MODELLAZIONE CON UML

Prof. Mariacarla Staffa

OVERVIEW

- *Introduzione*
  - *Astrazione e modellazione*
  - *Decomposizione*
  - *Gerarchia*
- *Introduzione alla notazione UML*
  - *Diagrammi dei casi d'uso*
  - *Diagrammi delle classi*
  - *Diagrammi delle sequenze*
  - *Diagrammi degli stati*
  - *Diagrammi delle attività*

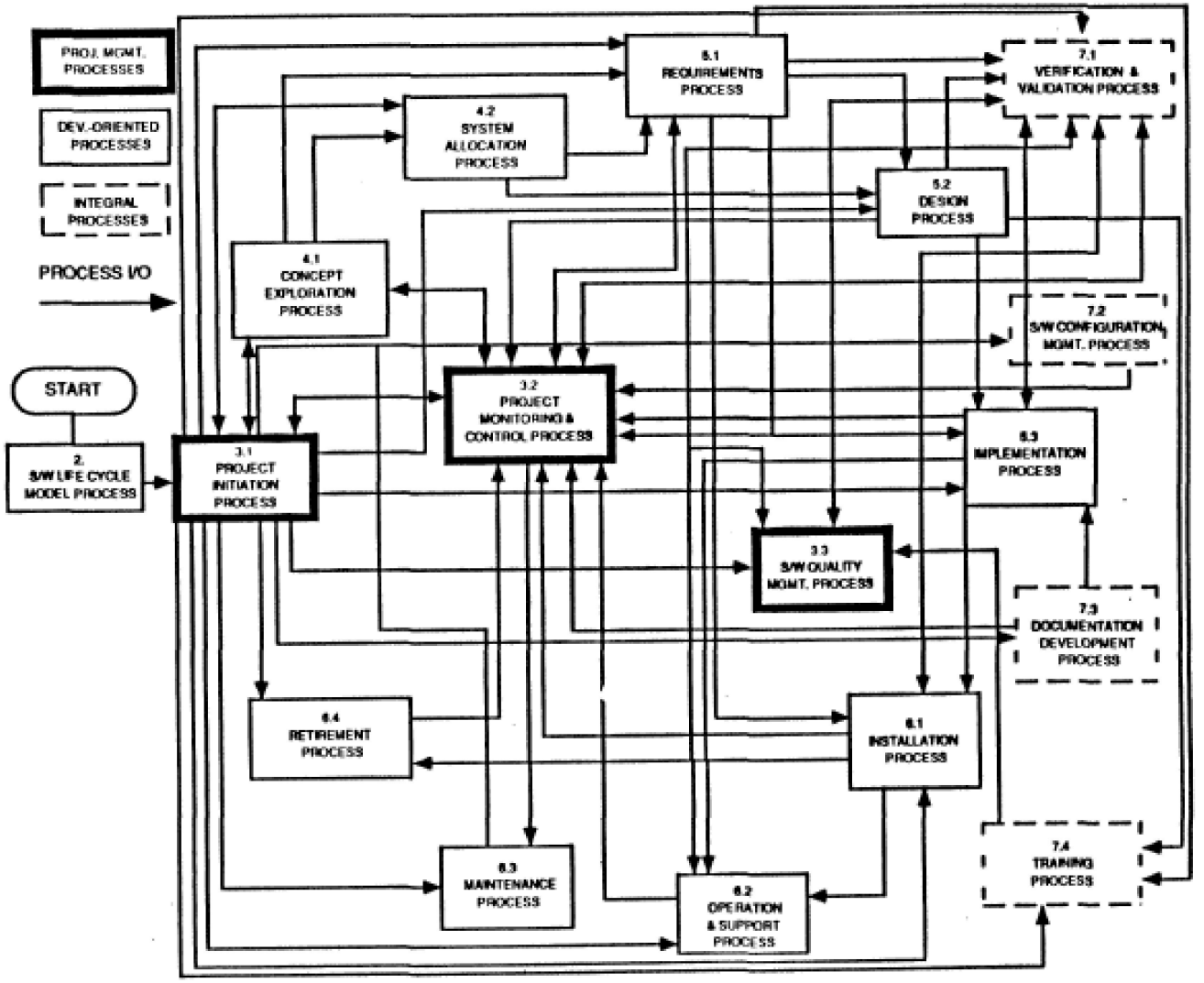
GESTIRE LA COMPLESSITÀ

Astrazione e modellazione

Decomposizione

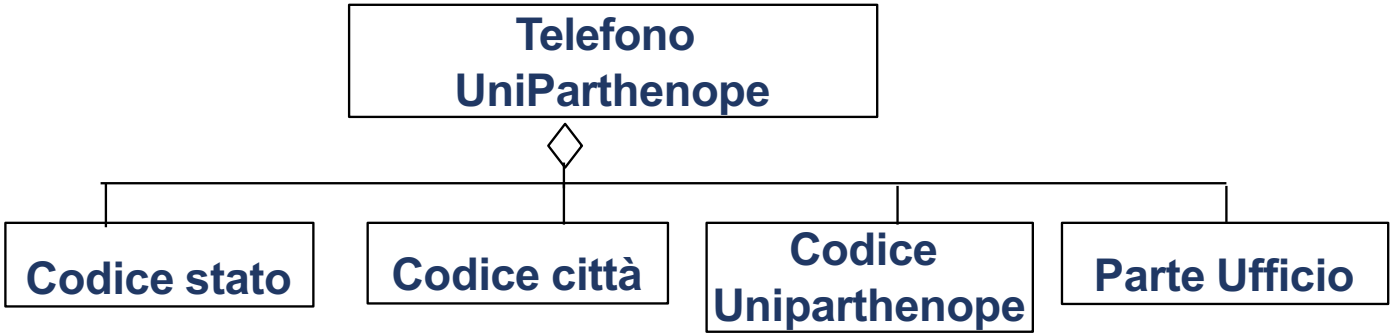
Gerarchia

QUAL È IL PROBLEMA CON QUESTO DISEGNO?



ASTRAZIONE

- I sistemi complessi sono difficili da capire
  - Fenomeno 7 +- 2
    - La nostra memoria a breve termine non può memorizzare più di 7 +- 2 pezzi allo stesso tempo -> limite del cervello
    - Telefono Uniparthenope: 390815476543
  - Spezzettamento: Collezione di gruppi di oggetti per ridurre la complessità
    - 4 pezzi: Codice paese, codice città, codice Uniparthenope, Ufficio



ASTRAZIONE

- L'astrazione consente di ignorare dettagli inutili
- Due definizioni di astrazione
  - *Un processo del pensiero dove le idee sono separate dagli oggetti*
    - *Astrazione come attività*
  - *Idea risultante di un processo di pensiero dove un'idea è stata separata da un oggetto*
    - *Astrazione come entità*
- Le idee possono essere espresse dai **modelli**

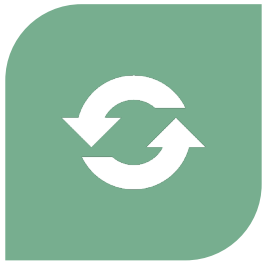
MODELLO

- Un modello è un'astrazione di un sistema
  - Un sistema che non esiste più
  - Un sistema esistente
  - Un sistema futuro da costruire

DESCRIZIONE DEI SISTEMI SOFTWARE CON I MODELLI

MODELLO AD OGGETTI: QUAL È LA STRUTTURA DEL SISTEMA?

MODELLO FUNZIONALE: QUALI SONO LE FUNZIONI DEL SISTEMA?

