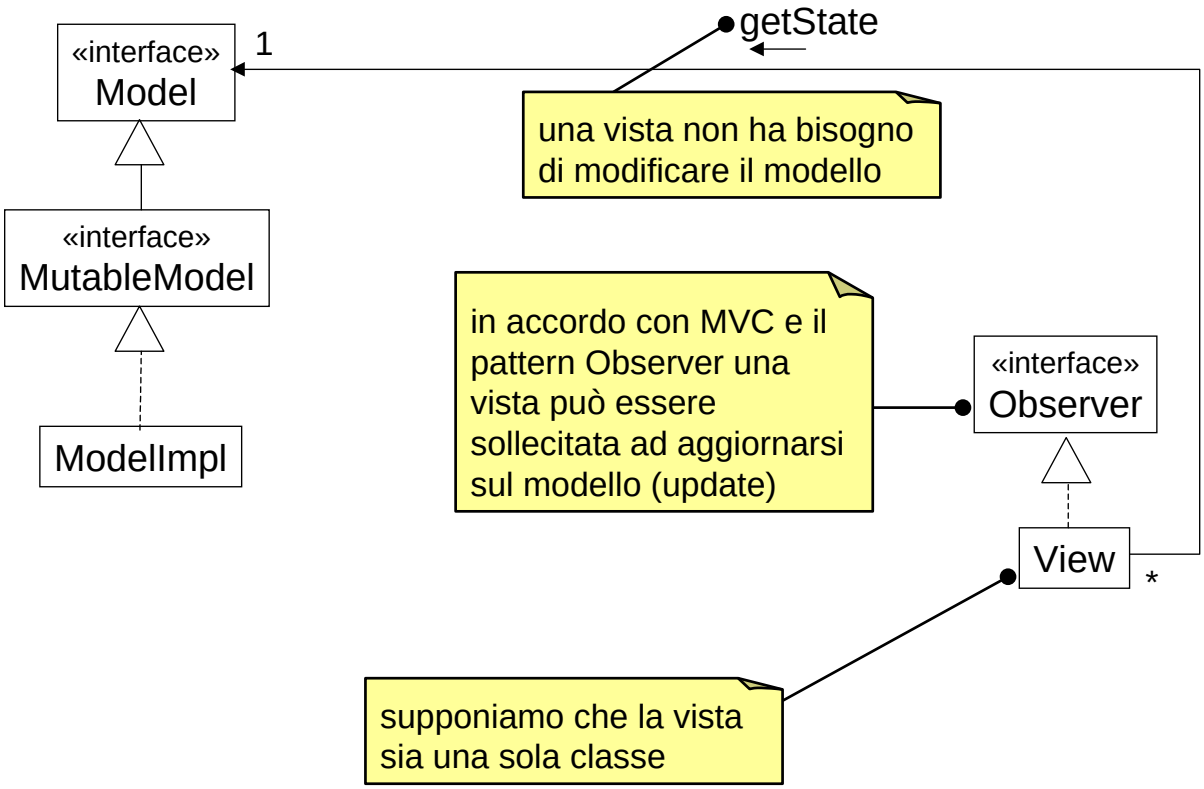
MVC: dettagli di progetto



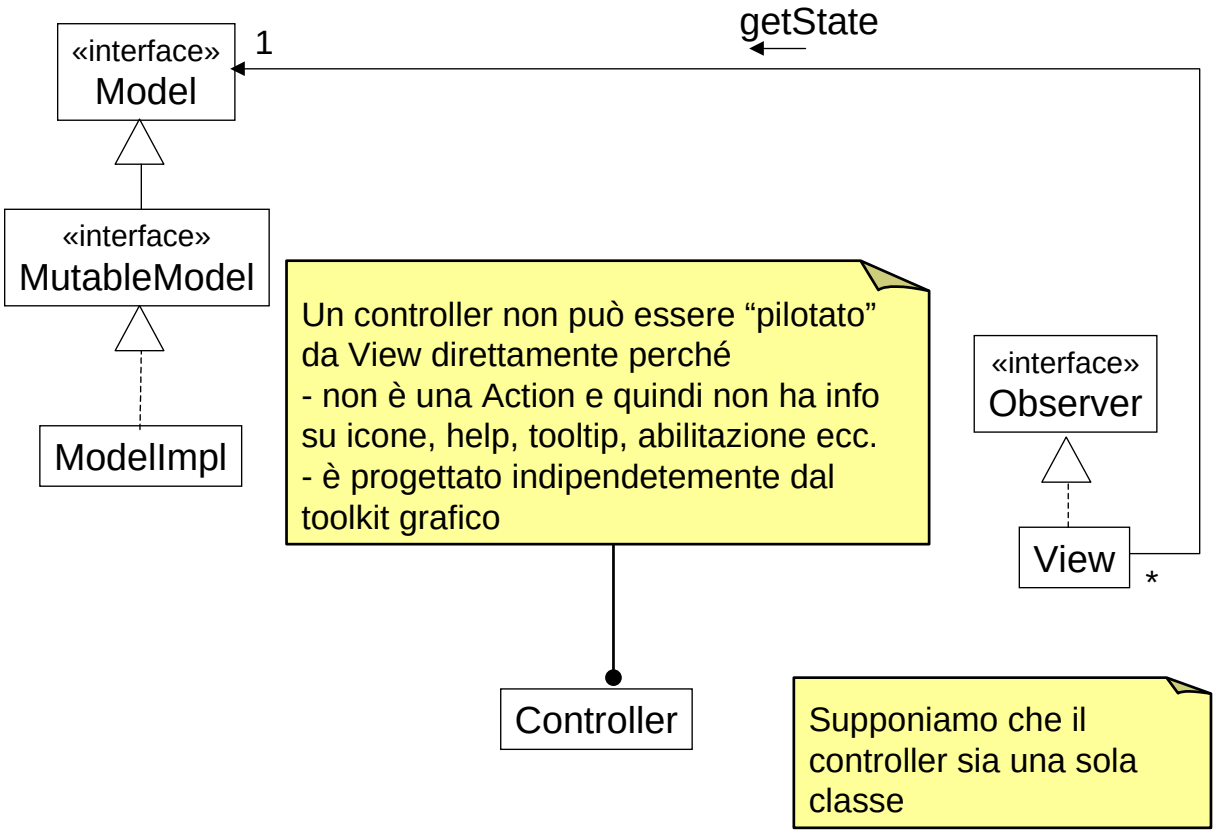
