Modelli di sistema e UML

Indice

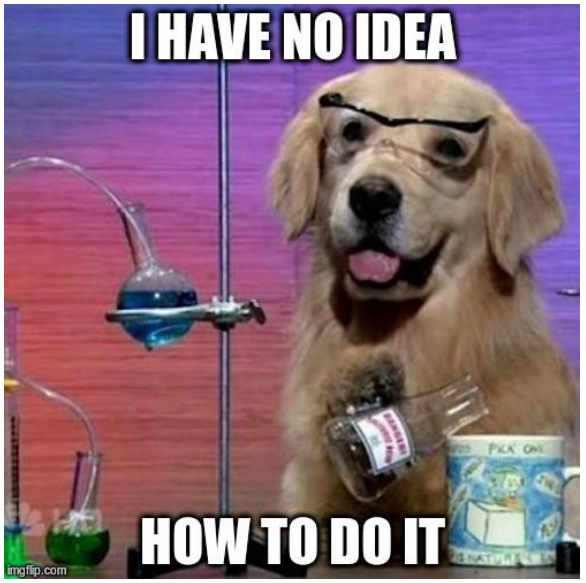
- Introduzione
- Unified Modeling Language (UML)
  - Diagrammi dei casi d’uso
  - Diagrammi di attività
  - Diagrammi di stato
  - Diagrammi di sequenza
  - Diagrammi di classe
- Architettura guidata da modelli (Model-Driven Architecture)

Riferimenti

- Ian Sommerville. *Ingegneria del Software.* Capitolo 5 (10a edizione).
- Specifica UML 2.5: http://www.omg.org/spec/UML/2.5/PDF
- Slide di “Ingegneria del Software” del Prof. Fabrizio Luglio

Problema da risolvere

Poco dopo…



Modellazione dei sistemi

- **La modellazione dei sistemi (system modelling) è il processo che sviluppa modelli astratti di un sistema, dove ogni modello rappresenta una differente vista o prospettiva del sistema.**
- Oggi per modellazione dei sistemi di solito s'intende la rappresentazione di un sistema utilizzando qualche tipo di notazione grafica, che oramai è quasi sempre basata sulla notazione UML (Unified Modeling Language)
- La modellazione dei sistemi aiuta a comprendere le funzionalità del sistema da sviluppare e i modelli possono essere utilizzati per comunicare con i clienti.

Modelli di sistemi esistenti e nuovi

È possibile sviluppare modelli sia di sistemi esistenti sia di nuovi sistemi:

- **I modelli di un sistema esistente** si usano durante l'ingegneria dei requisiti. Servono a chiarire che cosa fa il sistema esistente e possono essere utilizzati per focalizzare la discussione degli stakeholder sui punti di forza e di debolezza del sistema.
- **I modelli di un nuovo sistema** si usano durante l'ingegneria dei requisiti per descrivere con maggior chiarezza i requisiti proposti ad altri stakeholder del sistema. Gli ingegneri utilizzano questi modelli per discutere le proposte di progettazione e per documentare l'implementazione del sistema.

Prospettive del sistema

È possibile sviluppare vari modelli per rappresentare il sistema da differenti prospettive; per esempio:

- Una prospettiva esterna: viene modellato il contesto o l'ambiente in cui opera il sistema.
- Una prospettiva di interazioni: vengono modellate le interazioni tra il sistema e il suo ambiente, o tra i componenti del sistema.
- Una prospettiva strutturale: viene modellata l'organizzazione del sistema o la struttura dei dati elaborati dal sistema;
- Una prospettiva comportamentale: vengono modellati il comportamento dinamico del sistema e le sue risposte agli eventi.

Uso di modelli grafici

Ci sono tre modi in cui i modelli grafici sono comunemente utilizzati:

- Per facilitare la discussione su un sistema proposto o esistente.
  - I modelli possono essere incompleti
- Per documentare un sistema proposto o esistente.
  - I modelli devono essere corretti e descrivere accuratamente il sistema, ma non è necessario che siano completi
- Per fornire una descrizione dettagliata del sistema che può essere utilizzata per generare l'implementazione del sistema.
  - In questo caso i modelli devono essere sia corretti che completi

Unified Modeling Language
UML

Cos'è UML

- Linguaggio universale per creare modelli software
- È indipendente da metodi, tecnologie, produttori
- Sviluppato da Object Management Group (OMG): Hewlett-Packard, IBM, Oracle, Apple, Microsoft, ecc.

Struttura di UML

UML è costituito da:

- **Costituenti fondamentali:** gli elementi, le relazioni e i diagrammi di base
- **Meccanismi comuni:** tecniche comuni per raggiungere specifici obiettivi con UML
- **Architettura:** il modo con cui UML esprime l'architettura del sistema.

Costituenti fondamentali

UML è composto da 3 tipi di costituenti fondamentali:

- **Entità strutturali:** divise in quattro categorie:
  - strutturali (classe, interfaccia), comportamentali (interazioni, stati), di raggruppamento (package) e informative (annotazione)
- **Relazioni:** collegano fra di loro le entità:
  - associazione, dipendenza, generalizzazione, realizzazione.
- **Diagrammi:** I diagrammi UML sono rappresentazioni grafiche **parziali** di un modello UML. Sono viste che consentono di vedere il contenuto del modello da diverse prospettive.
  - Un'entità o una relazione può essere eliminata da un diagramma (per facilitare la comprensione) ma continuare ad esistere nel modello.

Meccanismi comuni

UML definisce 4 meccanismi comuni:

- **Specifiche:** descrizione testuale della semantica di un elemento (classe, relazione...). Completa la descrizione grafica rappresentata dai diagrammi.
- **Ornamenti:** informazioni aggiunte in un diagramma ad un elemento del modello per illustrare un concetto supplementare (visibilità, molteplicità...).
- **Distinzioni comuni:** meccanismi per distinguere tra astrazioni e loro istanze.
- **Meccanismi di estendibilità:** consentono di estendere il linguaggio per esigenze specifiche. Sono di tre tipi: vincoli, stereotipi e valori etichettati.

Meccanismi di estendibilità

- **Vincoli:** sono frasi testuali racchiuse fra parentesi graffe {}. Definiscono una condizione o una regola che si applica ad un elemento di modellazione e che deve risultare sempre vera.
- **Stereotipi:** rappresentano una variazione di un elemento esistente (entità, relazione o altro) che ha la stessa forma ma un diverso scopo. Sono parole racchiuse fra i simboli « e ».
  - L'esempio più noto è «interface»: un'interfaccia ha la stessa rappresentazione di una classe ma lo stereotipo indica che si tratta di una cosa diversa
- **Valori etichettati:** permettono di aggiungere nuove proprietà ad un elemento di modellazione.
  - Sono elenchi di coppie etichetta = valore: {etichetta1=valore1, etichetta2 =valore2, ...}