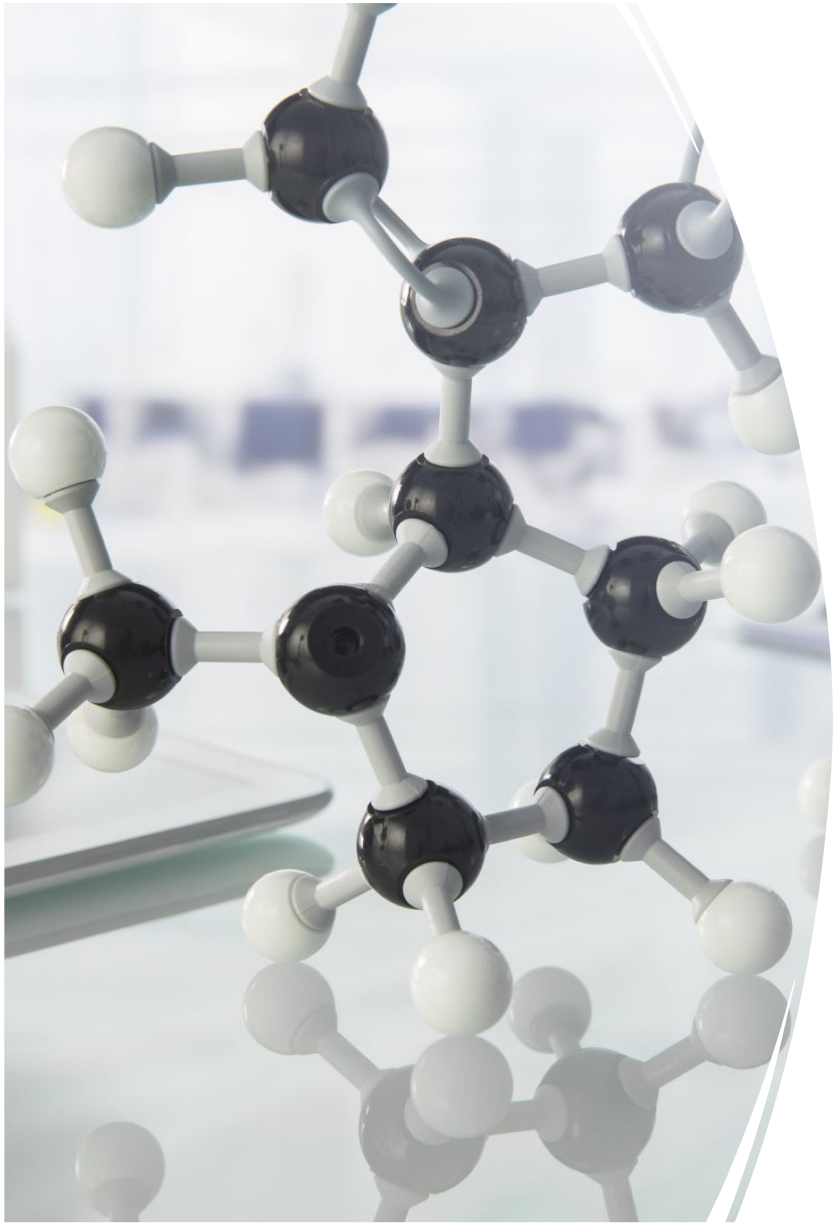


MODELLO DINAMICO: IL SISTEMA COME REAGISCE AGLI EVENTI ESTERNI?

MODELLO DEL SISTEMA: MODELLO AD OGGETTI + MODELLO FUNZIONALE + MODELLO DINAMICO

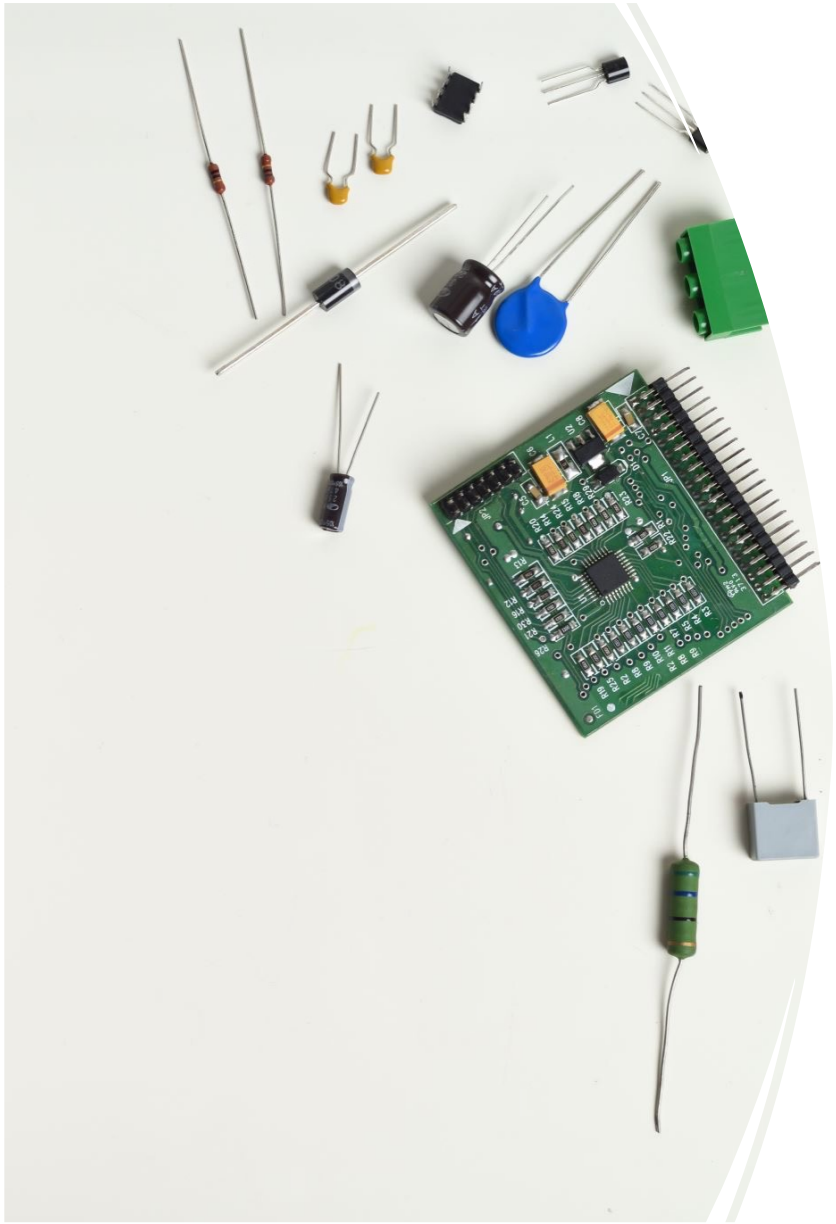
GESTIRE LA COMPLESSITÀ: DECOMPOSIZIONE

- Una tecnica usata per padroneggiare la complessità ('*divide and conquer*')
- Due tipi di decomposizione
  - *Decomposizione funzionale*
  - *Decomposizione orientata agli oggetti*



GESTIRE LA COMPLESSITÀ: DECOMPOSIZIONE FUNZIONALE

- Decomposizione funzionale
  - Il sistema è decomposto in moduli
  - Ogni modulo è una funzione principale nel dominio applicativo
  - I moduli possono essere decomposti in moduli più piccoli



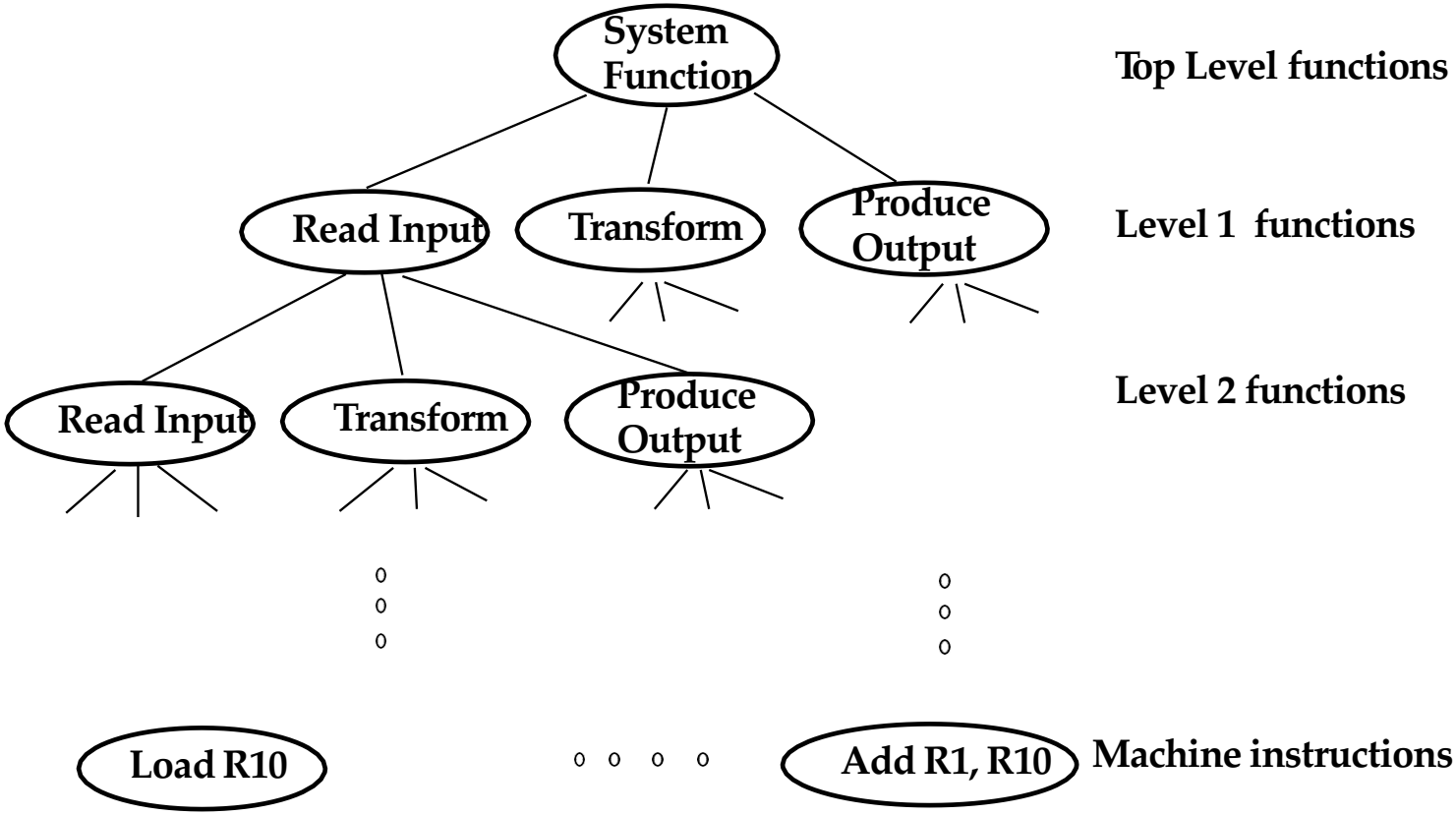
DECOMPOSIZIONE A OGGETTI

- Decomposizione orientata agli oggetti
  - Il sistema è decomposto in classi ('oggetti')
  - Ogni classe è un'entità principale nel dominio applicativo
  - Le classi possono essere decomposte in classi più piccole

Decomposizione orientata agli oggetti vs funzionale

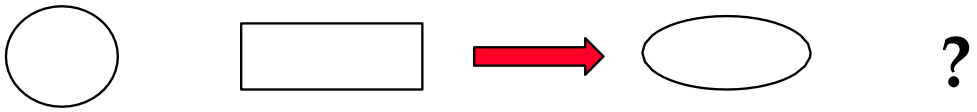
Quale quella giusta?