MVC: i dettagli un passo per volta



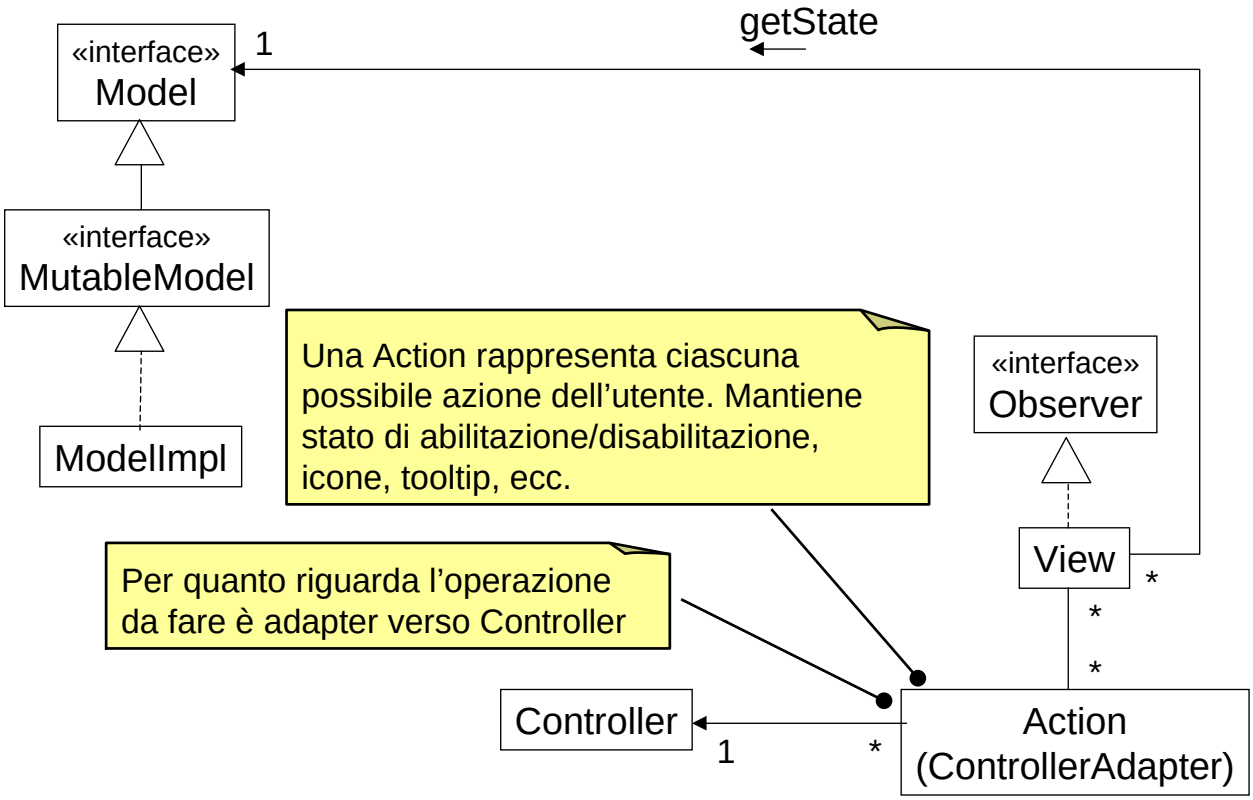
MVC: i dettagli un passo per volta