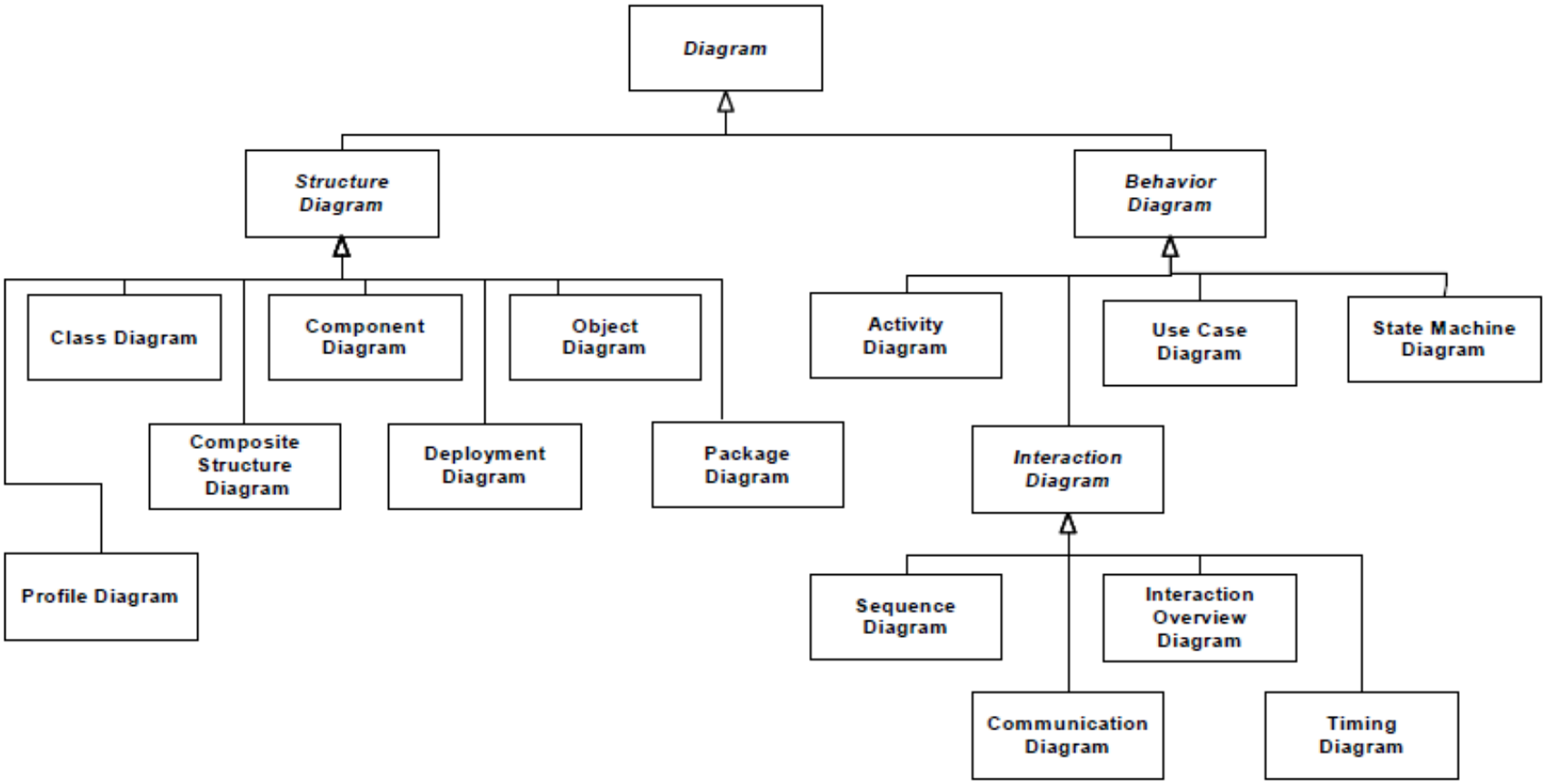
Architettura

- L’architettura di un sistema solitamente viene definita da più viste
- Il concetto di vista non è definito in modo formale dalle specifiche UML. Tuttavia, il concetto di vista è stato storicamente utilizzato dalla maggior parte degli ambienti software di modellazione UML.
- L’approccio più famoso per definire le viste è il "modello 4 + 1"
  - **Vista logica:** mostra le astrazioni chiave nel sistema come oggetti o classi di oggetti.
  - **Vista dei processi:** descrive il comportamento dinamico del sistema.
  - **Vista di implementazione:** descrive la struttura concreta del software che compone il sistema (processi, moduli, librerie, ecc.)
  - **Vista di deployment:** mostra come il software viene suddiviso per lo sviluppo.
  - **Vista dei casi d’uso (+1):** descrive i requisiti del sistema in termini di servizi offerti.

Diagrammi UML e viste

- **Vista logica**
  - Diagrammi di classe, diagrammi di stato, diagrammi di oggetti
- **Vista dei processi**
  - Diagrammi di stato, diagrammi di oggetti
- **Vista di implementazione**
  - Diagrammi dei componenti
- **Vista di deployment**
  - Diagrammi di deployment
- **Vista dei casi d'uso (+1):**
  - Diagrammi dei casi d'uso, diagrammi d'interazione

Tassonomia dei diagrammi UML



Suddivisione dei Diagrammi UML

- **Diagrammi strutturali: r**appresentano le entità di un sistema software e le loro relazioni.
- **Diagrammi comportamentali:** rappresentano il comportamento di un sistema software al trascorrere del tempo.
  - **Diagrammi di interazione:** descrivono il modo in cui le entità interagiscono per generare il comportamento richiesto al sistema software.

Diagrammi UML

L'UML ha 14 tipi di diagrammi. I più usati sono:

- **Diagrammi dei casi d'uso:** mostrano le interazioni tra un sistema e il suo ambiente.
- **Diagrammi di attività:** mostrano le attività coinvolte in un processo o nell'elaborazione dei dati.
- **Diagrammi di stato:** mostrano come il sistema reagisce agli eventi interni ed esterni.
- **Diagrammi di sequenza:** mostrano le interazioni tra attori e sistema e tra i componenti del sistema.
- **Diagrammi di classe:** mostrano le classi del sistema e le loro associazioni.

Unified Modeling Language
UML

Diagrammi dei casi d'uso

Diagrammi dei casi d'uso

- Furono introdotti per supportare la deduzione dei requisiti e ora sono incorporati nella specifica UML.
- Ogni caso d'uso (**ellisse**) rappresenta un task che coinvolge un'interazione esterna con il sistema.
- Gli attori (**omini stilizzati**) possono essere persone o altri sistemi.
- I diagrammi di caso forniscono una semplice panoramica di un'interazione, quindi è necessario fornire maggiori dettagli per capire l'interazione.
  - Questi dettagli possono essere definiti da una semplice descrizione testuale, una descrizione strutturata in una tabella o un diagramma di sequenza

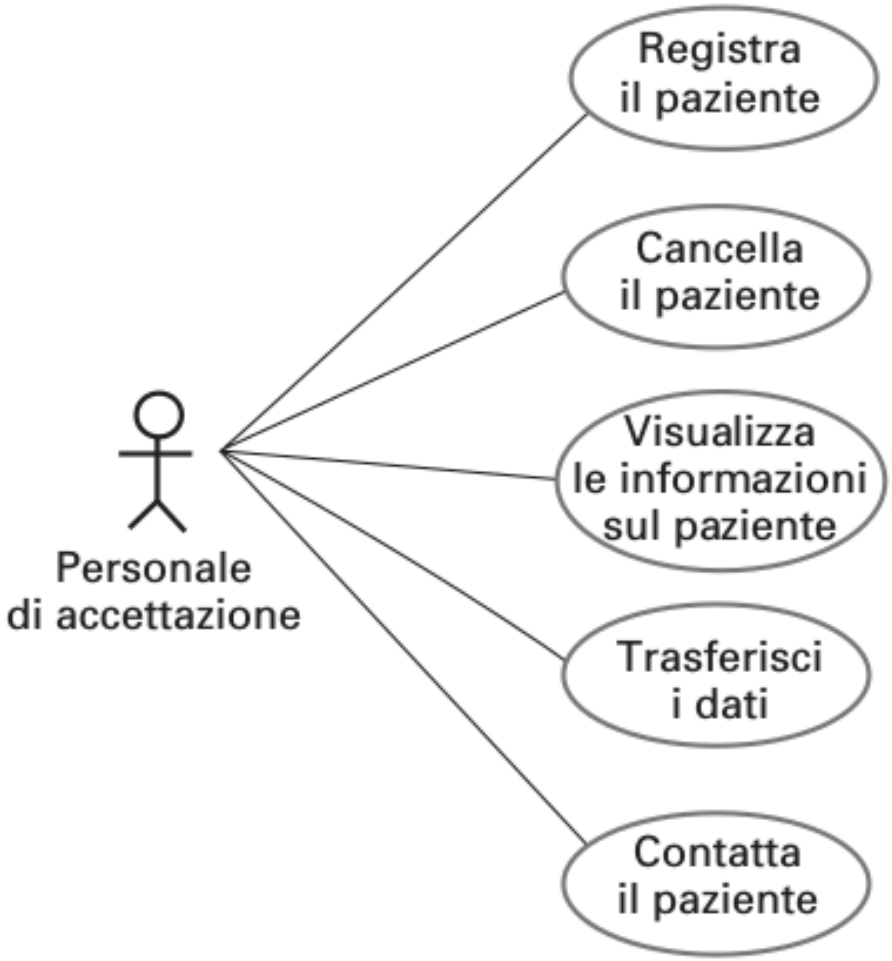
Esempio 1: diagramma dei casi d'uso per il trasferimento dati

- Diagramma dei casi d'uso per un sistema ospedaliero chiamato Mentcare.

Descrizione tabellare del caso d'uso "Trasferimento dei dati"

Sistema Mentcare – Trasferimento dei dati	
Attori	Personale di accettazione, sistema di registrazione dei pazienti (SRP).
Descrizione	Un addetto all'accettazione può trasferire i dati dal sistema Mentcare a un database generale di registrazione dei pazienti che è mantenuto da un'autorità sanitaria. Le informazioni trasferite possono essere informazioni personali aggiornate (indirizzo, numero di telefono ecc.) o una sintesi della diagnosi e della cura dei pazienti.
Dati	Informazioni personali dei pazienti, sintesi della cura.
Stimolo	Comando emesso dal personale di accettazione.
Risposta	Conferma che l'SRP è stato aggiornato.
Commenti	L'addetto all'accettazione deve avere le autorizzazioni appropriate per accedere alle informazioni dei pazienti e all'SRP.

Esempio 2: diagramma dei casi d'uso per il ruolo di 'Personale di accettazione'



Unified Modeling Language
UML

Diagrammi di attività

Diagrammi di attività

- I diagrammi di attività UML possono essere utilizzati per mostrare i processi aziendali in cui si usano i sistemi.
- I diagrammi di attività UML mostrano le attività in un processo e il flusso di controllo da un'attività all'altra.

