



---

**Università degli Studi di Parma**  
**Dipartimento di Scienze Matematiche,**  
**Fisiche e Informatiche**  
**Corso di Laurea in Informatica**

**Sistemi Informativi**

**Analisi di un processo  
aziendale**

**Giulio Destri**

# **Dr. Ing. Giulio Destri, Ph.D.**

---

**Professore a contratto di Sistemi Informativi  
@Università di Parma dal 2003**

**Digital Transformation Advisor, Innovation Manager,  
Business Coach, Trainer @LINDA**

**Esaminatore ISO27021 e UNI11506-11621 BA (EPBA)  
@Intertek**

**Membro commissione UNI/CT 526 @UNINFO**

**Blogger @6MEMES di MAPS**

**Certificazioni: ISO27001LA , ISO9001LA, ISO27021, ITILv3 e  
v4, COBIT-2019, SCRUM Master, EPBA, NLP Coach, NLP AMP**

**<https://www.linkedin.com/in/giuliodestri>**

**<http://www.giuliodestri.it/articoli.shtml>**

**[giulio.destri@unipr.it](mailto:giulio.destri@unipr.it)**

**[twitter.com/GiulioDestri](https://twitter.com/GiulioDestri)**

# **Scopo del modulo**

---

**Definire**

**I concetti base  
del dettaglio dei processi**

# Argomenti

---

- L'analisi dei processi business
- UML ed il suo ruolo
- Analisi del processo come successione di attività
- Analisi del processo come successione di casi d'uso
- Analisi delle entità che prendono parte ai processi
- Analisi delle interazioni tra gli elementi di un processo
- Un esempio completo di analisi
- Una visione d'insieme: il legame fra le viste e i pattern

# Business Analysis (o analisi business)

---

- Insieme di attività, conoscenze e tecniche necessarie per identificare le esigenze di business di un'organizzazione e determinare migliorie e soluzioni agli eventuali problemi esistenti.
- Disciplina che consente di identificare le **esigenze di business** e determinare le **soluzioni ai problemi aziendali**.

# Analisi dei requisiti

---

- Anche chiamata **Requirement Engineering**.
- Praticamente è l'analisi business orientata alla progettazione di soluzioni IT per la risoluzione delle esigenze che emergono
- Deve raccogliere i requisiti che servono al progetto software successivo

# Requisito? (IEEE 610,12-1.990)

---

- Una condizione o capacità necessaria per una delle parti interessate per risolvere un problema o raggiungere un obiettivo;
- Una condizione o capacità che deve essere soddisfatta o posseduta da un sistema o componente del sistema per soddisfare un contratto, standard, specifiche o altri documenti formalmente istituite;
- Una rappresentazione documentata di una condizione o capacità come ai punti precedenti.

# Requisiti vs. Progettazione

- Requisiti:
  - **Cosa** il sistema deve fare
  - Astrazione



- Progettazione:
  - **Come** il sistema deve farlo
  - Dettagli (anche tecnici)





# Suddivisione dei requisiti

---

- **Business Requirement** o requisiti di business
- **User Requirement** o requisiti utente
- **Functional Requirement** o requisiti funzionali
- **Assumption** e **Constraint** (ipotesi e vincoli)
- **Implementation requirement** o requisiti implementativi
- **Quality of Service Requirement** o requisiti di qualità di servizio

# Passaggio alla visione per processi

---

- **Business Process Management (BPM):** gestione ottimale di un processo, in modo da ridurre i suoi costi ed aumentare il valore da esso prodotto
- **Business Process Modeling:** analisi e modellazione del processo

# Business (process) modeling

---

Attività di rappresentare i processi di una organizzazione per potere:

- Analizzarli
- Scoprirne eventuali difetti
- Misurarne efficacia ed efficienza
- Migliorarne il funzionamento

# Business Architecture

---

“Una cianografia (figura disegnata nei contorni) dell’azienda (od organizzazione) che provvede una conoscenza comune della sua organizzazione e viene usata per allineare gli obiettivi strategici e le esigenze tattiche.” (OMG)

# Business Architecture

---

Definisce la struttura funzionale di una organizzazione in termini


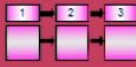



























- di servizi business e
- delle informazioni business per essi necessarie

# Business Capability

---

Abilità che i servizi business hanno di provvedere le funzionalità business necessarie portando i risultati business attesi ed il valore ad essi associato

# Business Architecture: Zachman Framework

	WHAT	HOW	WHERE	WHO	WHEN	WHY
	DATA	FUNCTION	NETWORK	PEOPLE	TIME	MOTIVATION
<b>SCOPE</b> {contextual}  Planner	List of things important to the business  Entity = Class of business things	List of processes the business performs  Process = Class of business process	List of locations in which the business operates  Node = Major business locations	List of organisations important to the business  People = Major business unit	List of event cycles significant to the business  Time = Major Business Event Cycle	List of business goals/strategies  End/Mean = Major Business Goal/Strategy
<b>BUSINESS MODEL</b> {Conceptual}  Owner	e.g., Semantic Model  Entity = Business Entity Relationship = Business	e.g., Business Process Model  Process = Business I/O = Business Resource	e.g., Business Logistics System  Node = Business Location Link = Business Linkage	e.g., Workflow Model  People = Organisation unit Work = Work Product	e.g., Master Schedule  Time = Business Event Cycle = Business Cycle	Business Plan  End = Business Objective Means = Business Strategy
<b>SYSTEM MODEL</b> {Logical}  Designer	e.g., Logical Data Model  Entity = Data Entity Relationship = Data Relationship	e.g., Application Architecture  Process = Application Function I/O = User Views	e.g., Distributed System Model  Node = I/O Function Relationship = Line Characteristics	e.g., Human Interface Architecture  People = Role Work = Deliverable	e.g., Processing Structure  Time = System Event Cycle = Processing Cycle	e.g., Business Rule Model  End = Structural Assertion Means = Action Assertion
<b>TECHNOLOGY MODEL</b> {Physical}  Builder	e.g., Physical Data Model  Entity = Segment/Table Relationship = Pointer/key	e.g., System Design  Process = Computer Function I/O = Data Elements/sets	e.g., Technology Architecture  Node = H/w /System s/w Relationship = Line Specifications	e.g., Presentation Architecture  People = User Work = Screen Formats	e.g., Control Structure  Time = Execute Cycle = Component Cycle	e.g., Rule Design  End = Condition Means = Action
<b>DETAILED REPRESENTATIONS</b> {Out-of-context}  Subcontractor	e.g., Data Definition  Entity = Field Relationship = Address	e.g., Program  Process = Language Statement I/O = Control Block	e.g., Network Architecture  Node = Address Link = Protocol	e.g., Security Architecture  People = Identity Work = Job	e.g., Timing Definition  Time = Interrupt Cycle = Machine Cycle	e.g., Rule Specification  End = Sub-condition Means = step
<b>FUNCTIONING ENTERPRISE</b>	e.g DATA	e.g FUNCTION	e.g NETWORK	e.g ORGANISATION	e.g SCHEDULE	e.g STRATEGY

# Zachman Framework:

## Le viste

---

- **Pianificatore** (Planner): comprende l'ambito aziendale e può offrire una visione contestuale dell'impresa.
- **Proprietario** (Owner): comprende il modello di business e può fornire una visione concettuale dell'impresa.
- **Costruttore** (Builder) - sviluppa il modello di sistema e può costruire una visione logica dell'impresa
- **Progettista** (Designer) - produce il modello tecnologico e può fornire una visione fisica dell'impresa.