DECOMPOSIZIONE FUNZIONALE

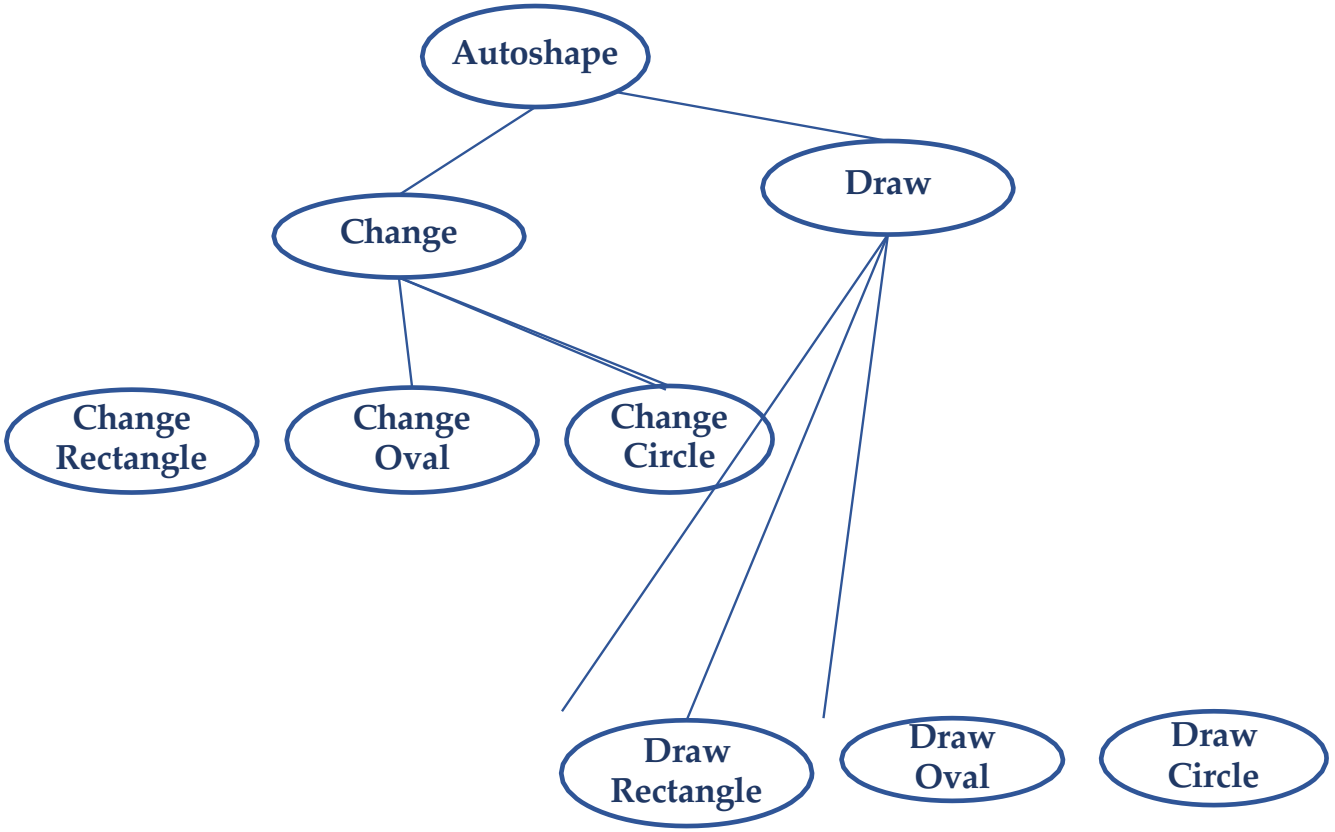


DECOMPOSIZIONE FUNZIONALE

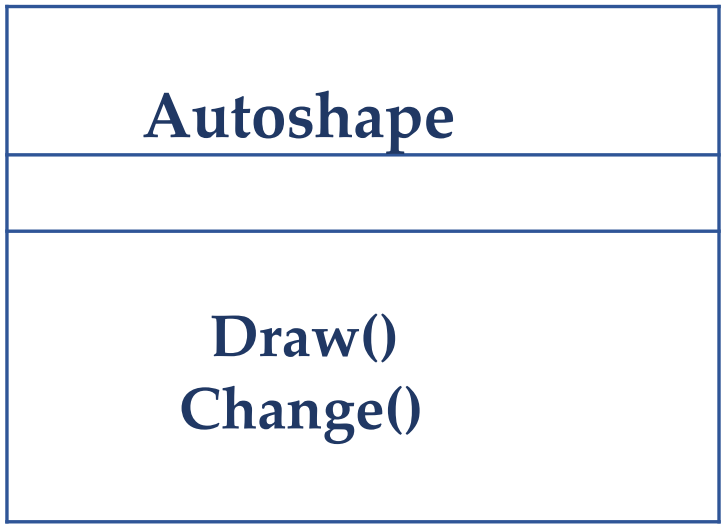
- La funzionalità è distribuita su tutto il sistema
- Il manutentore deve capire l'intero sistema per poter fare anche una singola modifica
- Conseguenza
  - Il codice sorgente è difficile da capire
  - Il codice sorgente è complesso e impossibile da manutenere
  - L'interfaccia utente spesso è scomoda e non intuitiva
- Esempio: *Forme* di Powerpoint
  - Come possiamo cambiare un quadrato in un cerchio?



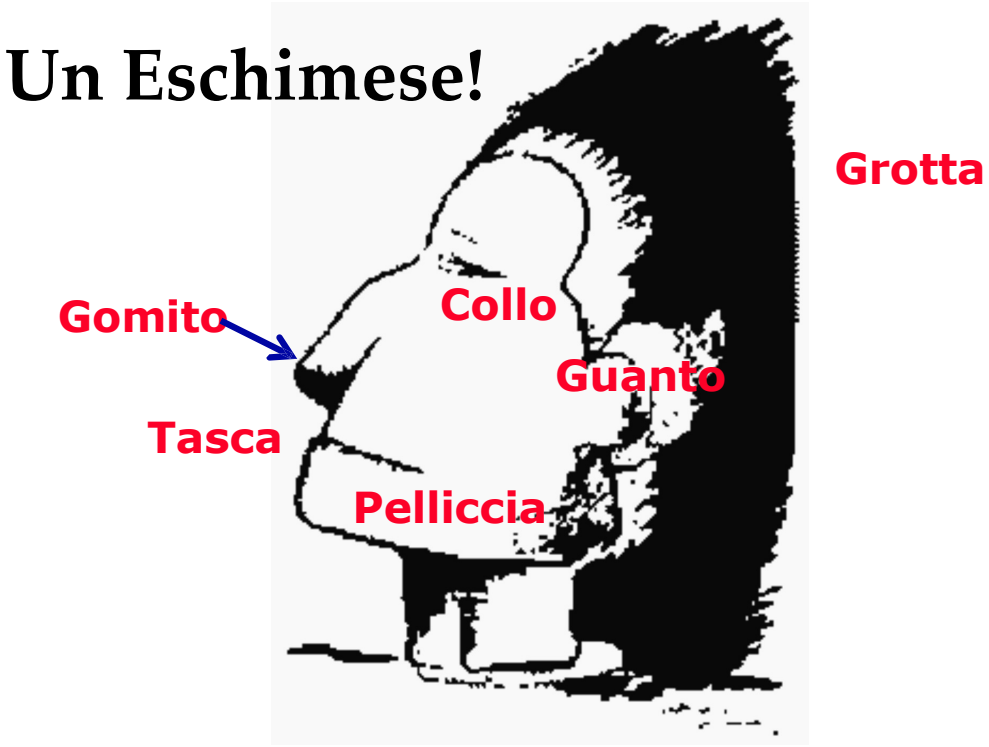
DECOMPOSIZIONE FUNZIONALE



VISTA ORIENTATA AGLI OGGETTI



COSA È QUESTO ?