Diagrammi di attività: notazione (a)

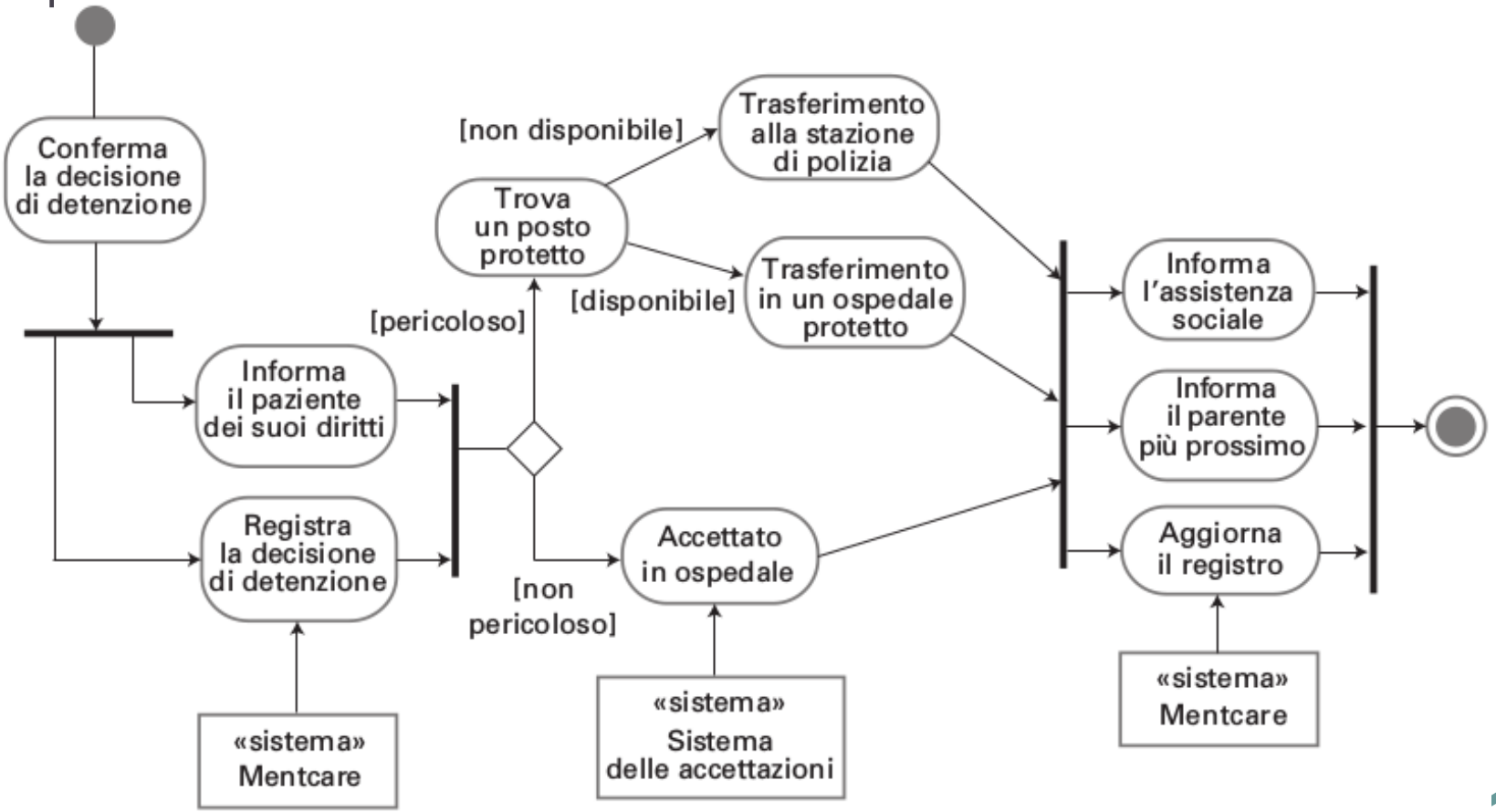
- L'inizio di un processo è indicato da un cerchio pieno.
- I rettangoli con gli spigoli arrotondati rappresentano le attività.
- La fine di un processo da un cerchio all'interno di un altro cerchio.
- Le frecce rappresentano il flusso di lavoro da un'attività all'altra.
  - Le annotazioni tra parentesi quadre specificano quando tale flusso viene seguito.
- Una barra piena indica il coordinamento delle attività
  - Quando il flusso da più attività porta a una barra piena, allora tutte queste attività devono essere completate prima che il processo possa avanzare.
  - Quando il flusso da una barra piena porta a più attività, queste possono essere eseguite in parallelo.

Diagrammi di attività: notazione (b)

- I rombi possono essere:
  - I nodi decisione: specificano percorsi alternativi, hanno un solo flusso di input e vari flussi di output sotto una condizione mutualmente esclusiva.
  - I nodi fusione: hanno vari input e un solo output, sul quale vengono indirizzati tutti i flussi in ingresso.

Diagrammi di attività: esempio

- Diagramma di attività per il processo di ricovero forzato per il sistema ospedaliero Mentcare



Unified Modeling Language
UML

Diagrammi di stato

Modelli di macchine a stati

- I modelli di macchine a stati mostrano come un sistema risponde agli eventi esterni e interni.
- I modelli di macchine a stati mostrano gli stati del sistema come nodi ed eventi come archi tra questi nodi. Quando si verifica un evento, il sistema si sposta da uno stato all' altro.
- Si basa sull'ipotesi che un sistema ha un numero finito di stati e che gli eventi (stimoli) possono causare una transizione da uno stato all'altro.
- Sono spesso utilizzati per modellare sistemi real-time.
- **I diagrammi di stato UML possono essere utilizzati per rappresentare i modelli di macchine a stati.**

Diagrammi di stato

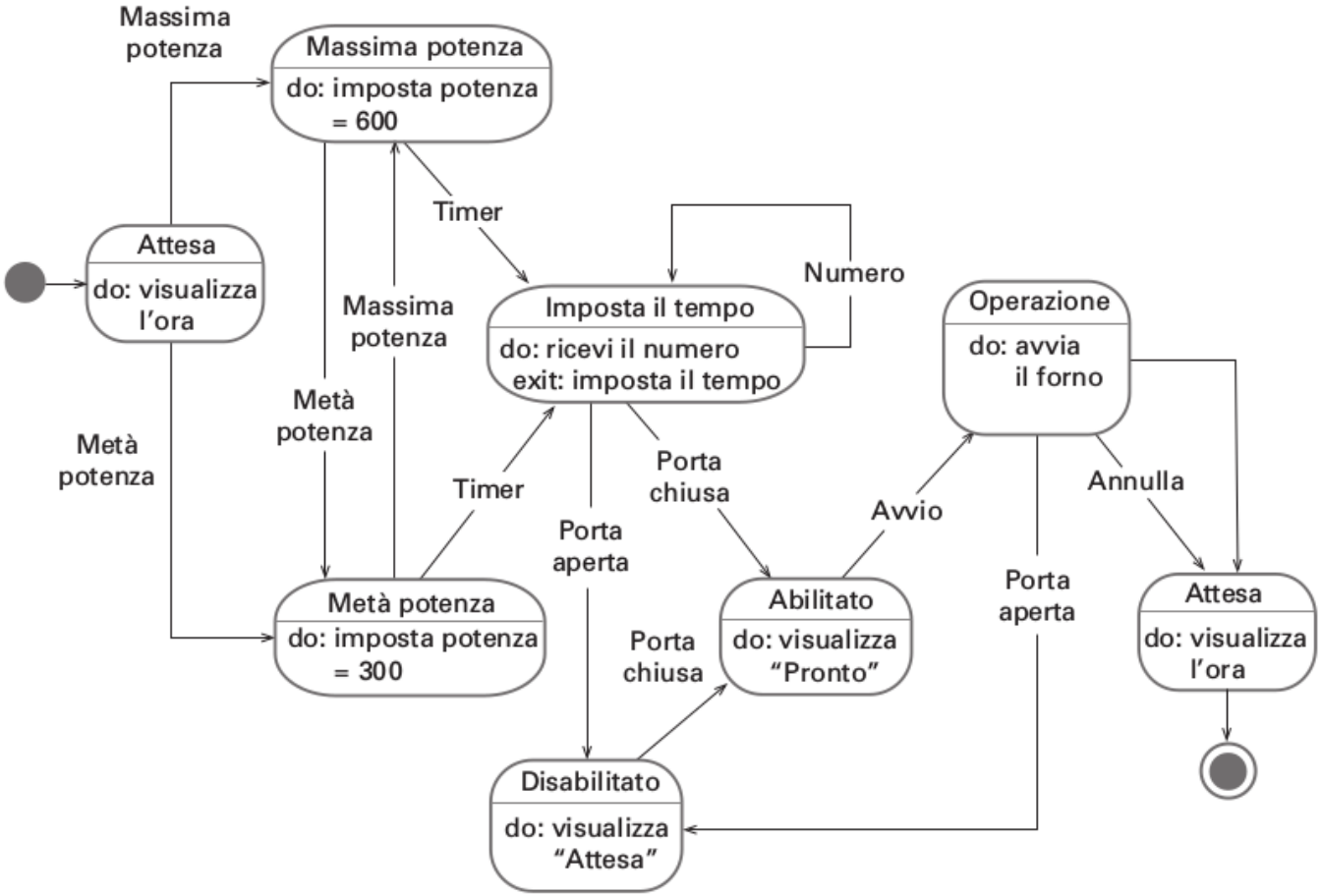
- I diagrammi di stato mostrano gli stati e gli eventi che causano le transizioni da uno stato all'altro.
- Non mostrano il flusso dei dati all'interno del sistema, ma possono includere informazioni addizionali sui calcoli da eseguire in ciascuno stato

Diagrammi di stato: notazione

- I rettangoli a spigoli arrotondati rappresentano gli stati del sistema
  - Possono includere una breve descrizione (dopo la parola “do”, letteralmente “fai”) delle azioni svolte in quello stato
- Le frecce con le annotazioni rappresentano gli stimoli che forzano una transizione da uno stato all’altro
- Si possono indicare gli stati iniziale e finale utilizzando i cerchi pieni, come nei diagrammi di attività.