# Zachman Framework:

## Le viste - 2

---

- **Integratore** (subcontractor) - comprenderà le rappresentazioni dettagliate di specifici elementi nel business, anche se avranno un vista del contesto dell'azienda.
- **Utente** (User): fornisce una visione dell'impresa funzionante, dal punto di vista di un utente (ad es. Un dipendente, un partner o un cliente).

# Zachman Framework:

## Le domande

---

- What / **Cosa** (dati) - quali sono i dati, le informazioni o gli oggetti aziendali?
- How / **Come** (funzione) - come funziona l'azienda, cioè quali sono i processi aziendali?
- Where / **Dove** (rete) - dove operano le imprese?
- Who / **Chi** (persone): chi sono le persone che gestiscono il business, quali sono le unità aziendali e la loro gerarchia?

# Zachman Framework:

## Le domande - 2

---

- When / **Quando** (tempo) - quando vengono eseguiti i processi aziendali, cioè quali sono le pianificazioni commerciali e i flussi di lavoro?
- Why / **Perché** (motivazione): perché i processi, le persone o le sedi sono importanti per l'azienda, ovvero i fattori di business o gli obiettivi aziendali?

# Passaggio alla visione per processi

---

- **Business Process Improvement (BPI)**: interventi di tipo incrementale e cioè volti al continuo e graduale miglioramento dei processi
- **Business Process Reengineering (BPR)**: interventi di tipo radicale e cioè volti al completo ridisegno del processo qualora esso si manifesti del tutto inadeguato agli obiettivi da raggiungere

# Argomenti

---

- L'analisi dei processi business
- UML ed il suo ruolo
- Analisi del processo come successione di attività
- Analisi del processo come successione di casi d'uso
- Analisi delle entità che prendono parte ai processi
- Analisi delle interazioni tra gli elementi di un processo
- Un esempio completo di analisi
- Una visione d'insieme: il legame fra le viste e i pattern

# UML

---

- Unified Modeling Language
- Linguaggio unificato per la modellazione di concetti, entità, funzionalità, processi e relazioni che fra essi intercorrono
- Nasce nel 1997
- Unisce precedenti sintassi di modellazione (Booch, OMT, OOSE)
- Standard gestito dal consorzio OMG (<http://www.omg.org>)

# UML come strumento di modellazione

---

Il linguaggio UML serve a descrivere, in modo grafico e “compatto”

- I requisiti utente (use case)
- Le componenti dei sistemi
- I dati in esse contenuti
- Le azioni da esse svolte
- Le relazioni che fra loro intercorrono (il processo in cui esse operano)

# Evoluzione di UML

---

- Dopo il 1997 le estensioni standard di UML si susseguono
- Anche l'uso viene esteso alle varie fasi di realizzazione dei sistemi IT
- Attualmente i riferimenti sono lo standard 1.5 (per alcuni strumenti) e il 2.5 (per la maggioranza degli strumenti)



# Usare UML

---

- Normalmente si usano programmi appositi, i Computer-Aided Software Engineering tool (CASE)
- Molti CASE sul mercato, sia open, sia commerciali
- Non sempre interscambio formati possibile
- Un formato standard: XMI (con limiti)

# Usare UML: alcuni CASE

---

- IBM Rational Rose (commerciale)
- Gentleware Poseidon for UML (com.)
- Tigris ArgoUML (free)
- StarUML (free ma desupportato)
- Visual Paradigm (free e com.)
- PlugIn di Eclipse: AmaterasUML (free)
- PlugIn di VS e di NetBeans
- ...

# Obiettivo di UML

---

- Linguaggio di modellazione, semigrafico, utilizzabile da tecnici, da non tecnici e dalle macchine
- Un modello è una semplificazione della realtà che si ottiene
  - Riducendo le caratteristiche in esame
  - Considerando solo quelle utili al fine dell'analisi in corso
- Potenza espressiva nella documentazione

# L'evoluzione logica di UML

---

- Nella catena di produzione dei sistemi IT, l'accento viene posto sempre più sulle fasi di analisi e progettazione
- Una fase di analisi deve provvedere la conoscenza di dove si è (AS IS) e dove si vuole andare (TO BE)
- Uso di UML anche nel Business Modeling

# Tipi di diagrammi UML: analisi

---

- dei casi d'uso - **Use Case Diagram**
- delle classi - **Class Diagram**
- degli oggetti - **Object Diagram**
- di sequenza - **Sequence Diagram**
- di collaborazione - **Collaboration o Communication Diagram**
- di transizione di stato - **Statechart Diagram**
- delle attività - **Activity Diagram**

# Tipi di diagrammi UML: implementazione

---

- dei componenti - **Component Diagram**
- di distribuzione - **Deployment Diagram**

# Diagrammi UML 2.x: strutturali

---

- **Package diagram**
- **Class o Structural diagram**
- **Object diagram**
- **Composite Structure diagram**
- **Component diagram**
- **Deployment diagram**
- **Profile diagram**

# Diagrammi UML 2.x: comportamentali

---

- **Use Case diagram**
- **Activity diagram**
- **State Machine diagram**

## **Comportamentali di interazione**

- **Communication diagram**
- **Sequence diagram**
- **Timing diagram**
- **Interaction Overview diagram**



# Diagrammi UML 2.x: estensione

---

- **Business Process Model Diagram**

# Il concetto di Vista

---

Vista, traduzione dell'inglese view

- Un diagramma esprime solo alcune caratteristiche
  - definisce un modello che proietta e rende visibili solo alcune caratteristiche
  - Quelle che si vuole esaminare

# L'uso dei diagrammi UML: analisi e design

---

- **Use Case Diagram:** per capire nei dettagli “cosa” il sistema deve fare
- **Class/Object Diagram:** per definire le entità fondamentali
- **Activity/Statechart Diagram:** per definire i processi fondamentali (fortemente “imparentati” con gli Use Case Diagram)
- **Collaboration Diagram:** per definire le interazioni fra le entità fondamentali
- **Sequence Diagram:** per definire la sequenza delle interazioni fra entità