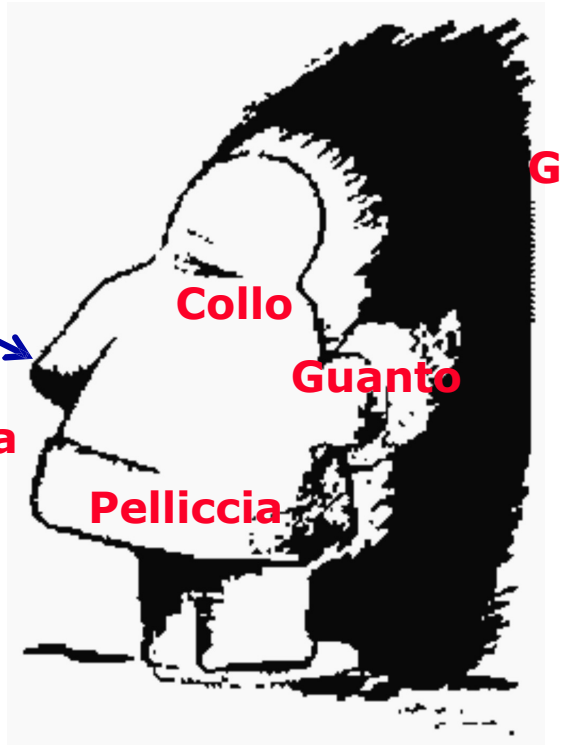


COSA È QUESTO ?

Un volto!

COSA È QUESTO ?

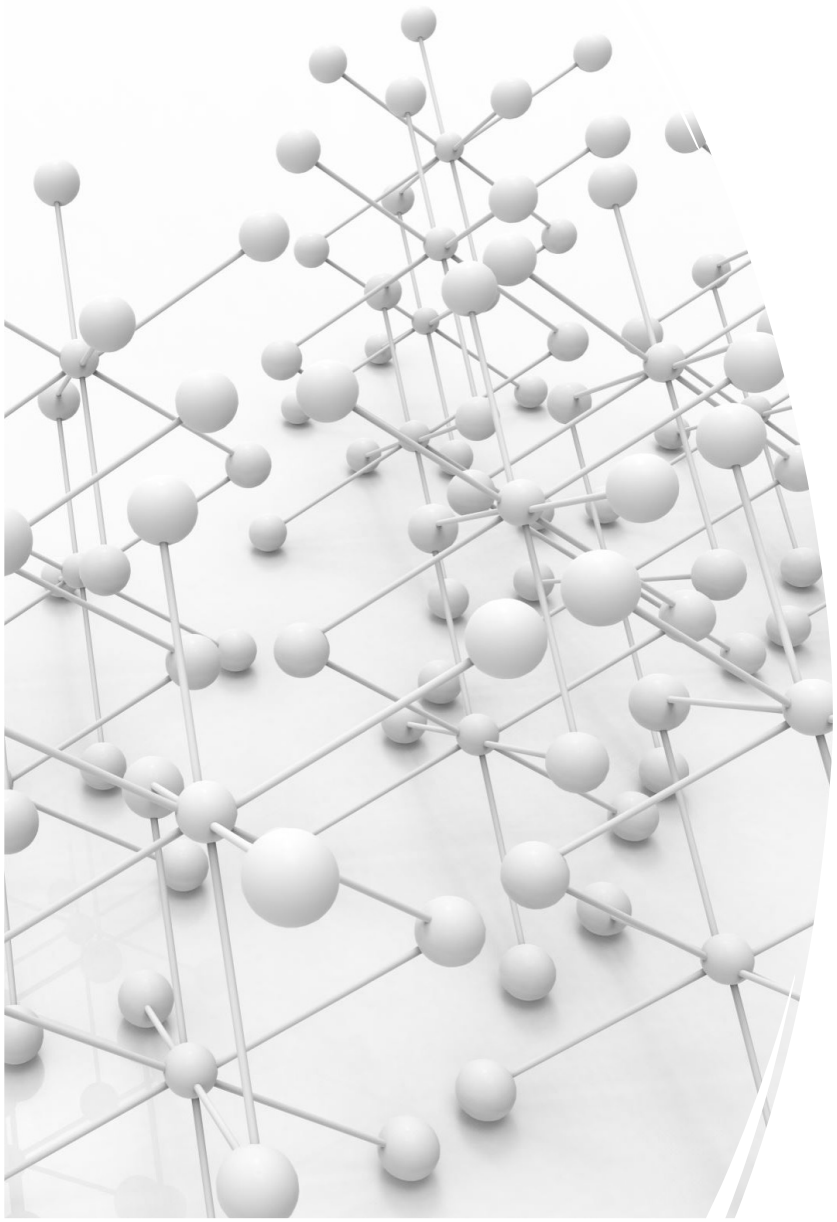
Un Eschimese!



Un volto!

IDENTIFICAZIONE DELLE CLASSI

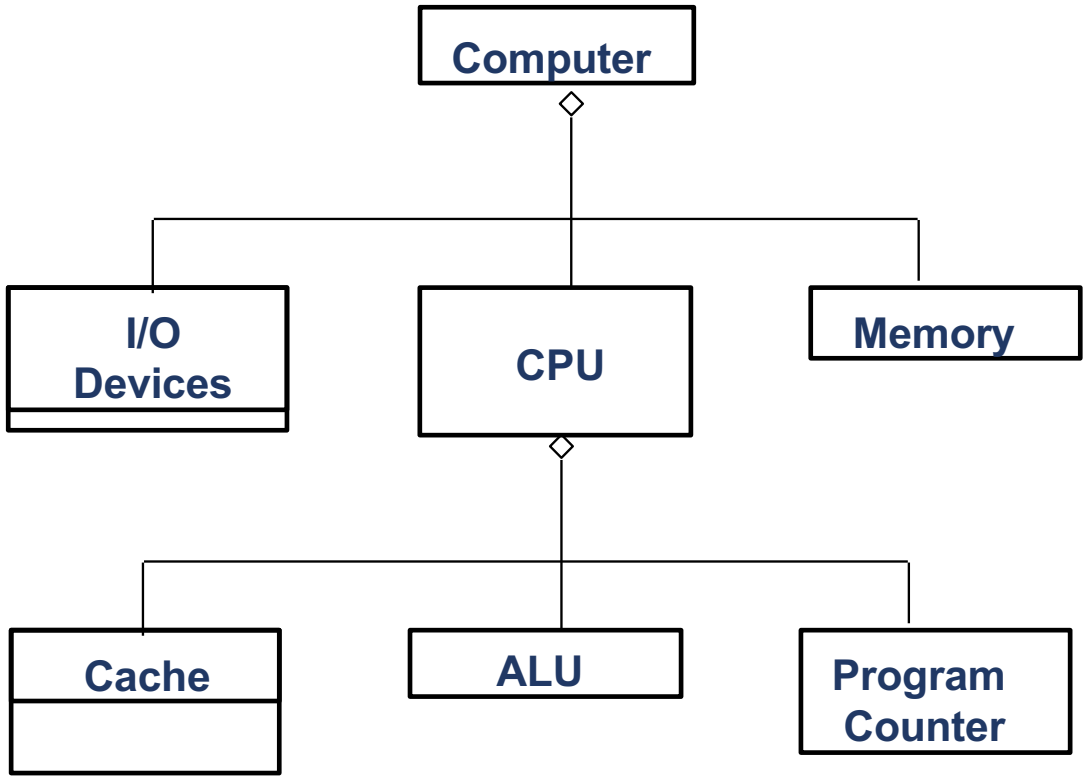
- Assunzioni di base
  - Possiamo trovare le classi per un nuovo sistema software
    - **Greenfield Engineering**
  - Possiamo identificare le classi in un sistema esistente
    - **Reengineering**
  - Possiamo creare un'interfaccia basata su una classe ad un sistema esistente
    - **Interface Engineering**



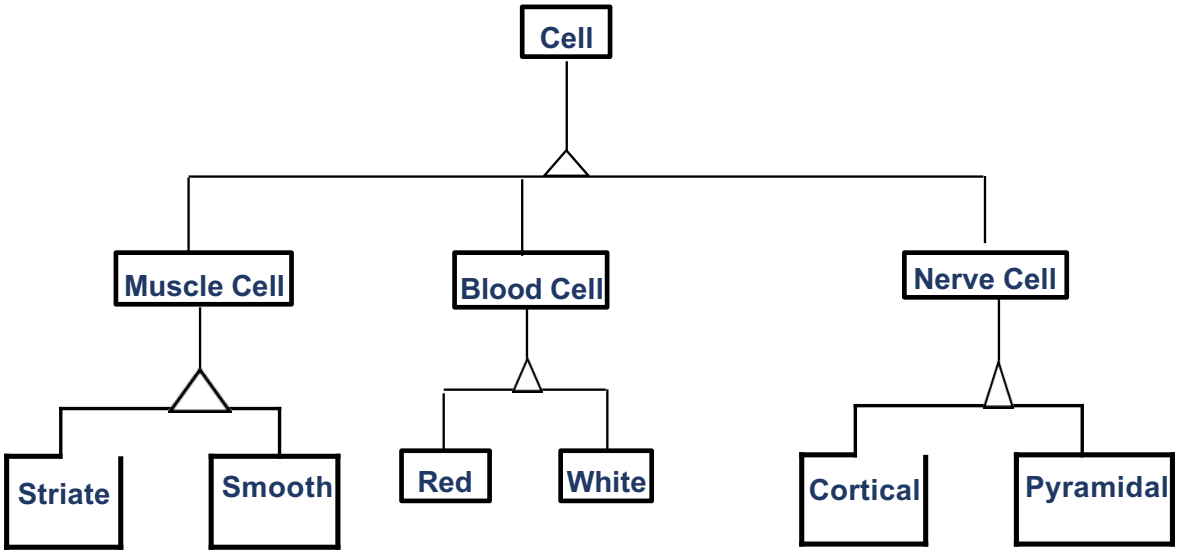
GERARCHIA

- Fin qui abbiamo le astrazioni
  - Ci hanno condotto a classi e oggetti
  - Pezzi
- Un altro modo per gestire la complessità è fornire relazioni tra questi pezzi
- Una delle relazioni più importanti sono le gerarchie
- Due gerarchie speciali
  - La gerarchia '*parte di*'
  - La gerarchia '*è tipo di*'

GERARCHIA 'PARTE DI'



GERARCHIA 'È TIPO DI'



A CHE PUNTO SIAMO?