Diagrammi di stato: esempio 1

- Diagramma di stato per un forno a microonde

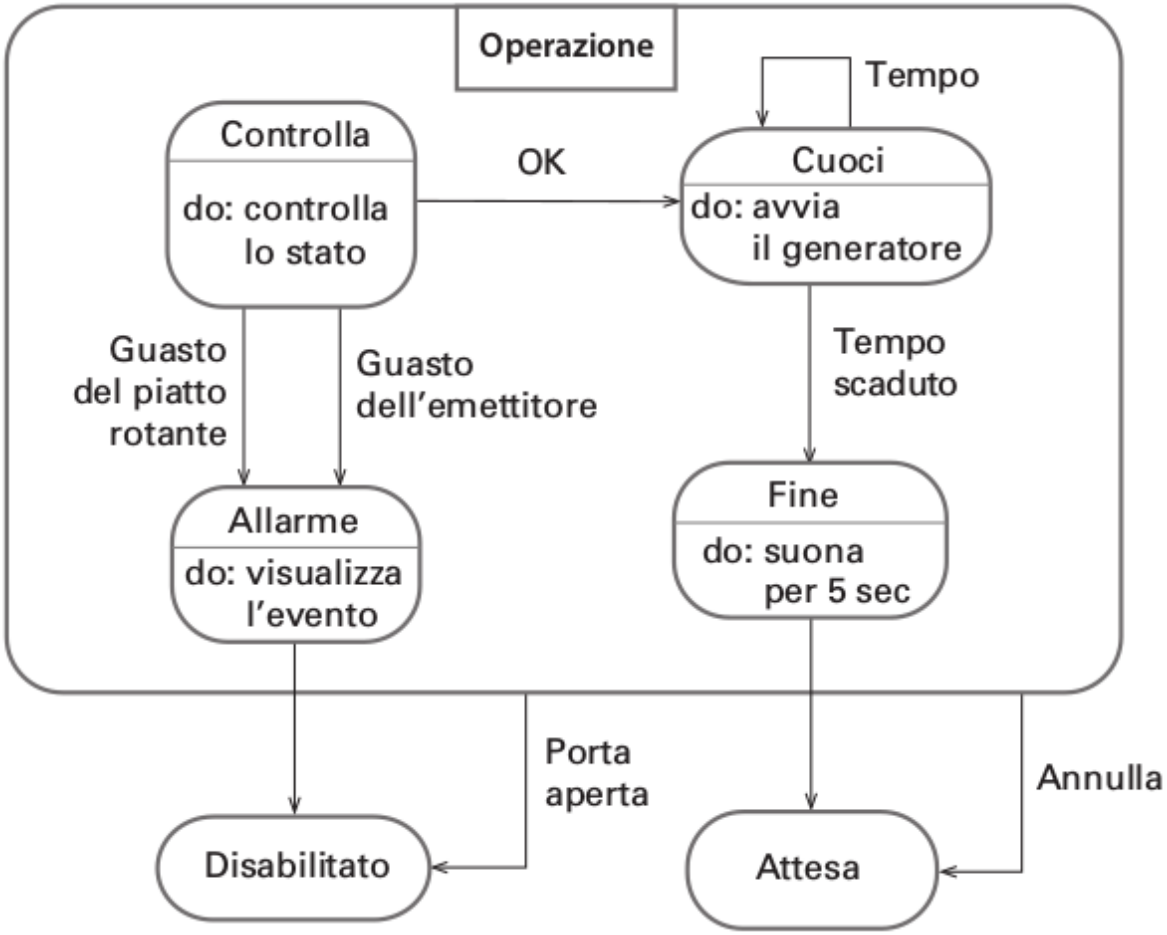


Problemi con i diagrammi di stato

- Il numero di possibili stati aumenta rapidamente.
- È necessario nascondere i dettagli nei modelli di grandi sistemi.
- Un modo per farlo è usare il concetto di “superstato” che comprende una serie di stati separati.
  - Il superstato somiglia a un singolo stato in un modello di alto livello, che poi viene espanso per mostrare maggiori dettagli su un diagramma distinto.
- Di solito occorre estendere questi modelli con una descrizione più dettagliata degli stimoli e degli stati del sistema.
  - Si può usare una descrizione tabellare

Diagrammi di stato: esempio 2

- Diagramma di stato dei sottostati dello stato "Operazione"



Stati per il forno a microonde

Stato	Descrizione
Attesa	Il forno aspetta l'input. Il display mostra l'ora corrente.
Metà potenza	La potenza del forno è impostata a 300 W. Il display visualizza "Metà potenza".
Massima potenza	La potenza del forno è impostata a 600 W. Il display visualizza "Massima potenza".
Imposta il tempo	Il tempo di cottura è impostato al valore immesso dall'utente. Il display visualizza il tempo di cottura selezionato e viene aggiornato non appena l'utente ha finito di impostarlo.
Disabilitato	Le operazioni del forno sono disattivate per sicurezza; la luce interna è spenta, il display visualizza "Attesa".
Abilitato	Le operazioni del forno sono attive; la luce interna è accesa; il display visualizza "Pronto".
Operazione	Il forno è in funzione; la luce interna è accesa; il display visualizza il conto alla rovescia del timer. Alla fine della cottura, il cicalino suona per 5 secondi. La luce del forno è accesa. Il display mostra "Fine cottura" mentre il cicalino suona.

Stimoli per il forno a microonde

Stimulus	Description
Metà potenza	L'utente ha premuto il pulsante di metà potenza.
Massima potenza	L'utente ha premuto il pulsante di massima potenza.
Timer	L'utente ha premuto uno dei pulsanti del timer.
Numero	L'utente ha premuto un tasto numerico.
Porta aperta	L'interruttore della porta non è chiuso.
Porta chiusa	L'interruttore della porta è chiuso.
Avvio	L'utente ha premuto il pulsante Avvio.
Annulla	L'utente ha premuto il pulsante Annulla.

Unified Modeling Language
UML

Diagrammi di sequenza

Modelli di interazione

- I modelli di interazione sono importanti perché aiutano ad identificare i requisiti dell'utente finale.
- La modellazione dell'interazione tra sistemi mette in evidenza eventuali problemi di comunicazione.
- La modellazione dell'interazione tra i componenti aiuta a capire se la struttura proposta per un sistema sarà in grado di fornire la fidatezza e le prestazioni richieste.
- **I diagrammi di sequenza possono essere utilizzati per la modellazione delle interazioni.**

Diagrammi di sequenza

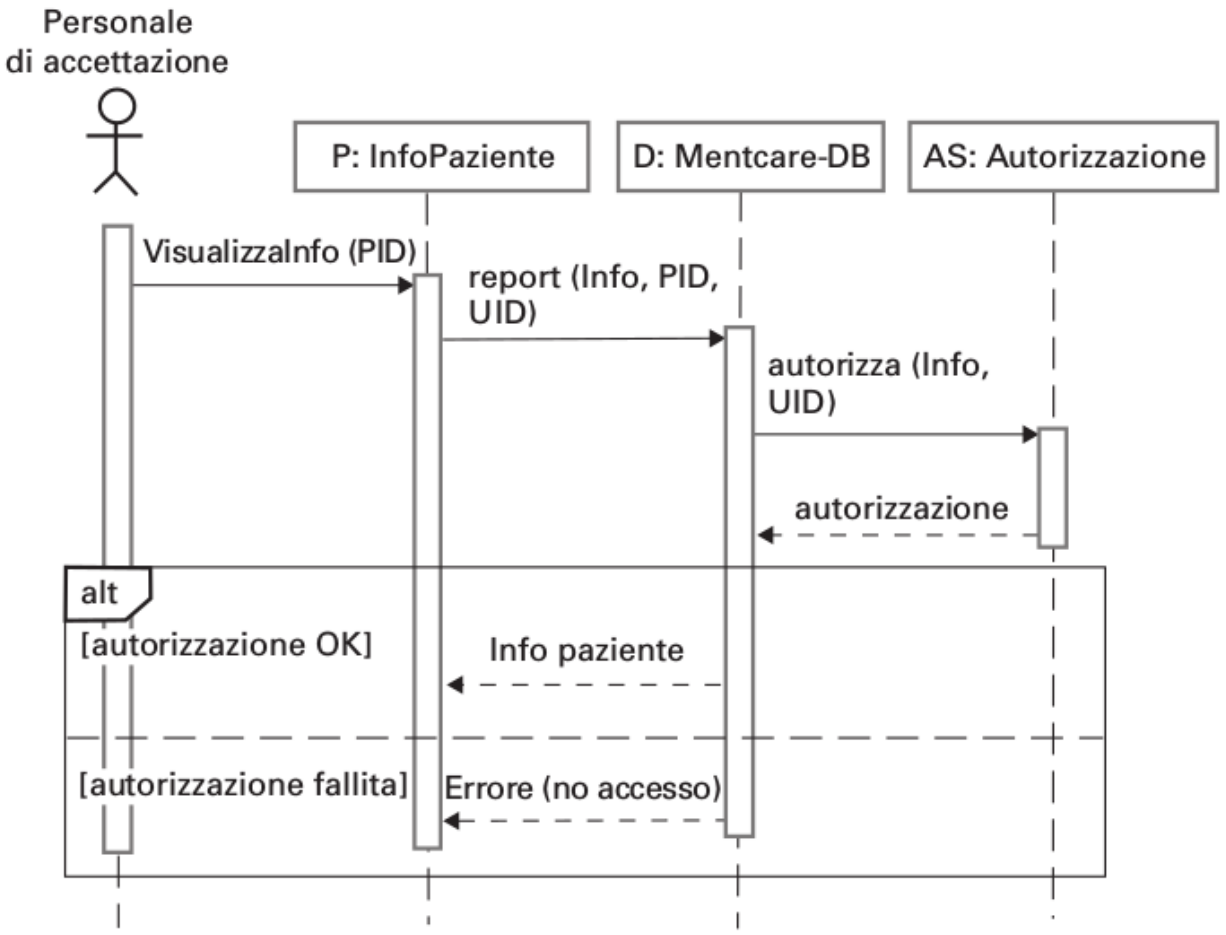
- I diagrammi di sequenza sono parte del linguaggio UML e sono utilizzati principalmente per modellare le interazioni tra gli attori e gli oggetti di un sistema e le interazioni tra gli stessi oggetti
- Un diagramma di sequenza mostra la sequenza delle interazioni che si svolgono durante un particolare caso d'uso.