# L'uso dei diagrammi UML: implementazione/installazione

---

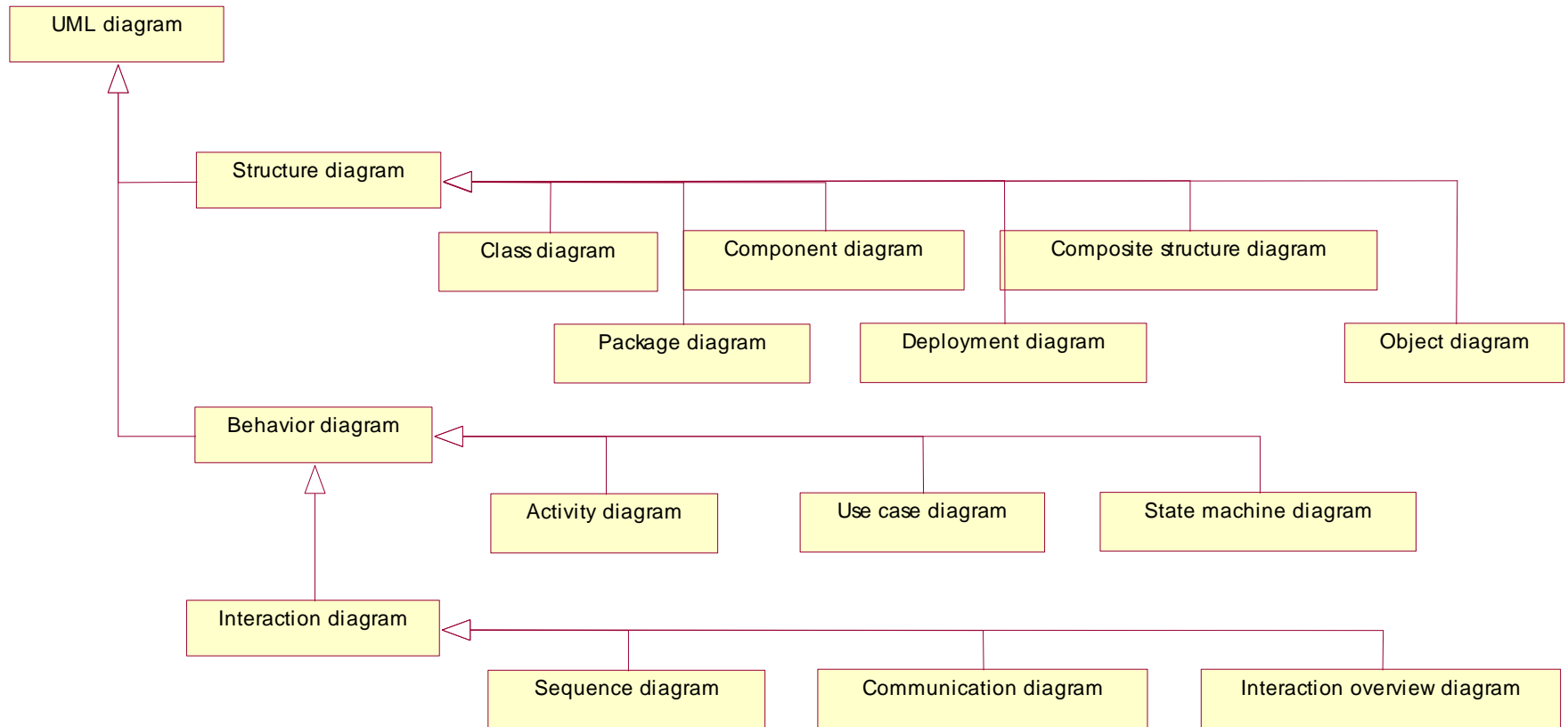
- **Component Diagram:** per definire nei dettagli quali parti compongono il sistema/prodotto software finito e le loro relazioni
- **Deployment Diagram:** per definire dove le varie parti di un programma devono essere poste in un'architettura distribuita (es. client-server)

# Riassumendo

---

- Modelli Funzionali
  - Use case diagram
- Modelli di strutture (statici)
  - Class diagram
- Modelli dinamici o di comportamento
  - Statechart diagram
  - Sequence diagram
  - Activity diagram

# Riassumendo



# Diagrammi UML 2.x: meno usati

---

- **Package diagram**
- **Composite Structure diagram**
- **Profile diagram**
- **Timing diagram**
- **Interaction Overview diagram**

# Uso di UML: IT e Business Modeling

---

- Il sistema informativo è l'insieme delle risorse umane, materiali, organizzative (regole) che gestisce e distribuisce le informazioni entro l'azienda
- Il sistema informatico è l'insieme degli strumenti IT di un sistema informativo
- Serve un linguaggio chiaro comune a tutti i componenti del sistema informativo



# Uso di UML: Business Modeling

---

- Usare le stesse notazioni di UML per
  - Esprimere i concetti del business
  - Per uso diretto (per uso di analisi business)
  - Per uso indiretto (in vista di una successiva implementazione di sistemi IT di ausilio al business)

# Uso di UML: Business Modeling

---

- Usare UML in tutto il sistema informativo, come linguaggio comune
- Il processo di sviluppo software usa lo stesso linguaggio sin dalle prime fasi dell'analisi
- Comunicazione con fornitori IT con linguaggio comune

# Uso di UML: Business Modeling

---

- Metodologie unificate
  - RUP (Rational Unified Process) di Rational (<http://www.rational.com>)
  - Scuola scandinava (Eriksson e Penker) (<http://www.opentraining.com>)
- Approcci nuovi
  - EbXML

# Vantaggi di UML nel Business Modeling

---

L'uso di UML consente di focalizzare l'attenzione

- sui vari elementi costitutivi un processo
- sui vari punti di vista che si possono seguire



- 
- **Activity Diagram**
  - **Use Case Diagram**
  - **Class Diagram**
  - **Sequence Diagram**
  - **Collaboration Diagram**
  - **Statechart Diagram**

# Activity Diagram (diagrammi d'attività)

---

- Rappresentano una procedura o un workflow
- Mostrando l'evoluzione di un flusso di attività
- Ogni attività è definita come un'evoluzione continua, non necessariamente atomica, di uno stato
- Sono una evoluzione dei flow-chart

# Activity Diagram: elementi base

---

- Activity (Attività)

Esecuzione non atomica entro un sistema dotato di stati.

Può essere scomposta in azioni.

- Action (Azione)

Operazione atomica eseguibile che produce come risultato un cambiamento nello stato di un sistema o il ritorno di un valore.

# Activity Diagram: concetti base

---

- **Action State (Stato di azione)**  
Uno stato che rappresenta l'esecuzione di un'azione (atomica), tipicamente l'invocazione di una operazione.
- **Activity State (Stato di attività)**  
Stato composito, in cui il flusso di controllo è formato di altri stati di attività e stati di azione. Non è atomico, il che significa anche che può essere interrotto. Può anche essere ulteriormente scomposto in altri diagrammi di attività



# Activity Diagram: elementi di contorno

---

- Transition (Transizione)

Rappresenta il flusso di controllo fra due attività, che mostra il percorso da un action o activity state al successivo action o activity state.

- Object Flow

Rappresenta un oggetto (un'entità) coinvolta nel flusso di controllo associato con un activity diagram.

# Activity Diagram: elementi di contorno

---

- Object State

Una condizione o situazione operativa nella vita di un oggetto (un'entità) durante la quale l'oggetto soddisfa certe condizioni, compie certe attività o attende certi eventi.

- Swimlane

Una suddivisione per l'organizzazione di responsabilità per le attività. Non ha un significato fisso, ma spesso corrisponde alla unità organizzativa entro un business model (es. ufficio acquisti, vendite...).