- Tre modi per trattare la complessità
- Astrazione, decomposizione, gerarchia
- La decomposizione orientata agli oggetti è utile
  - Sfortunatamente, a seconda dello scopo del sistema, possono essere trovati oggetti diversi
- Come possiamo procedere nel modo giusto?
  - Iniziamo con una descrizione delle funzionalità di un sistema
  - Descriviamo poi la sua struttura
- Organizzare le attività di sviluppo
  - Ciclo di vita del software

I MODELLI DEVONO ESSERE FALSIFICABILI

- Karl Popper (‘Conoscenza oggettiva’)
  - Non c’è verità assoluta quando si cerca di capire la realtà
  - Si possono solo costruire teorie che sono ‘vere’ fino a quando qualcuno trova un controesempio
- **Falsificazione**: l’atto di confutare una teoria o ipotesi
- La verità di una teoria non è mai certa. Dobbiamo usare frasi come:
  - ‘a nostro giudizio’, ‘allo stato dell’arte’
- Nell’ingegneria del software ciascun modello è una teoria
  - Costruiamo modelli e cerchiamo di trovare dei controesempi mediante
    - Validazione dei requisiti, test interfaccia utente, revisione della progettazione, testing del codice sorgente, testing del sistema, ecc.
- **Testing**: l’atto di confutare un modello

CONCETTI E FENOMENI

- Fenomeno
  - Un oggetto nel mondo di un dominio così come lo percepiamo
    - Esempi: questa lezione alle 10:05, il mio orologio nero
- Concetto
  - Descrive le proprietà comuni di un fenomeno
    - Esempio: tutte lezioni di ingegneria del software
    - Esempio: tutti gli orologi neri
- Un concetto è una 3-tupla:
  - Nome: il nome distingue il concetto da altri concetti
  - Scopo: proprietà che determinano se un fenomeno è membro di un concetto
  - Membri: l'insieme di fenomeni che sono parte del concetto

CONCETTI, FENOMENI, ASTRAZIONI E MODELLAZIONE

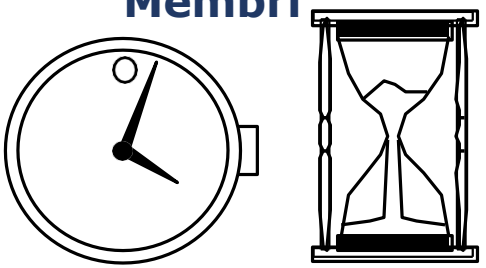
Nome

Orologio

Scopo

Un dispositivo che misura il tempo

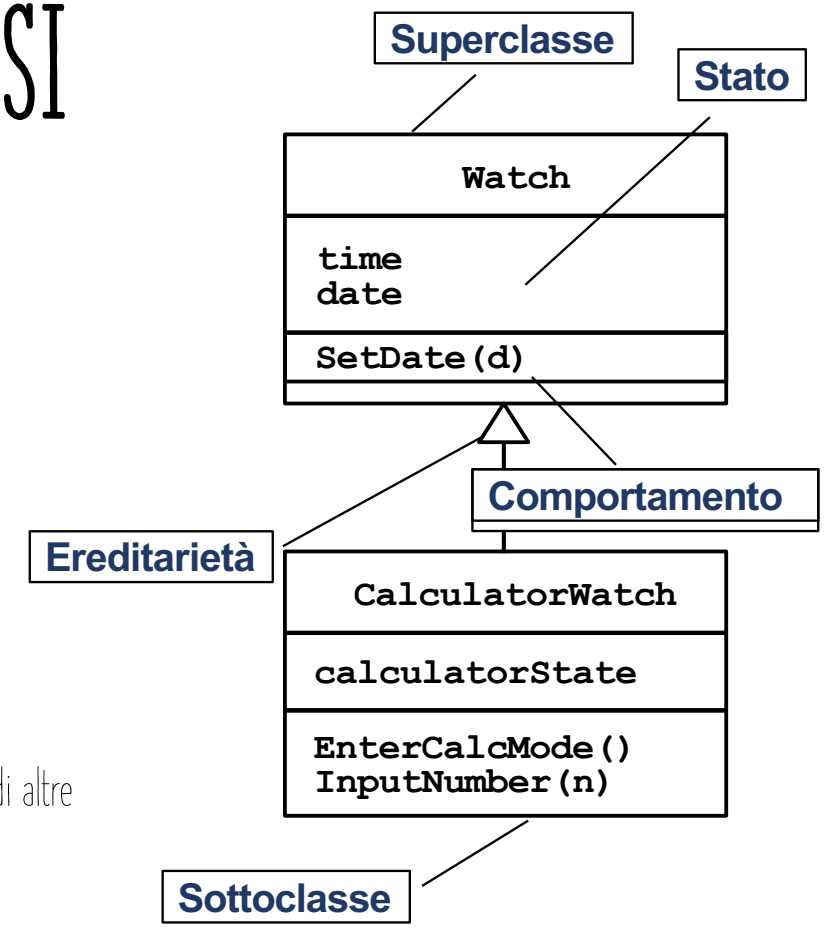
Membri

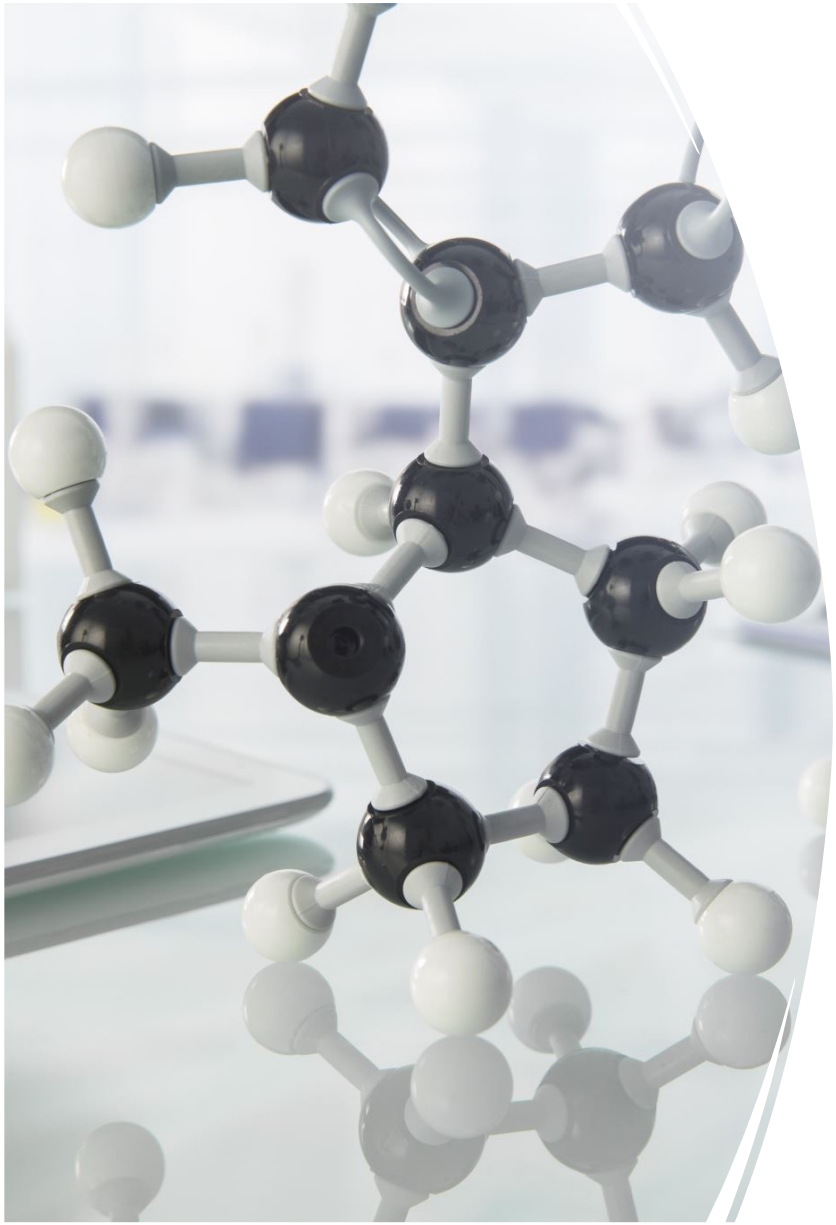


- **Astrazione** definizione
  - Classificazione di fenomeni in concetti
- **Modellazione** definizione
  - Sviluppo di astrazioni per rispondere a domande specifiche su un insieme di fenomeni ignorando dettagli irrilevanti