Diagrammi di sequenza: notazione

- Gli oggetti e gli attori coinvolti sono elencati all'inizio del diagramma, con una linea tratteggiata verticale che parte da ciascuno di essi.
- Le frecce con le annotazioni indicano le interazioni tra gli oggetti.
  - Le annotazioni sulle frecce indicano le chiamate degli oggetti, i loro parametri e i valori di ritorno.
- Il rettangolo sulle linee tratteggiate indica la linea di vita dell'oggetto interessato (ovvero il tempo durante il quale l'istanza dell'oggetto prende parte al calcolo)
- Un box chiamato "alt" viene utilizzato con le condizioni indicate tra parentesi quadre, con le opzioni di interazione alternative separate da una linea tratteggiata.
- La sequenza delle interazioni va letta dall'alto verso il basso.

Diagramma di sequenza: esempio

- Diagramma di sequenza per la visualizzazione delle informazioni dei pazienti



Unified Modeling Language
UML

Diagrammi di classe

Modelli strutturali

- I modelli strutturali del software mostrano l'organizzazione di un sistema in funzione dei suoi componenti e le relazioni fra questi componenti.
- I modelli strutturali possono essere statici, se rappresentano l'organizzazione del progetto del sistema, o dinamici, se rappresentano l'organizzazione del sistema durante la sua esecuzione.
- **I diagrammi di classe possono essere utilizzati per modellare la struttura statica delle classi degli oggetti in un sistema software.**

Diagrammi di classe

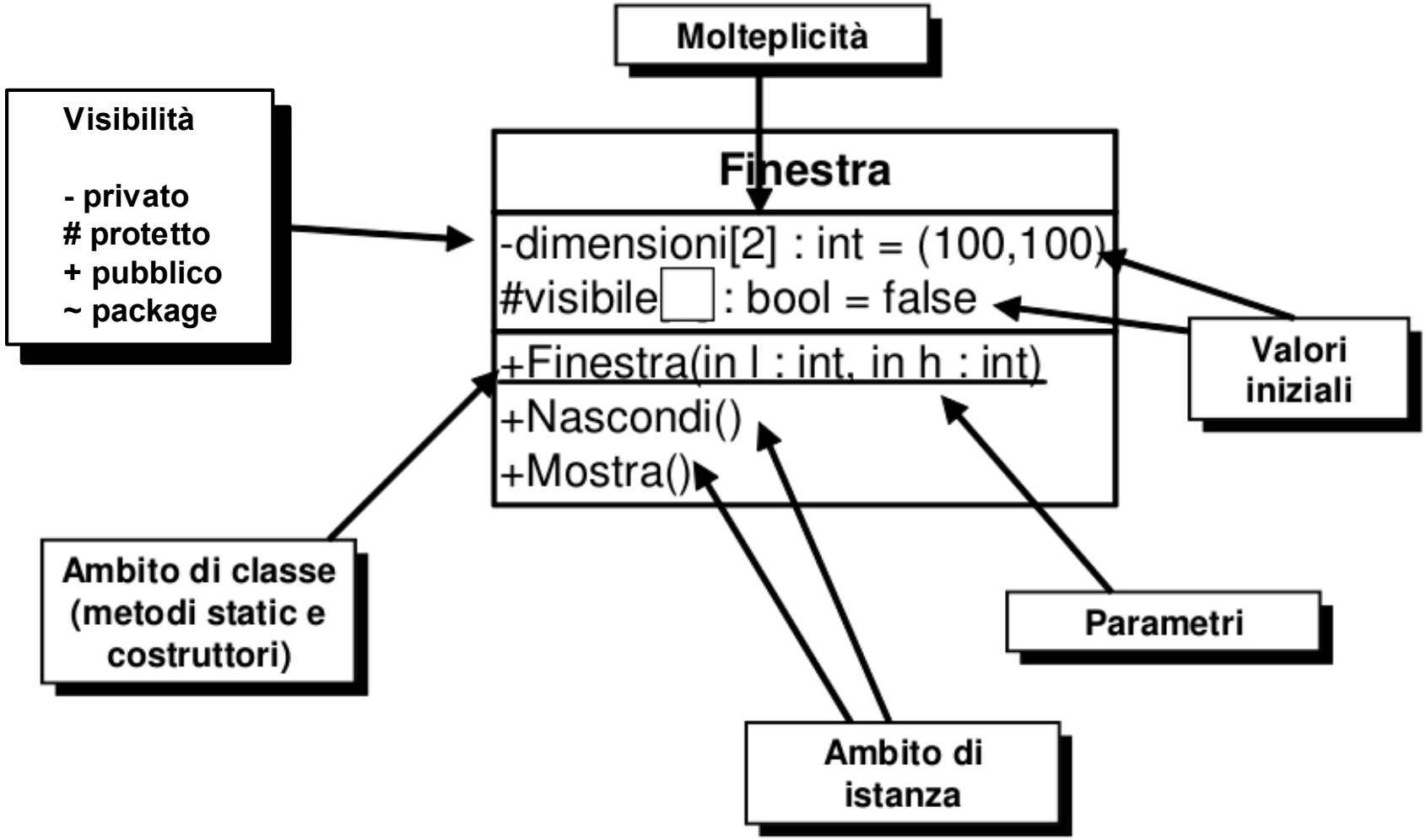
- I diagrammi di classe si usano quando si sviluppa un modello di sistema orientato agli oggetti per mostrare le classi di un sistema e le loro associazioni.
- Una classe di oggetti può essere pensata come una definizione generale di un tipo di oggetti del sistema.
- Un’associazione è un collegamento tra classi che indica che esiste qualche relazione tra queste classi.

Diagrammi di classe: notazione classe (a)

- Una classe è rappresentata da un rettangolo diviso in 3 sottosezioni: nome, attributi e operazioni
  - Solo il nome è obbligatorio ed è posto in cima al rettangolo
  - gli attributi della classe sono nella parte centrale; sono inclusi i nomi degli attributi e, facoltativamente, i loro tipi
  - le operazioni (chiamate metodi in Java o in altri linguaggi di programma zione orientata agli oggetti) associate alla classe di oggetti sono indicate nella parte inferiore del rettangolo.

Diagrammi di classe: notazione classe (b)

Diagrammi di classe: dettagli notazione



Attributi

- Gli attributi hanno una struttura di questo tipo:
  visibilità **nome** molteplicità: tipo = valoreIniziale
  (opzionale) (obbligatorio) (opzionale)
- La visibilità può essere:
  - private (-)
  - protected (#)
  - public (+)
  - package (~)
- La molteplicità si indica con la notazione tipica degli array: [n]
- Un'eventuale sottolineatura indica che l'attributo appartiene ad un ambito di classe (static) e non di istanza
- E' possibile indicare un valore iniziale

Operazioni

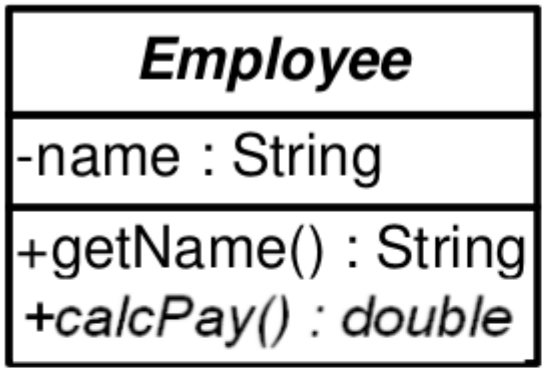
- Le operazioni sono i metodi della classe e hanno una struttura del tipo:

  visibilità **nome**(nomeParam: tipoParam, ...) : tipoRestituito

- Anche per le operazioni solo il nome è obbligatorio
- Le visibilità sono uguali a quelle degli attributi
- La sottolineatura indica metodi static e costruttori
- I parametri possono essere preceduti da un modificatore che indica la "direzione" di uso:
  - in: parametro in ingresso (per valore)
  - out: parametro di uscita (serve per restituire un risultato: per riferimento)
  - inout: parametro di ingresso e uscita (per riferimento)

Classi e metodi astratti

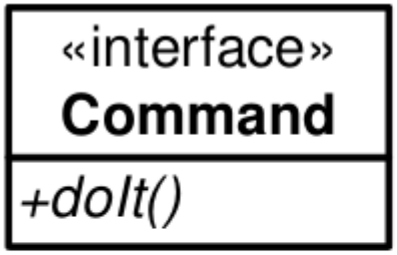
- Le entità astratte vengono rappresentate in corsivo: questo vale sia per le classi che per i metodi



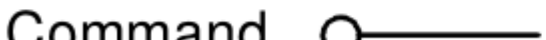

```
abstract class Employee
{
  public getName() { return name; }
  public abstract double calcPay();
  private String name;
}
```

Interfacce

- Le interfacce hanno rappresentazione simile a quella delle classi
- Vengono identificate tramite lo stereotipo «interface»
- Manca la sezione degli attributi
- I metodi sono in corsivo (astratti)
- Esiste anche una rappresentazione grafica compatta costituita da un cerchietto e una linea



Versione estesa



Versione compatta

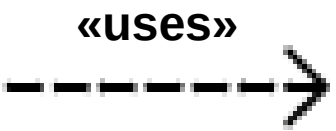
```
interface Command
{
  void doIt ();
}
```