# Activity Diagram: tipologie

---

- Activity state (Diagram)

Le singole attività hanno una durata e possono essere ulteriormente scomposte, dando origine ad altri diagrammi
- Action state (Diagram)

Le singole attività sono atomiche e non possono essere ulteriormente scomposte

# Activity Diagram: significato

---

- Enfasi posta sulle attività e non su chi le compie
- Enfasi sulla sequenza di azioni di una particolare procedura
- Vengono evidenziati vincoli di precedenza o di concorrenza

# Activity Diagram: simboli base

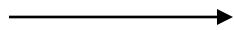
---

 Inizio di un diagramma di attività

 Termine di un diagramma di attività

Attività 1

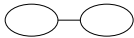
Singola attività generica con nome “Attività 1”



Connessione fra attività

Attività 2

Subattività generica (scomponibile)



Non vi è differenza fra simboli di attività e simboli di azione  
Nome del diagramma di attività (in fondo)

# Activity Diagram: simboli base

---

 Inizio di un diagramma di attività

 Termine di un diagramma di attività

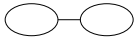
Attività 1

Singola attività o azione generica  
con nome “Attività 1”



Connessione fra attività

Attività 2



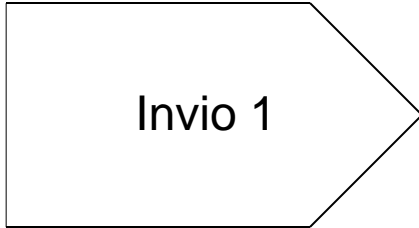
Attività generica di cui viene  
esplicitata la scomponibilità

Processo 1

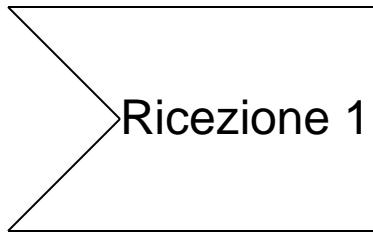
Processo generico con nome “processo 1”

# Activity Diagram: simboli di invio/ricezione

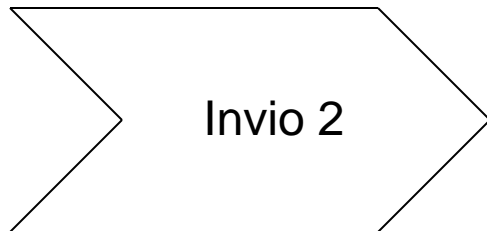
---



Attività con invio con nome “Invio 1”



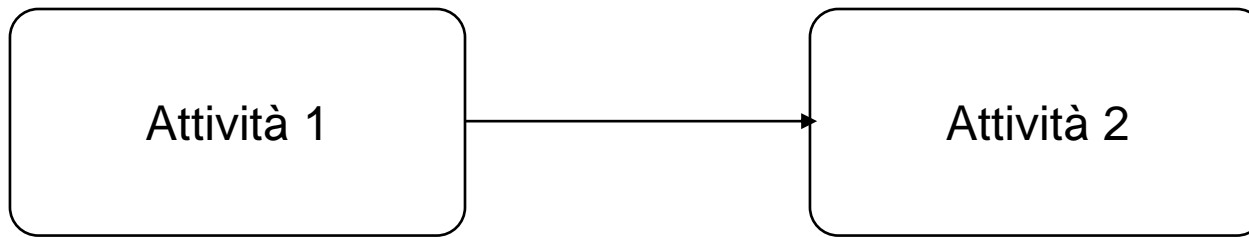
Attività con ricezione con nome “Ricezione 1”



Attività con invio e ricezione con nome “Invio 2”

# Tipi di AD: sequenza semplice

---

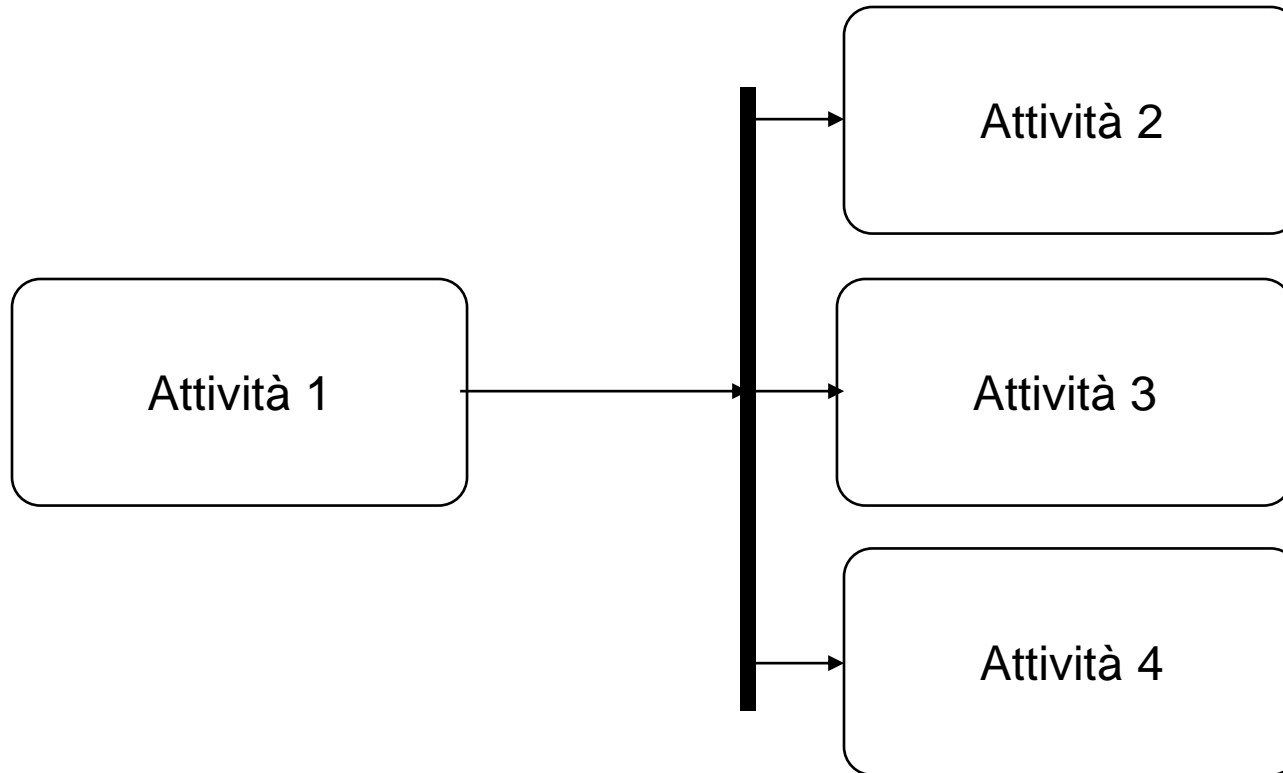


Un'attività (Attività 2) viene eseguita dopo la fine della  
Precedente (Attività 1)  
**(Single Thread)**