TIPI DI DATI ASTRATTI E CLASSI

- Tipo di dato astratto
  - Un tipo la cui implementazione è nascosta al resto del sistema
- Classe
  - Un'astrazione nel contesto dei linguaggi orientati agli oggetti
  - Una classe incapsula stato e comportamento
    - Esempio: Orologio
- Differentemente dai tipi di dati astratti, le sottoclassi possono essere definite in termini di altre classi usando l'ereditarietà
  - Esempio: CalculatorWatch





SISTEMI

- Un sistema è un insieme organizzato di parti comunicanti
  - Sistema naturale: un sistema il cui scopo ultimo non è noto
  - Sistema ingegnerizzato: Un sistema che è progettato e costruito da ingegneri per uno scopo sp
- Le parti del sistema possono essere considerate a loro volta sistemi
  - Chiamate sottosistemi
- Esempi di sistemi naturali: Universo, pianeta terra, oceano
- Esempi di sistemi ingegnerizzati:
- Aeroplano, orologio, GPS
- Esempi di sottosistemi: Motore aereo, batteria, satellite

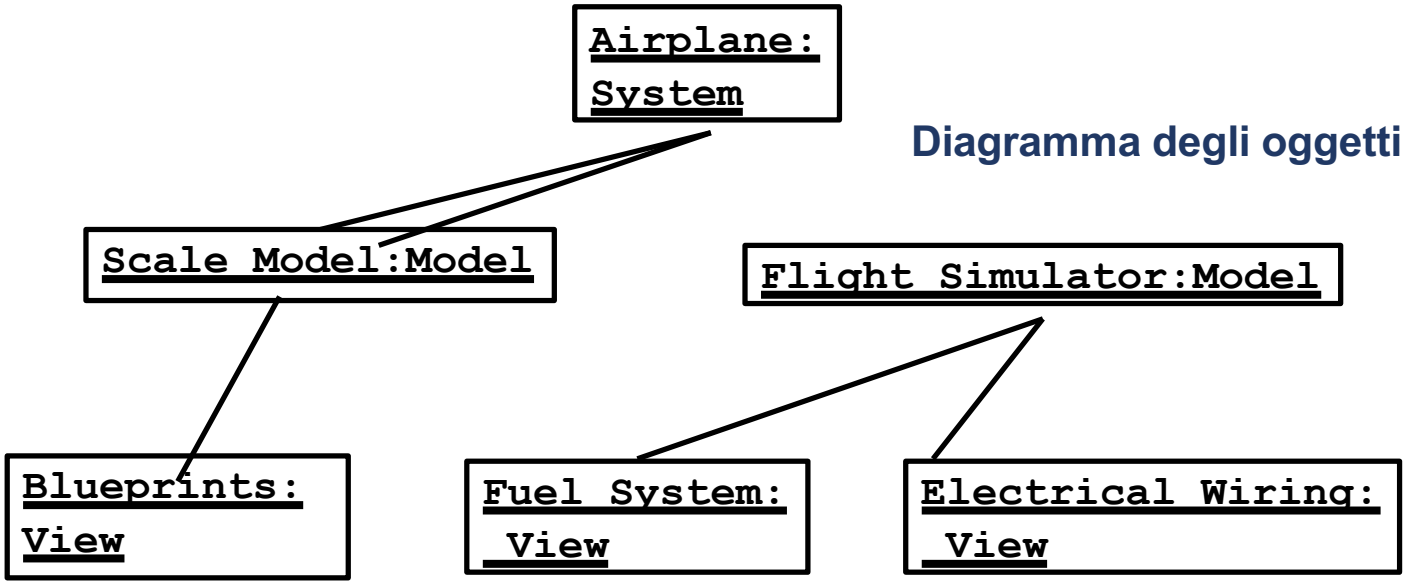
SISTEMI, MODELLI E VIEW

- Un modello è un'astrazione che descrive un sistema o sottosistema
- Una view raffigura aspetti selezionati di un modello
- Una notazione è un insieme di regole grafiche o testuali per illustrare modelli e view
  - Notazioni formali
- Sistema: aeroplano
- Modelli:
  - simulatore di volo
  - Modello in scala
- View:
  - Planimetria dei componenti di un aereo
  - Diagramma dei cavi elettrici, sistema del carburante, onda sonora creata dall'aeroplano

SISTEMI, MODELLI E VIEW (UML)

Diagramma delle classi

Diagramma degli oggetti

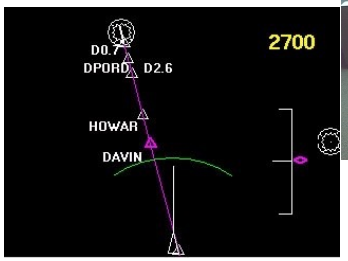


DOMINIO DELL'APPLICAZIONE VS DOMINIO DELLA SOLUZIONE

- Dominio dell'applicazione (Analisi)
  - L'ambiente in cui il sistema sta funzionando
- Dominio della soluzione (Progettazione, Implementazione)
  - Le tecnologie usate per costruire il sistema
- Entrambi i domini contengono astrazioni che possiamo usare per la costruzione del modello del sistema

MODELLAZIONE ORIENTATA AGLI OGGETTI

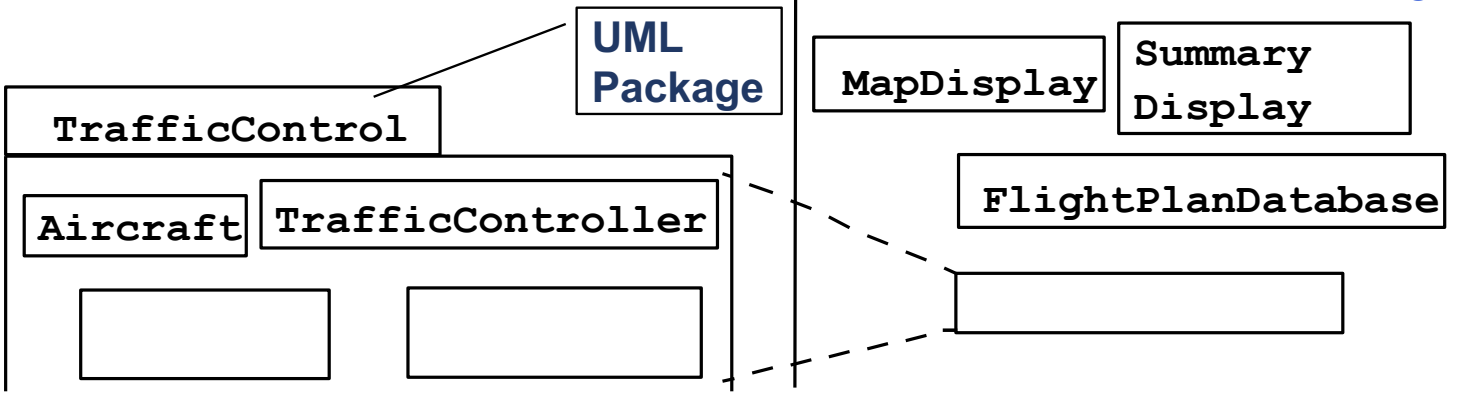
Dominio applicazione
(Fenomeni)



Dominio soluzione
(Fenomeni)

Application domain (Concetti) *(Analisi)*

System Model (Concetti) *(Design)*



COSA È L'UML?

- UML (Unified Modeling Language)
  - Standard non proprietario per modellare sistemi software
  - Convergenza delle notazioni usate nei metodi orientati agli oggetti
    - OMT (James Rumbaugh e colleghi)
    - Booch (Grady Booch)
    - OOSE (Ivar Jacobson)
  - Versione corrente 2.5
    - Info sul portale OMG http://www.uml.org/
  - Tool commerciali: rational (IBM), Together (Borland), Visual Architect (Business processes, BCD)
  - Tool open surce: ArgoUML, StarUML, Umbrello
  - Commerciale e Open source: PoseidonUML (Gentleware)

25 year at GE Research, where he developed OMT, joined (IBM) Rational in 1994, CASE tool OMTool

At Ericsson until 1994, developed use cases and the CASE tool Objectory, at IBM Rational since 1995