Relazioni

- In UML una relazione è definita come:
  "Una connessione semanticamente significativa tra elementi di modellazione"
- Nei diagrammi delle classi troviamo 3 tipi di relazioni:
  - **Dipendenze** (relazioni d'uso)
  - **Associazioni** (associazione semplice, aggregazione, composizione)
  - **Generalizzazioni e realizzazioni** (ereditarietà fra classi e implementazione di interfacce)

Dipendenze

- "***Una dipendenza è una relazione tra due elementi dove un cambiamento ad uno di essi (fornitore) può influenzare o fornire informazioni necessarie all'altro (cliente)***"
- Il tipo più comune è la relazione d'uso che esprime un rapporto client-server fra due classi o fra una classe e un'interfaccia
- Si rappresenta con una linea tratteggiata e con lo stereotipo «uses» (che di solito però viene omesso)

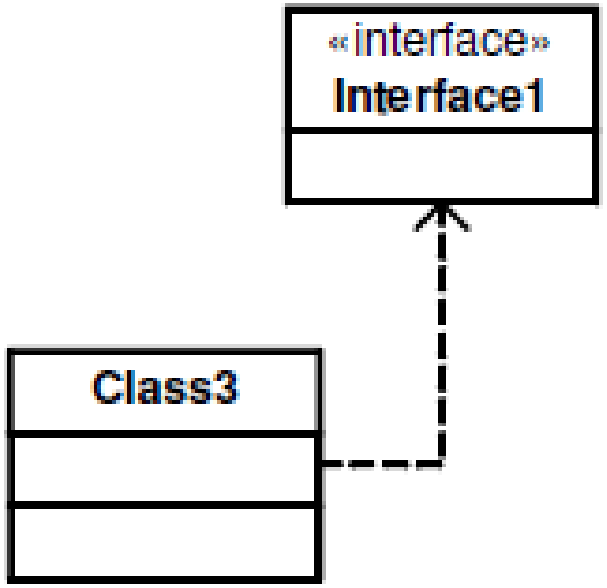


Relazione d'uso tra classi

- Una relazione d'uso fra due classi si ha nei seguenti 3 casi:
  - Un metodo di Class1 ha un parametro di tipo Class2
  - Un metodo di Class1 restituisce un valore di tipo Class2
  - Un metodo di Class1 usa un oggetto di tipo Class2 ma non come attributo.
- L'esempio più comune del terzo caso si ha quando in un metodo di Class1 si dichiara una variabile locale di tipo Class2

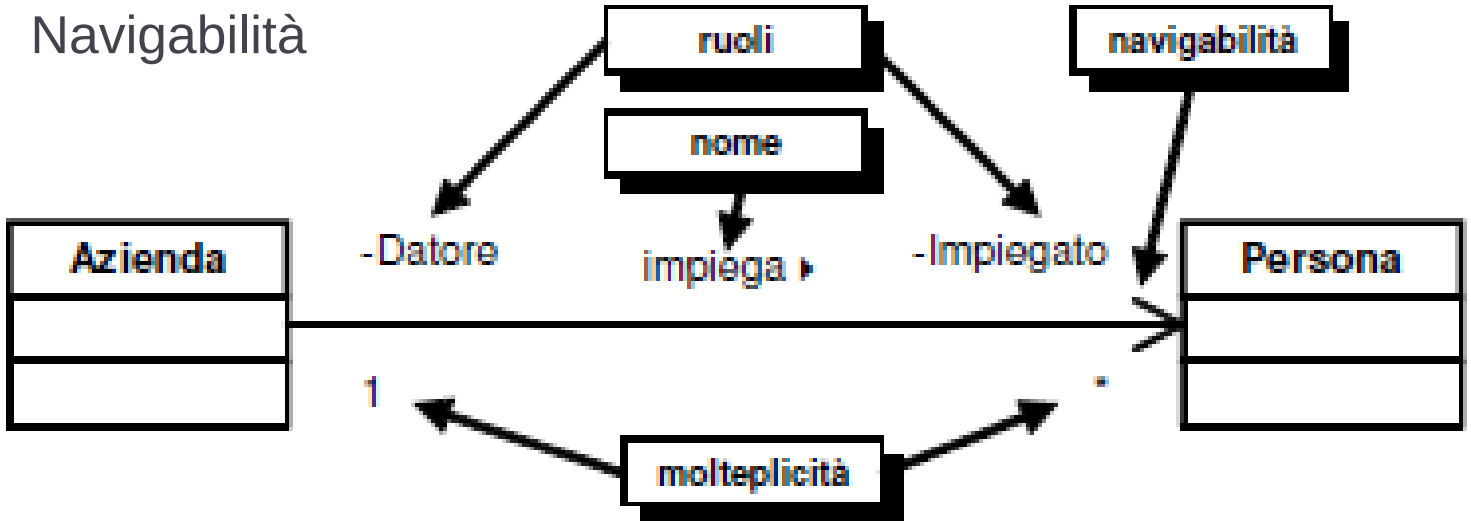
Relazione d’uso fra una classe e un’interfaccia

- Nel caso di una classe e di un’interfaccia la relazione d’uso ha un significato più generico
- Indica semplicemente che la classe invoca uno o più metodi definiti nell’interfaccia
- Talvolta si parla di relazione client-of



Associazioni

- Un’associazione è una relazione fra classi
- Può essere caratterizzata da:
  - Nome
  - Nomi dei ruoli
  - Molteplicità
  - Navigabilità

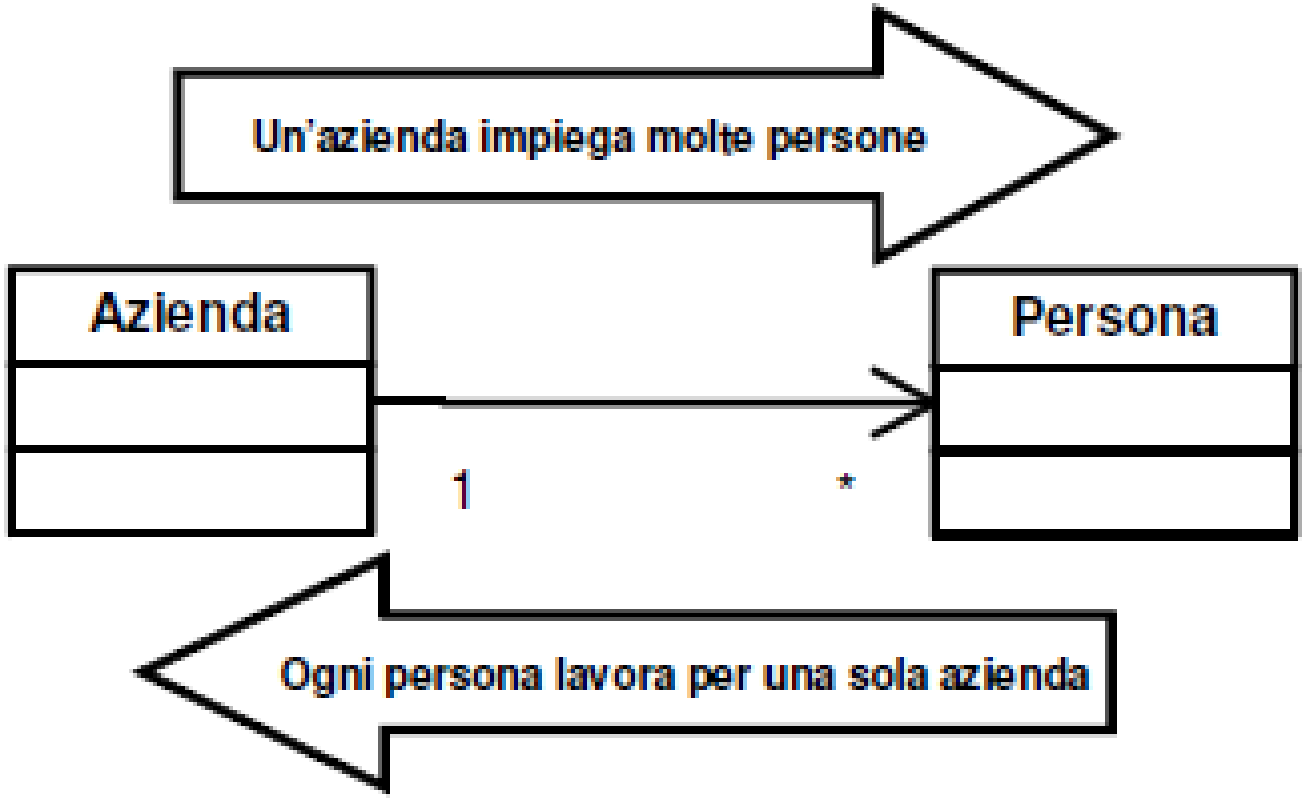


Nomi e ruoli

- Un’associazione può essere indicata con un nome o semplicemente con i nomi dei ruoli
- Sia nome che ruoli sono facoltativi
- La freccetta accanto al nome indica la direzione di lettura del nome:
  - Il termine “impiega” ha senso se si legge l’associazione da destra a sinistra
  - Si potrebbe invertire la direzione di lettura cambiando anche il nome dell’associazione in “è impiegato”