# Tipi di AD: AND-split

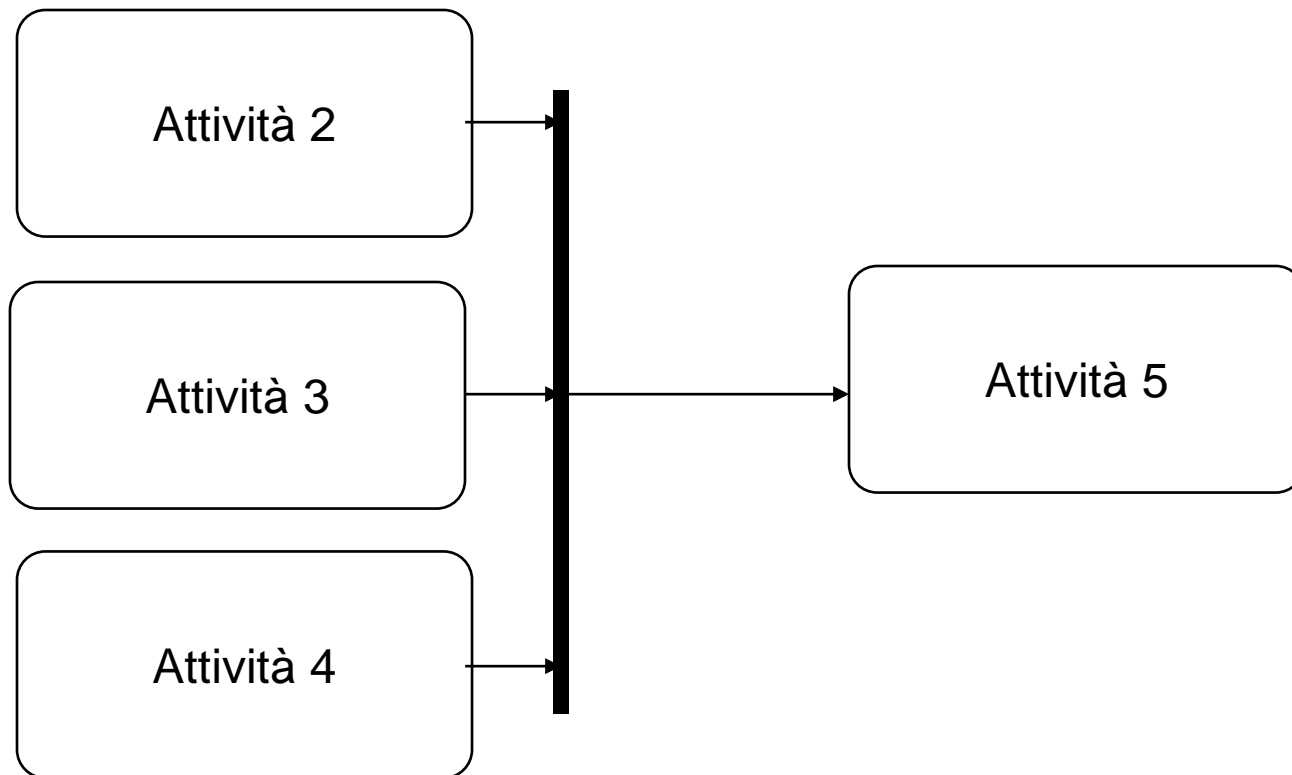
---



Un singolo flusso di attività si divide in più flussi, consentendo l'esecuzione simultanea di più attività  
**(Multiple Thread)**

# Tipi di AD: AND-join

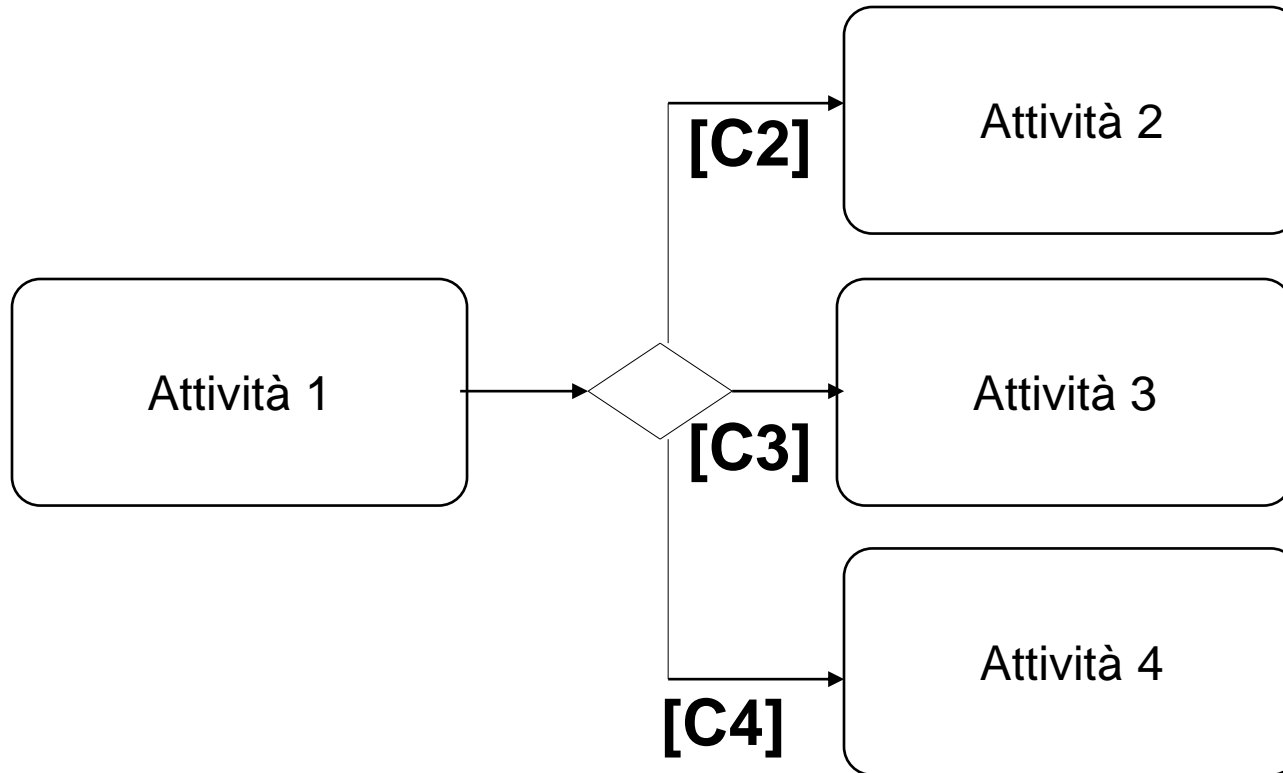
---



Due o più flussi di attività convergono in uno solo  
E' un punto di sincronizzazione per il workflow: non si va avanti finché non sono terminate tutte le attività precedenti