Developed the Booch method ("clouds"), ACM Fellow 1995, and IBM Fellow

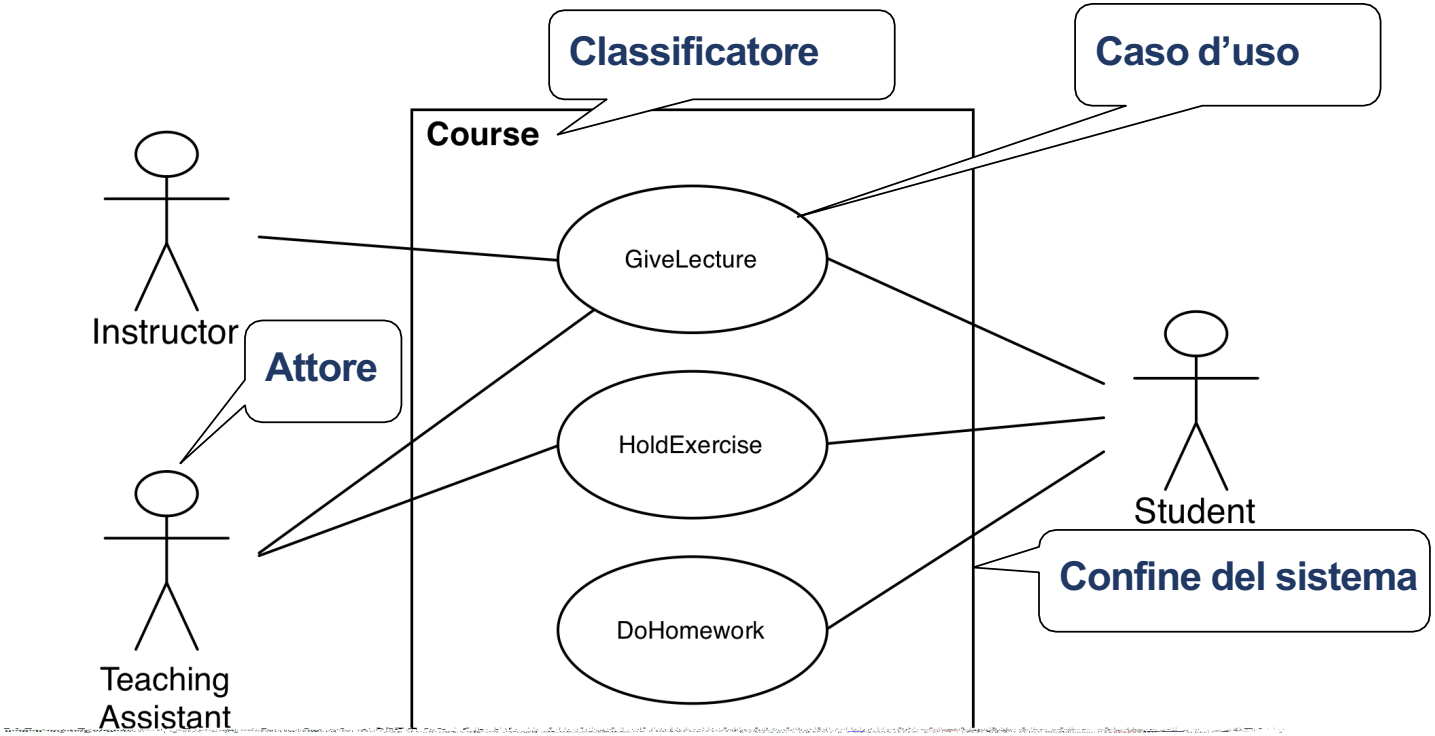
DIAGRAMMI UML

Diagrammi dei casi d'uso	Descrivono il comportamento funzionale del sistema come sono visti dagli utenti
Diagrammi delle classi	Descrivono la struttura statica del sistema: oggetti, attributi, associazioni
Diagrammi delle sequenze	Descrivono il comportamento dinamico tra gli oggetti del sistema
Diagrammi degli stati	Descrivono il comportamento dinamico di un singolo oggetto
Diagrammi delle attività	Descrivono il comportamento dinamico di un sistema, in particolare il flusso di lavoro

UML: CONVENZIONI DI BASE

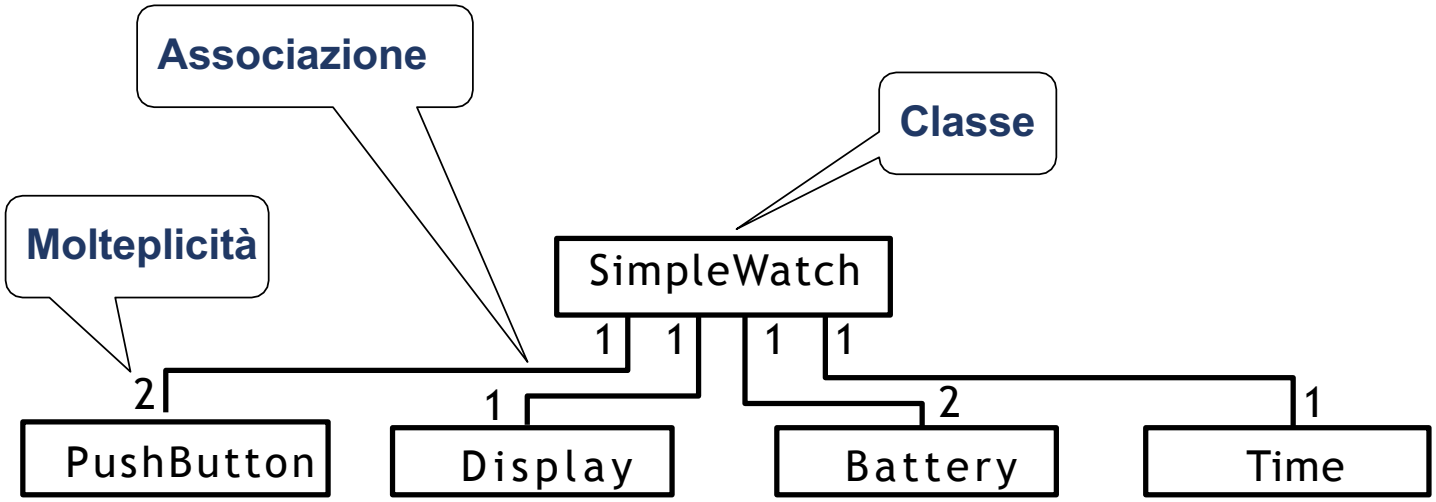
- Tutti i diagrammi UML denotano grafi di nodi e vertici
  - I nodi sono entità disegnati come rettangoli o ovali:
  - I rettangoli denotano classi o istanze 
  - Gli ovali denotano funzioni 
- I nomi delle classi non sono sottolineati
  - SimpleWatch
  - Firefighter
- I nomi delle istanze sono sottolineati
  - myWatch:SimpleWatch
  - Joe:Fireighter
- Un arco tra due nodi indica una relazione tra le entità corrispondenti

DIAGRAMMI DEI CASI D'USO



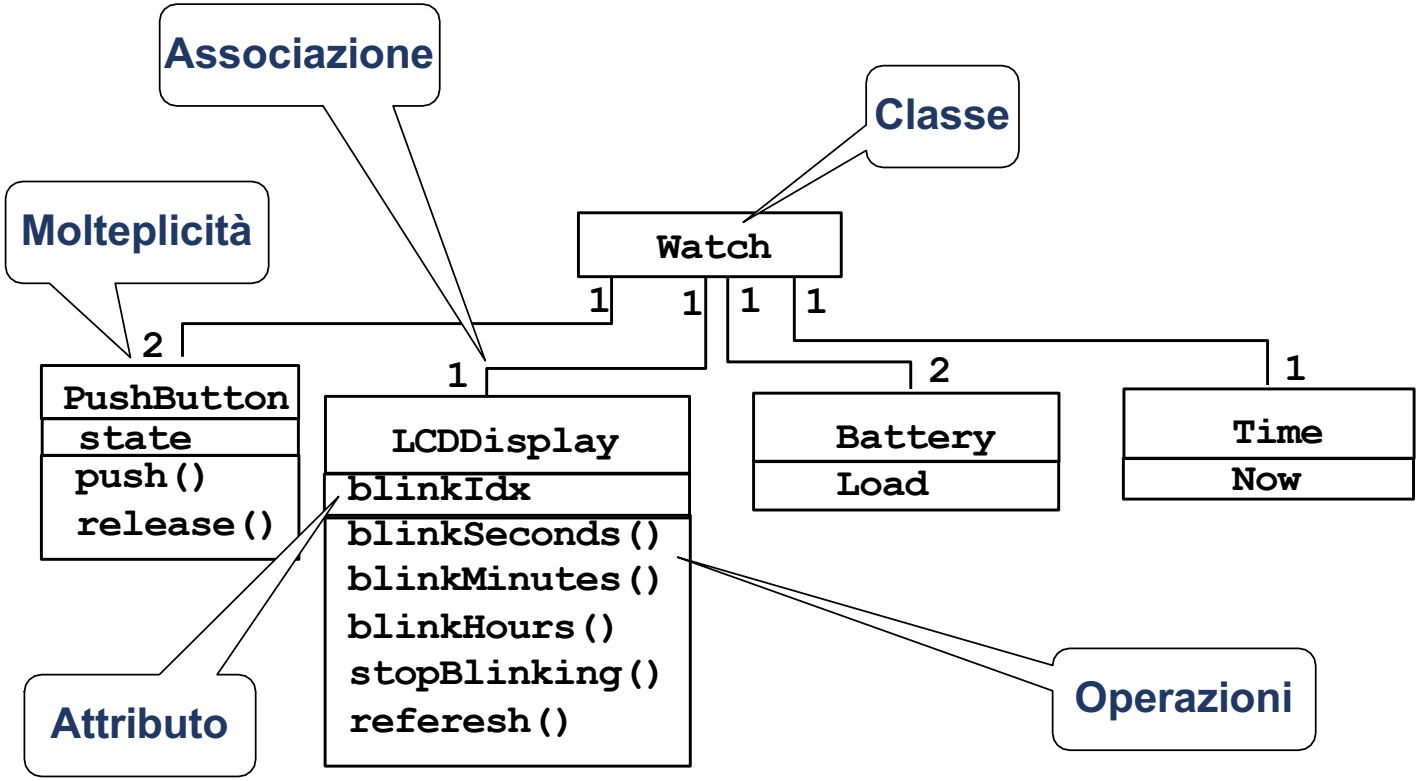
I diagrammi dei casi d'uso rappresentano le funzionalità del sistema dal punto di vista dell'utente