Molteplicità

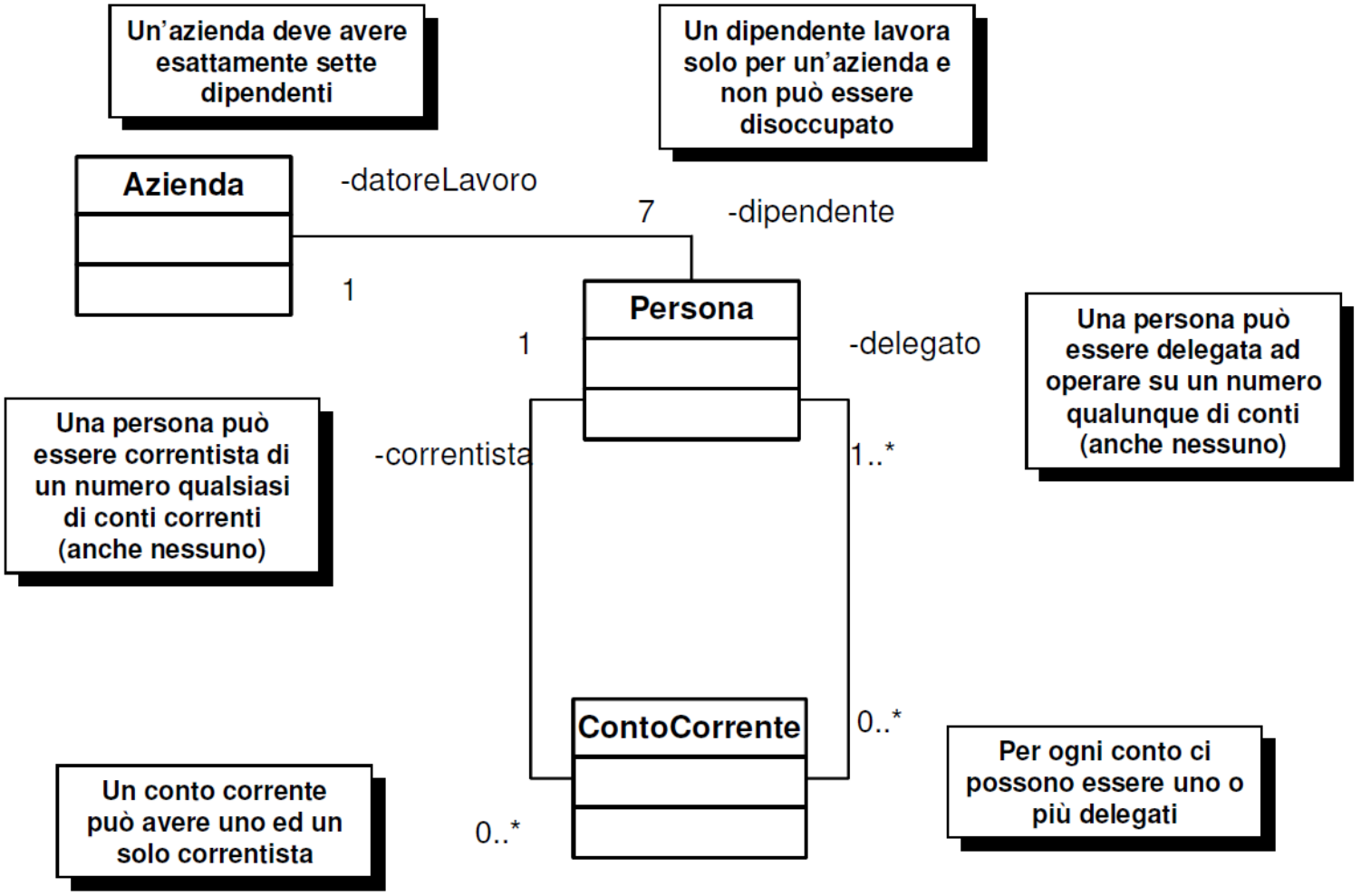
- La molteplicità è un vincolo la cui funzione è quella di limitare il numero di oggetti di una classe che possono partecipare ad un'associazione in un dato istante



Espressioni di molteplicità

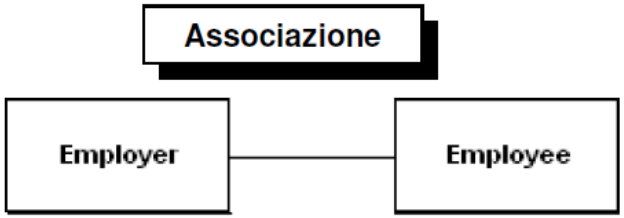
- La molteplicità può essere espressa:
  - Con un solo simbolo (0 1 *)
  - Con un intervallo (0..2 1..*)
  - Con una lista di simboli o intervalli separati da virgole (0..3,5,6..9)
- Esempi:
  - 0..1 Zero o uno
  - 1 Esattamente uno
  - 0..* Zero o più
  - * Zero o più
  - 1..* Uno o più
  - 1..6 Da uno a 6
  - 1..3,7,19..* Da 1 a 3, oppure 7 oppure da 19 in su

Esempio di molteplicità



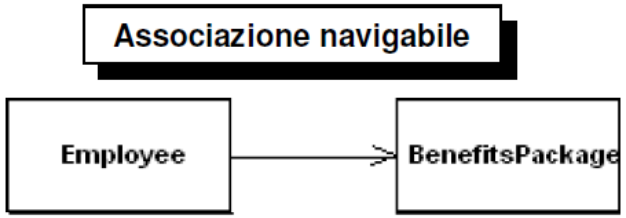
Navigabilità

- La navigabilità, indicata con una punta di freccia su un lato dell'associazione, indica una direzione nella relazione
- Specificando la navigabilità si riduce la dipendenza fra le classi: i legami bidirezionali sono critici




```
class Employer
{
  // ...
  private Employee itsWorker;
}

class Employee
{
  // ...
  private Employer itsBoss;
}
```




```
class Employee
{
  // ...
  private BenefitsPackage itsBenefits;
}

class BenefitsPackage
{
  // ... Non c'è un riferimento
    all'immpiegato
}
```

Aggregazione e composizione

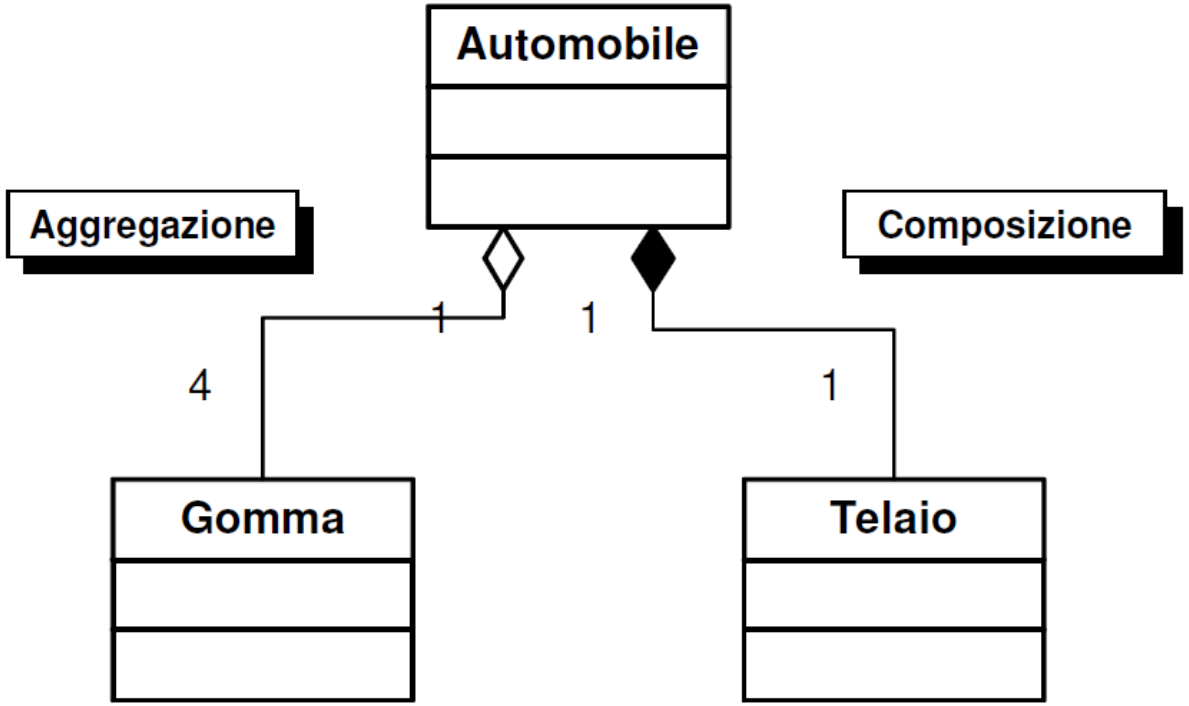
- Oltre all'associazione semplice esistono altre due relazioni di tipo analogo: l'aggregazione e la composizione
- Entrambe sono relazioni part-of, in cui un oggetto di una classe diventa parte di un oggetto dell'altra
- **Aggregazione**
  - Rappresentata come associazione con un **rombo bianco** vicino alla classe contenitore
- **Composizione**
  - Rappresentata come associazione con un **rombo nero** vicino alla classe contenitore

Aggregazione vs Composizione

- **Aggregazione:** è una forma specializzata di associazione in cui tutti gli oggetti hanno il proprio ciclo di vita, ma oggetti che sono proprietari di altri oggetti.
  - Esempio: Dipartimento e insegnante. Un singolo insegnante può appartenere a più dipartimenti e un dipartimento ha più insegnanti, ma se cancelliamo l'oggetto dipartimento, l'oggetto insegnante non sarà distrutto
- **Composizione:** è una forma specializzata di Aggregazione. L' oggetto figlio non ha il suo ciclo di vita e se l' oggetto genitore viene eliminato, anche tutti gli oggetti figli vengono eliminati. Gli oggetti figli non possono appartenere ad un altro oggetto genitore
  - Esempio: Università e dipartimento. L'università ha più dipartimenti e un dipartimento può appartenere ad una sola università. Se eliminiamo l'università eliminiamo anche i dipartimenti.