# Tipi di AD: OR-split

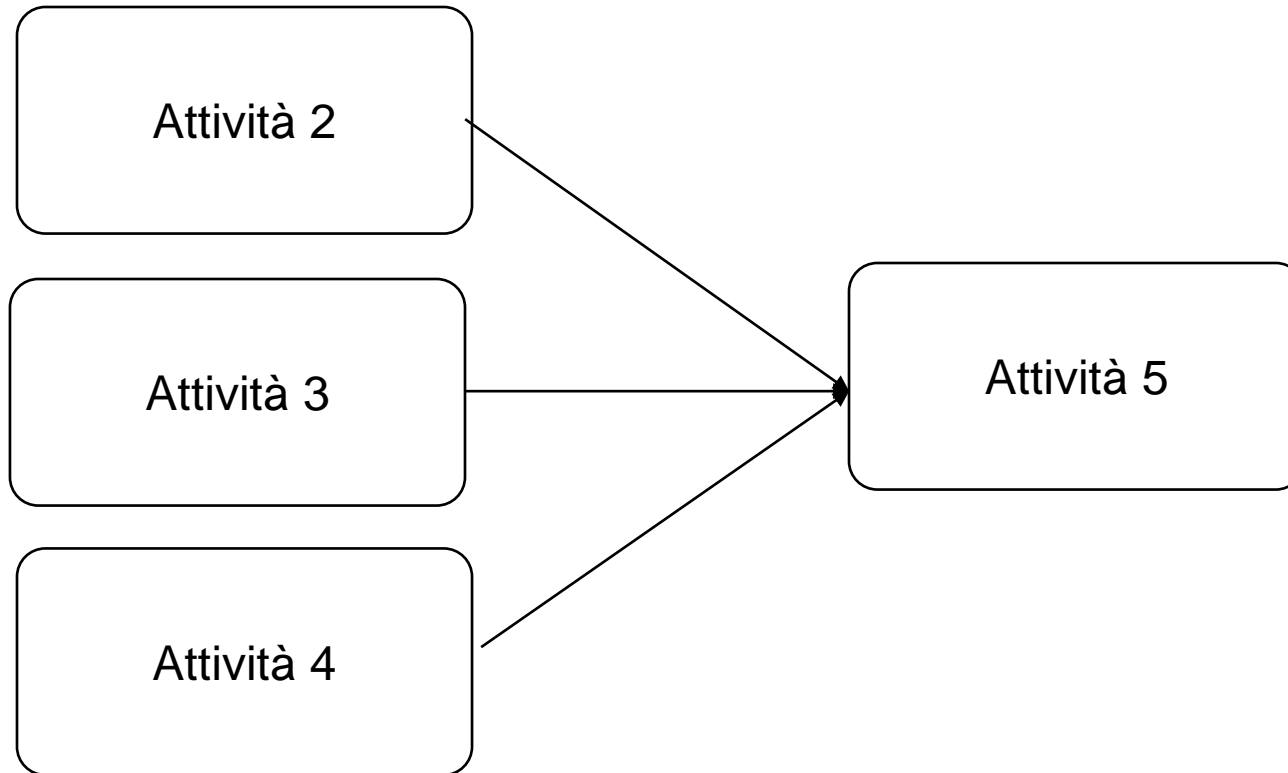
---



Un singolo flusso di attività prosegue per uno dei cammini in base al verificarsi delle condizioni di transizione, indicate fra parentesi quadre

# Tipi di AD: OR-join

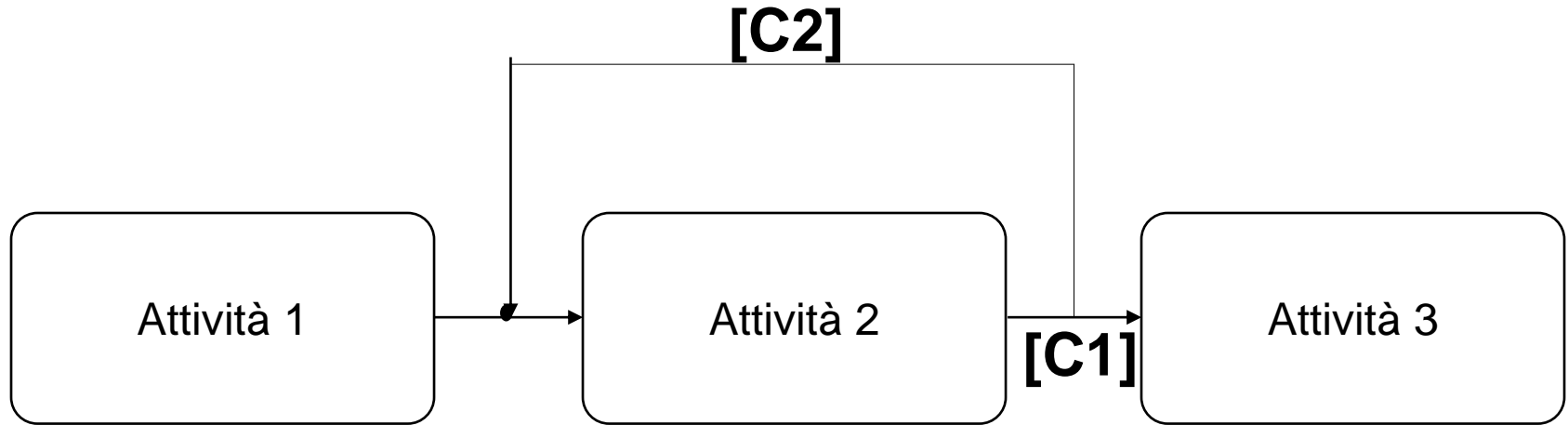
---



Un punto dove due o più flussi di attività ri-convergono in uno solo  
ovvero hanno tutti Attività 5 come elemento successivo

# Tipi di AD: iterazione

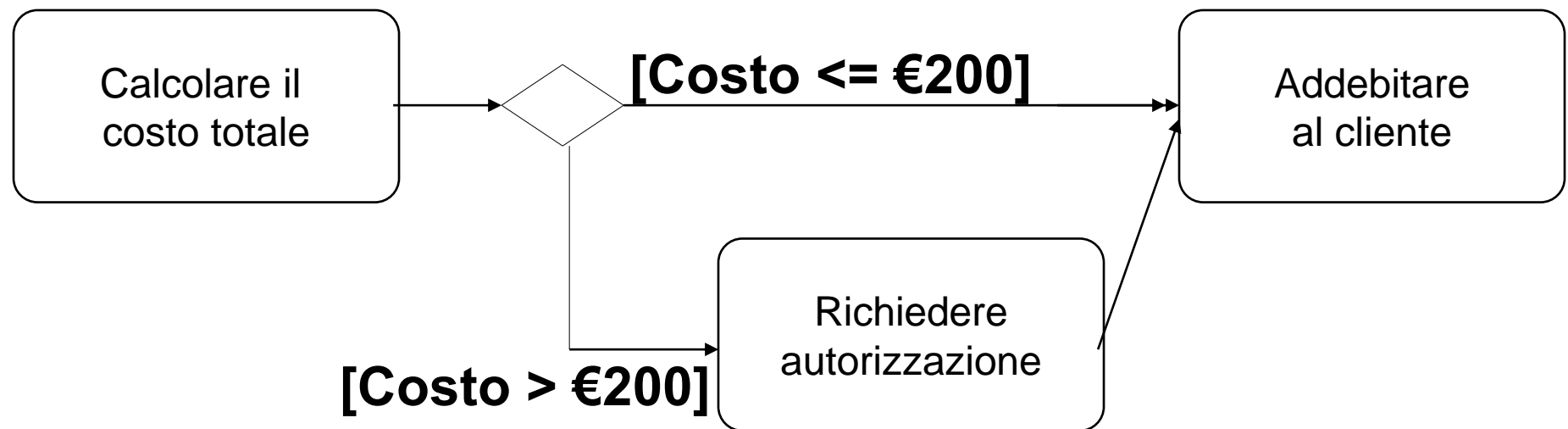
---



Un'attività (Attività 2) viene ripetuta più volte,  
in base al verificarsi o meno di opportune condizioni  
di controllo