DIAGRAMMI DELLE CLASSI

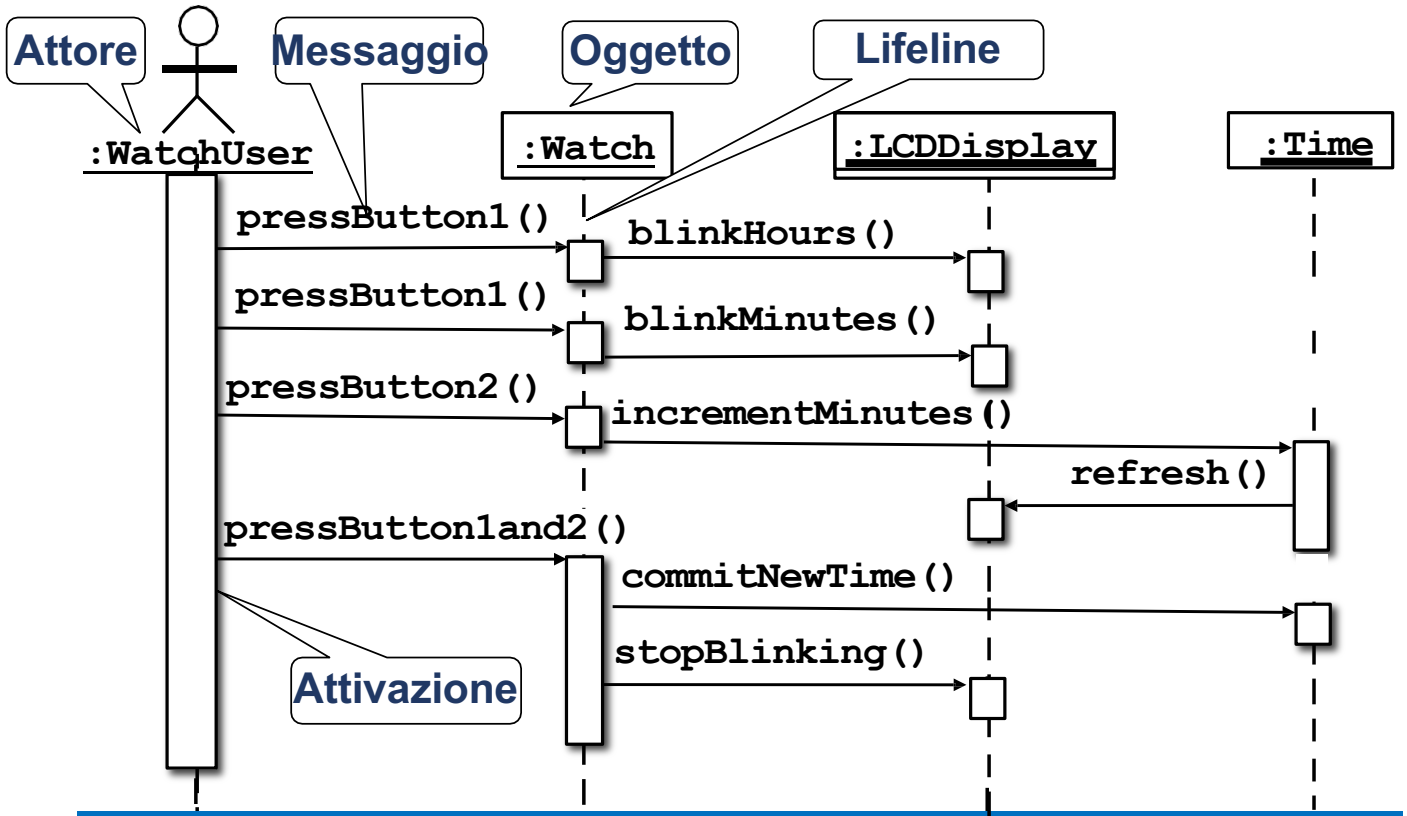


I diagrammi delle classi rappresentano la struttura del sistema

DIAGRAMMI DELLE CLASSI

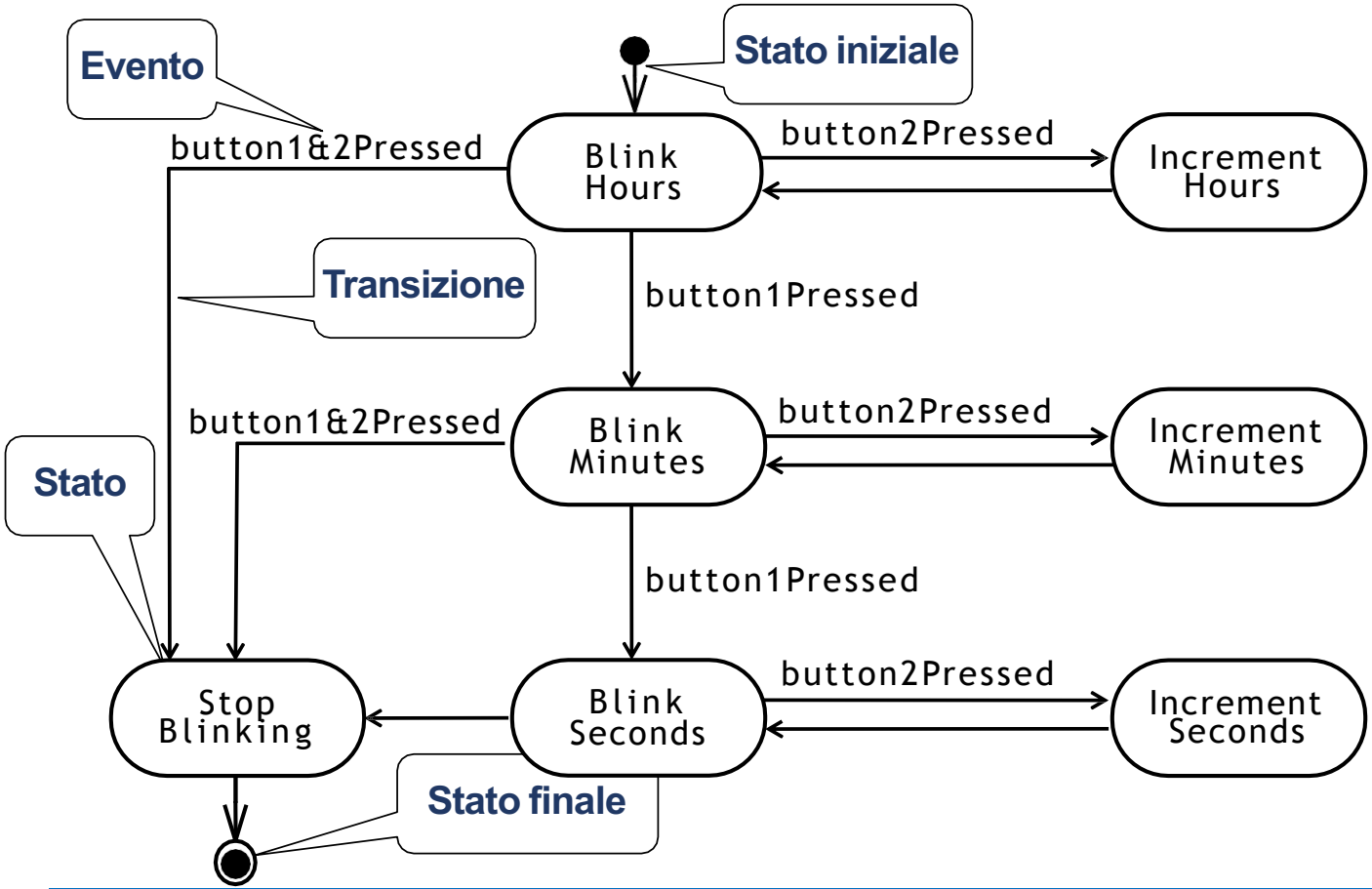


DIAGRAMMI DELLE SEQUENZE



I diagrammi delle sequenze rappresentano il comportamento di un Sistema come messaggi (*interazioni*) tra oggetti differenti

DIAGRAMMI DEGLI STATI



Rappresenta il comportamento di un singolo oggetto con un comportamento dinamico interessante

COSA FARE PRIMA? LA CODIFICA O LA MODELLAZIONE?

- Tutto dipende …
- **Forward engineering**
  - Creazione del codice da un modello
  - Inizia con la modellazione
  - Progetti Greenfield
- **Reverse engineering**
  - Creazione di un modello da codice esistente
  - Interfaccia o progetti di reingegnerizzazione
- **Roundtrip engineering**
  - Si sposta costantemente tra forward e reverse engineering
  - Progetti di reingegnerizzazione
  - Utile quando i requisiti, le tecnologie e la schedulazione cambiano frequentemente

RIEPILOGO DELLA NOTAZIONE UML BASE

- UML fornisce un'ampia varietà di notazioni per modellare molti aspetti dei sistemi software
- Ci siamo concentrati su poche notazioni
  - Modello funzionale: diagrammi dei casi d'uso
  - Modello ad oggetti: diagramma delle classi
  - Modello dinamico: diagrammi delle sequenze, degli stati