Esempio su aggregazione e composizione

- Un esempio classico per comprendere la differenza è quello dell'automobile: sia il telaio che le gomme sono parti dell'auto
- Ma la relazione con il telaio è più stretta: il telaio può appartenere ad una sola auto e nasce e muore assieme all'auto



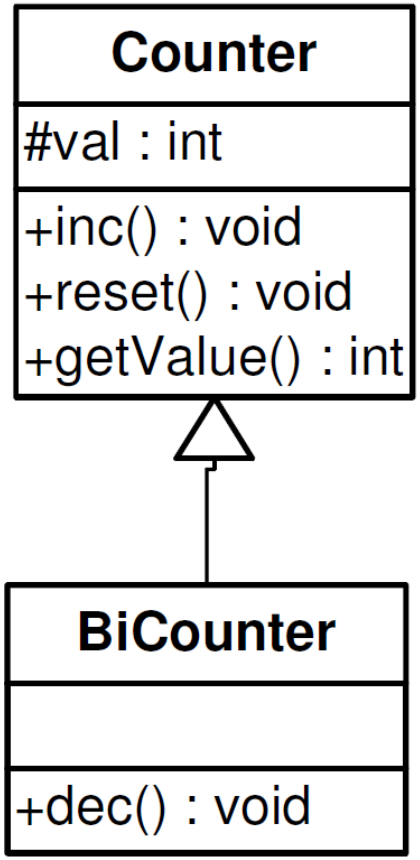
Esempio su aggregazione e composizione – 2

- Vediamo l'implementazione della classe Automobile:

```
class Automobile
{
    private Gomma gomme[];
    private Telaio telaio;
    public Automobile()
    {
      telaio = new Telaio();
      ...
    }
    public Gomma getGomma(int n)
    { return Gomme[n]; }
    public void setGomma(Gomma g; int n)
    { gomme[n] = g; }
}
```

Ereditarietà (Generalization)

- Rappresentata da una freccia con punta a triangolo che punta a una classe più generale

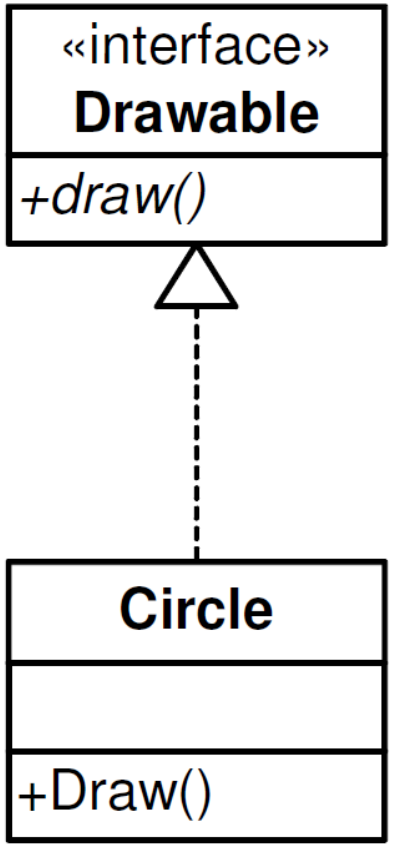



```
public class Counter
{
  protected int val;
  public void reset()
  { val = 0; }
  public void inc()
  { val++; }
  public int getValue()
  { return val;}
}

public class BiCounter
 extends Counter
{
 public void dec()
 { val--; }
}
```

Implementazione di interfacce (Realization)

- Rappresentata da una freccia con punta a triangolo e linea tratteggiata che punta alla classe implementata




```
interface Drawable
{
  public void draw();
}

class Circle
  implements Drawable
{
  public void draw()
  {
     ...
  }
 ...
}
```

Model-Driven Architecture

Model-driven architecture

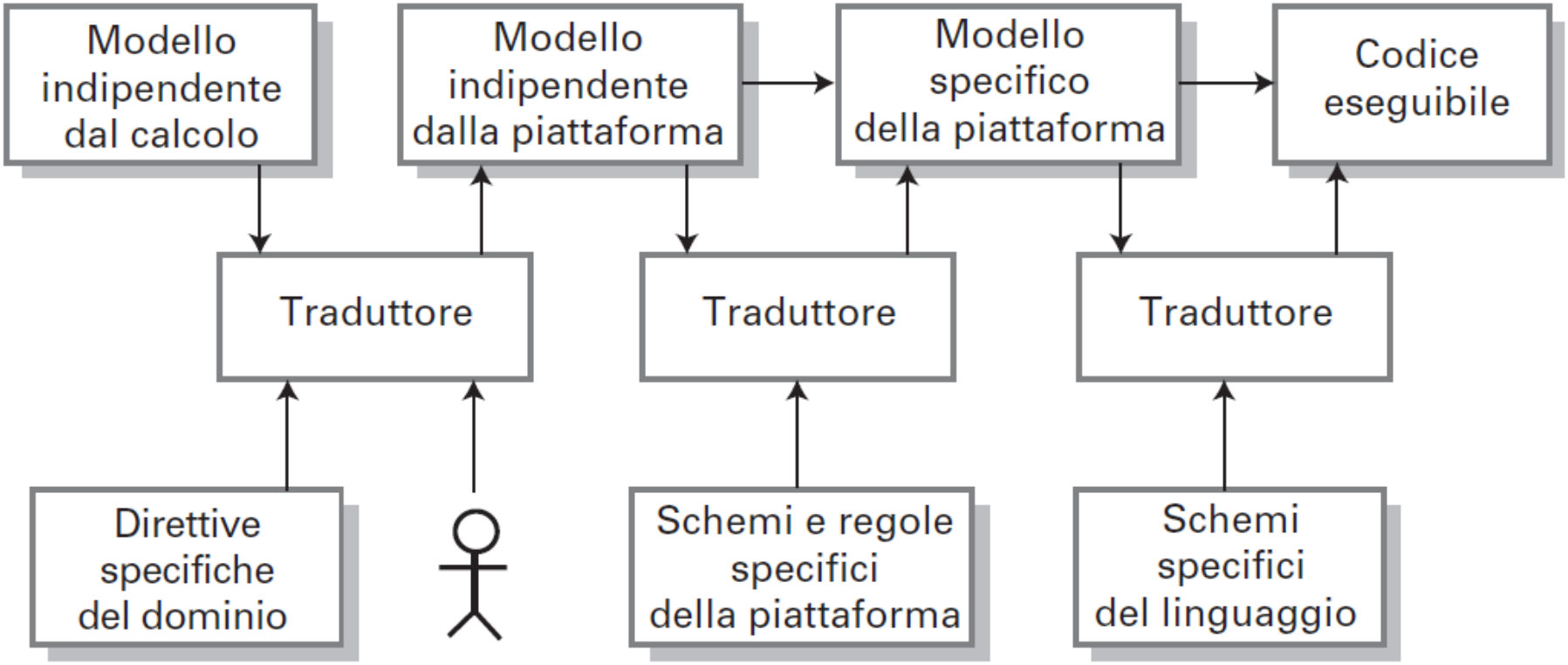
- La Model-driven architecture (MDA) è una tecnica basata sui modelli per progettare e implementare il software che usa un sottoinsieme di modelli UML per descrivere un sistema.
- I modelli sono creati a diversi livelli di astrazione. Da un modello di alto livello, indipendente dalla piattaforma, teoricamente è possibile generare un programma funzionante senza interventi manuali.

Tipi di modelli

L'MDA prevede 3 tipi di modelli astratti per un sistema

- I modelli indipendenti dal calcolo (CIM, *Computation Independent Model*)
  - si basano sulle principali astrazioni dei domini utilizzate in un sistema
- I modelli indipendenti dalle piattaforme (PIM, *Platform Independent Model*)
  - si basano sul funzionamento del sistema, senza riferimento alla sua implementazione
- I modelli specifici delle piattaforme (PSM, *Platform Specific Model*)
  - Sono trasformazioni dei modelli indipendenti dalle piattaforme con un PSM distinto per ciascuna piattaforma di applicazioni.

Trasformazioni MDA



Problemi con l’adozione dell’MDA (a)

Diversi fattori hanno limitato l’uso dell’MDA

- Per convertire i modelli da un livello all'altro sono necessari degli strumenti specializzati.
- La disponibilità degli strumenti è limitata e le organizzazioni possono richiedere l' adattamento e la personalizzazione degli strumenti al proprio ambiente.
- Per i sistemi di lunga durata sviluppati con MDA, le aziende sono riluttanti a sviluppare i propri strumenti o affidarsi a piccole aziende che potrebbero fallire.

Problemi con l'adozione dell'MDA (b)

- I modelli sono un buon modo per facilitare le discussioni sul progetto di un software. Tuttavia, non è sempre vero che le astrazioni utili per le discussioni siano quelle appropriate per l'implementazione.
- Per molti sistemi complessi, l'implementazione non è il problema principale.
  - L'ingegneria dei requisiti, la protezione e la fidatezza, l'integrazione con i sistemi tradizionali e i test sono tutti più importanti.
- Gli argomenti a favore dell'indipendenza dalle piattaforme sono validi soltanto per grandi sistemi di lunga durata. I risparmi derivanti dall'uso dell'MDA probabilmente sono vanificati dai costi della sua introduzione e dei suoi strumenti.