# Activity Diagram: esempi vari

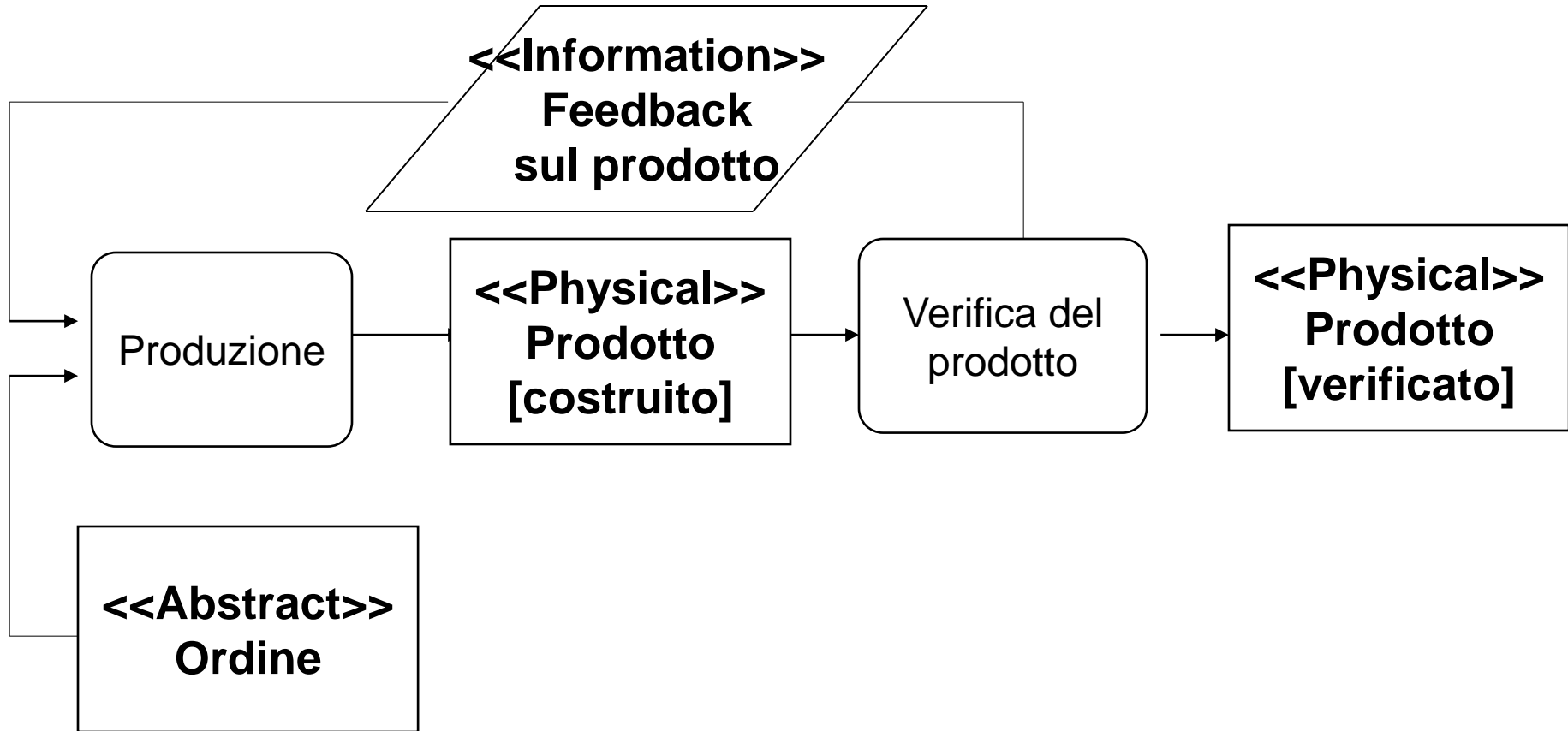
---



Se il costo totale è maggiore di 200€, bisogna chiedere l'autorizzazione prima di addebitarlo al cliente.