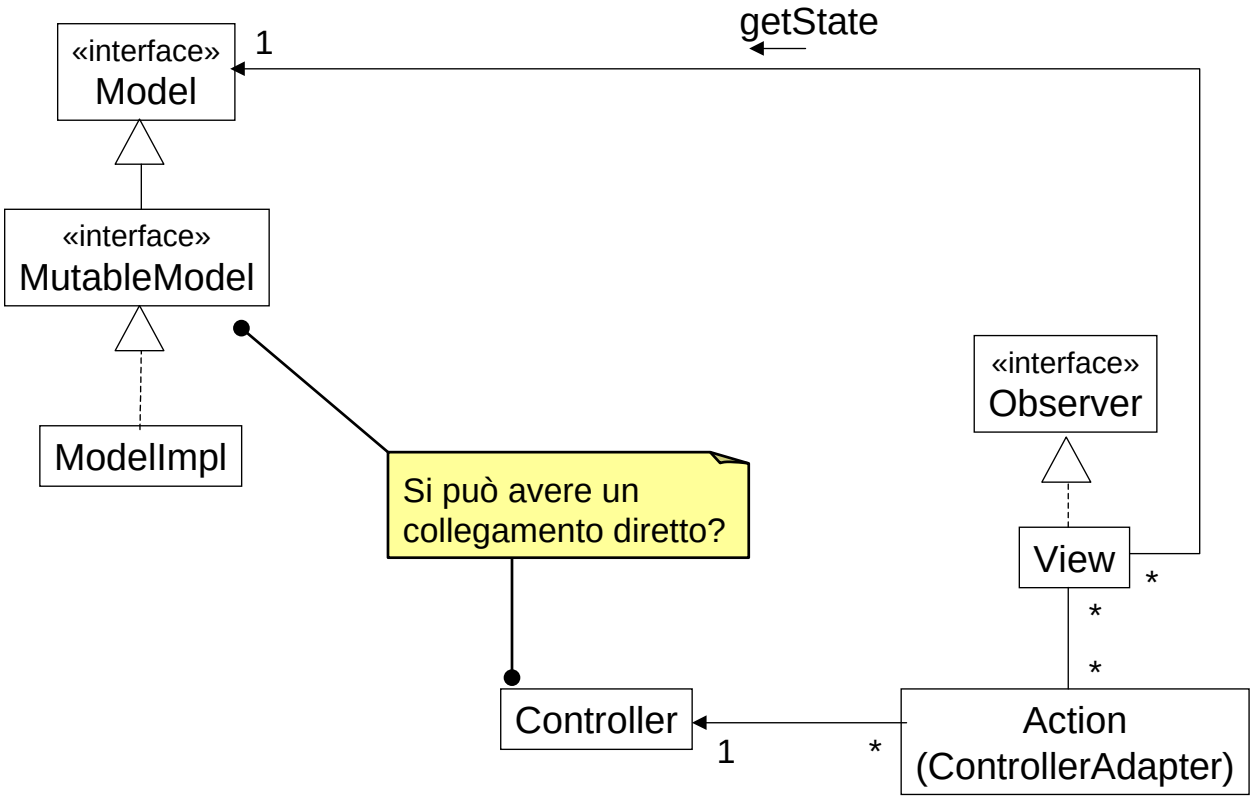
MVC: i dettagli un passo per volta



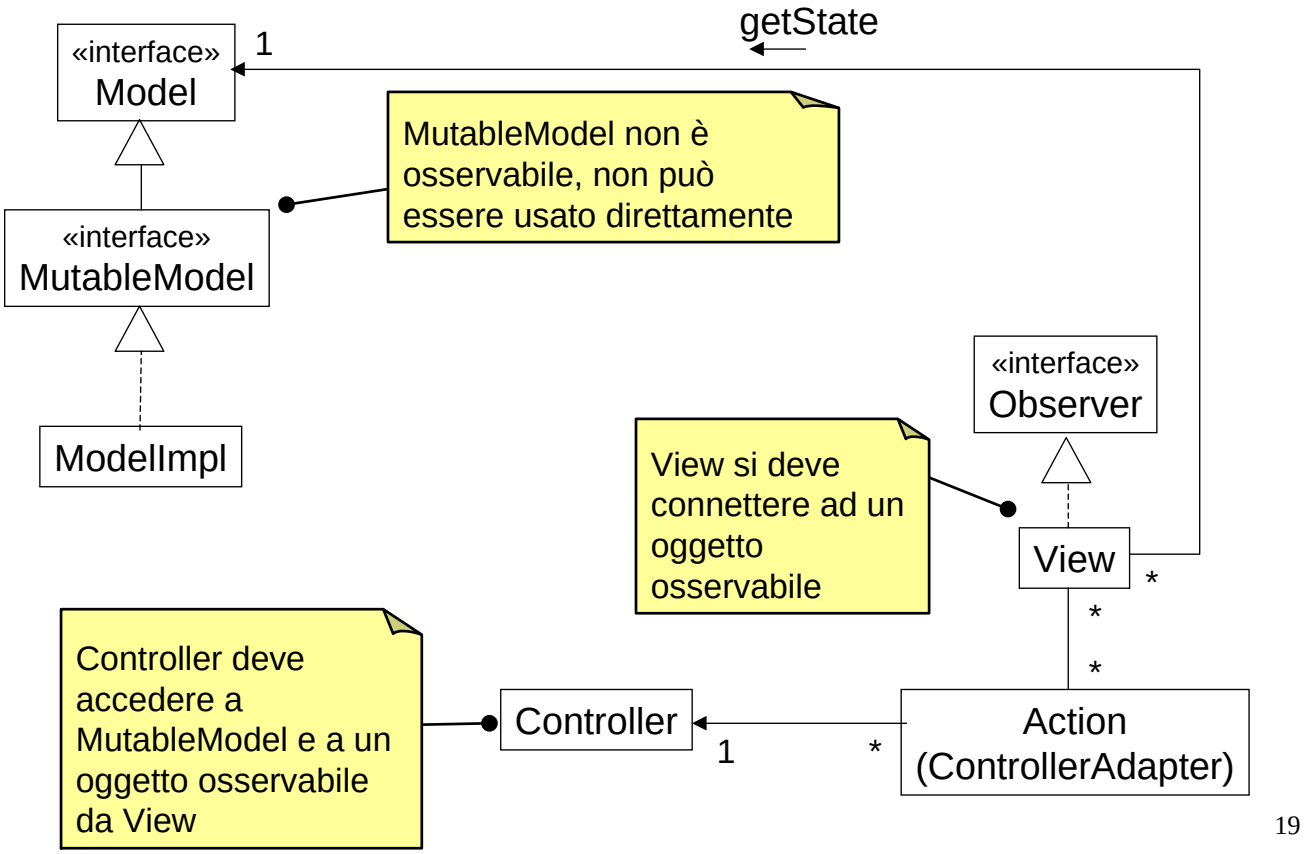
MVC: i dettagli un passo per volta



MVC: i dettagli un passo per volta



MVC: i dettagli un passo per volta



MVC: i dettagli un passo per volta