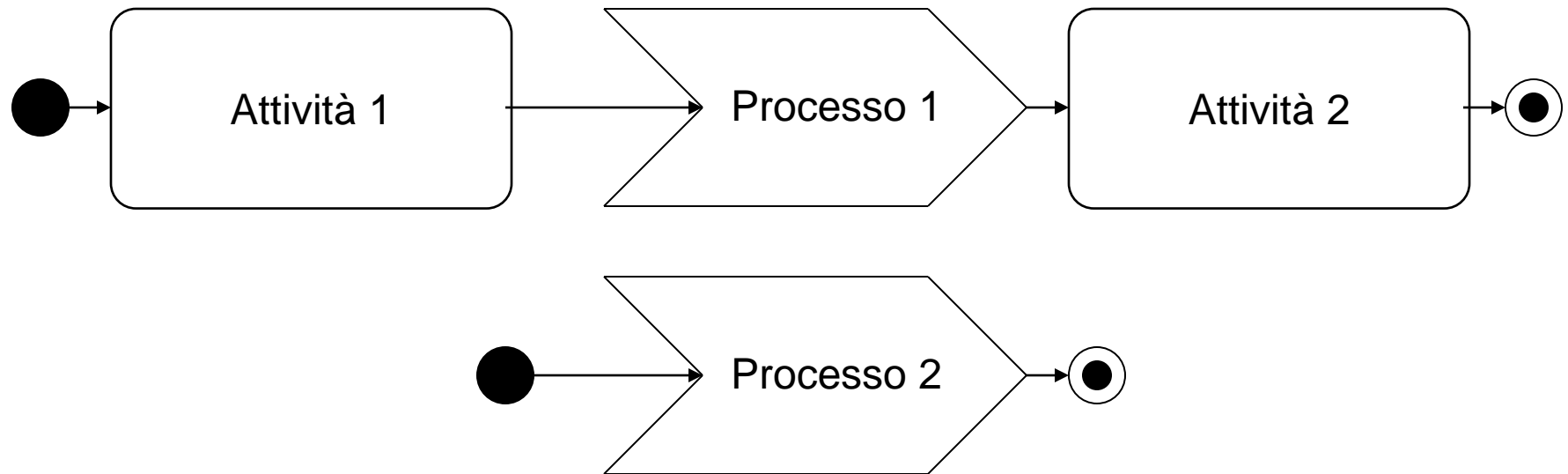
# Activity Diagram: flusso di oggetti

---



# Activity Diagram con processi

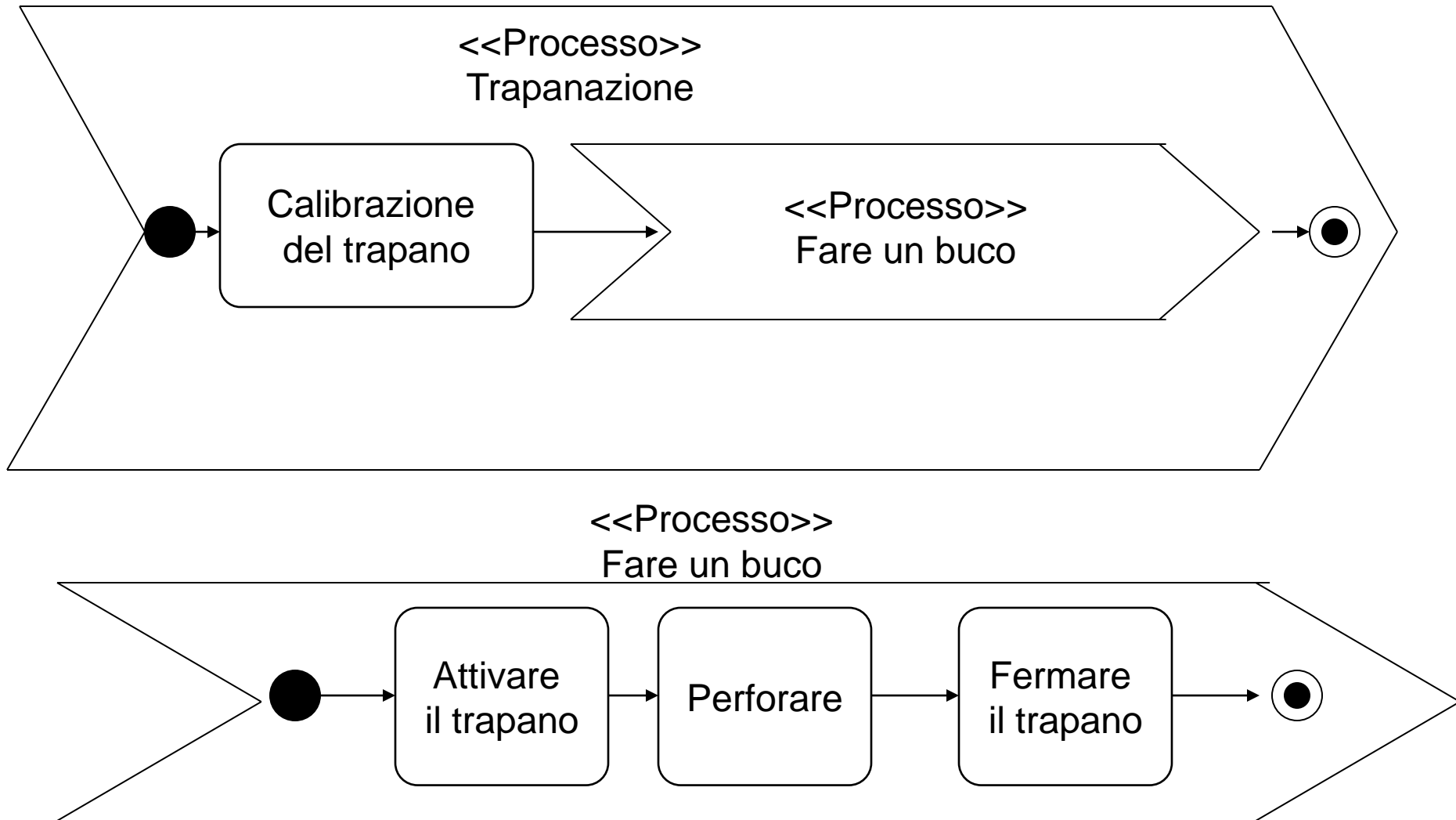
---



Nome del diagramma di attività



# Activity Diagram con processi: esempio



- **Activity Diagram**
- **Use Case Diagram**
- **Class Diagram**
- **Sequence Diagram**
- **Collaboration Diagram**
- **Statechart Diagram**

# Lo Use Case (caso d'uso)

---

- Lo use case è un contratto
- Descrive l'interazione fra due entità che interagiscono fra loro
- Consente di stabilire :
  - **Servizi forniti**
  - **Servizi richiesti**
  - **Utenti abilitati**
  - **Vincoli nell'erogazione**

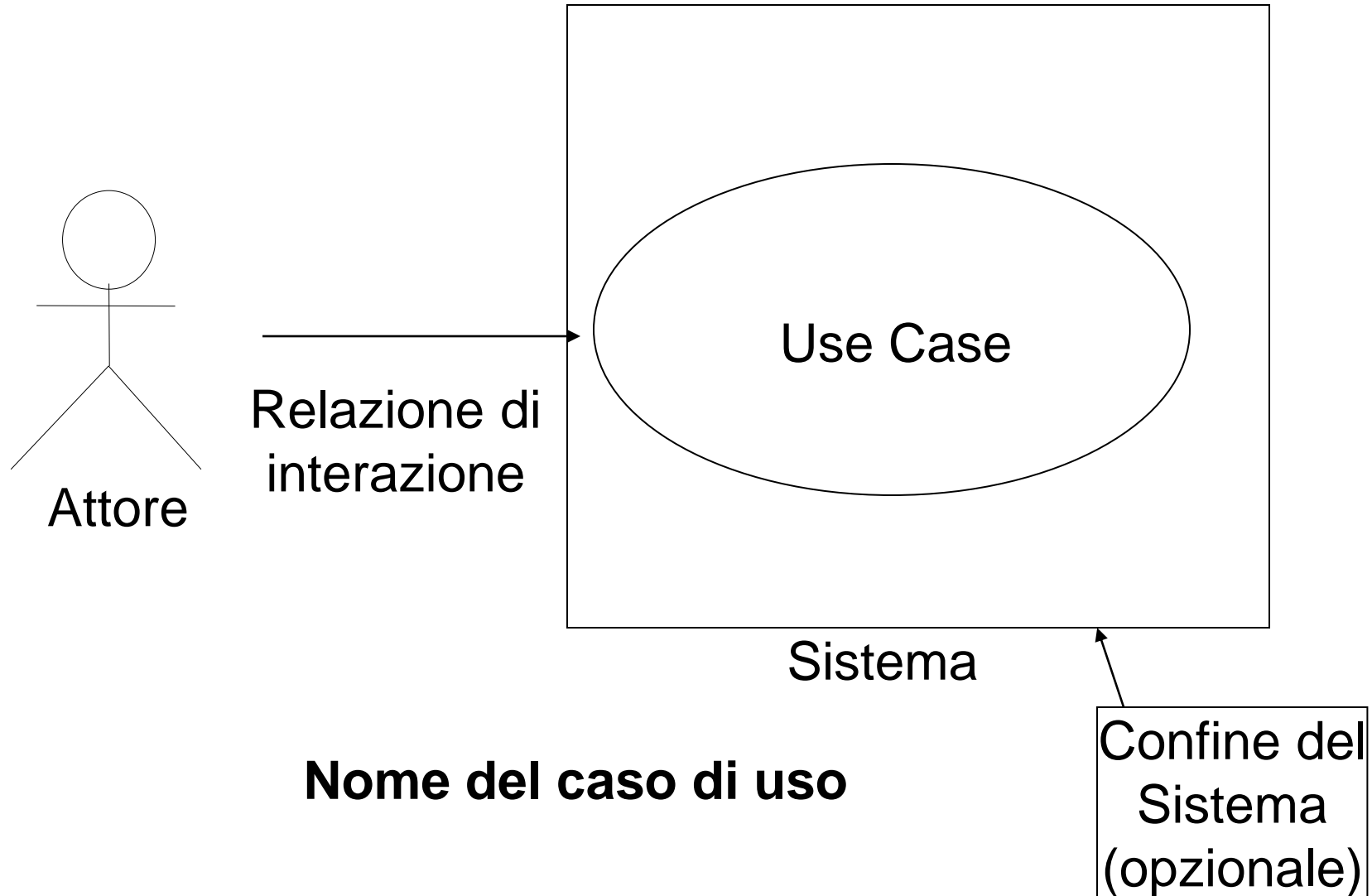
# Lo Use Case (caso d'uso)

---

- Il primo scopo è quello di trovare un confine preciso (boundary) per il sistema/sottosistema/componente che si sta analizzando
- Una volta definito il confine si può stabilire
- **cosa fa il sistema rispetto all'esterno**
- **e identificare attori e use case**

# Use Case Diagram: sintassi

---



# Definizione formale di uno Use Case

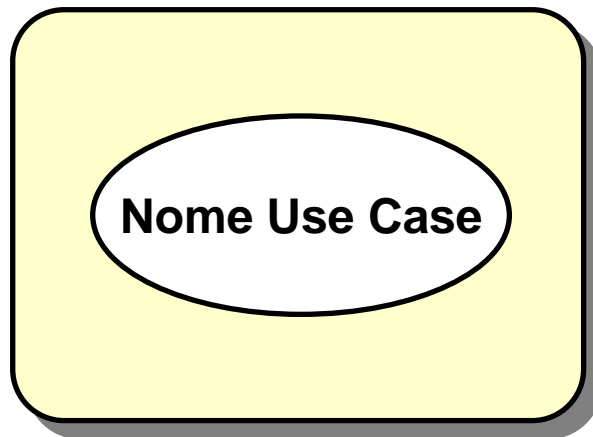
---

SEQUENZA DI TRANSAZIONI,  
eseguita da un ATTORE  
in interazione col SISTEMA,  
la quale fornisce  
un VALORE MISURABILE  
per l'attore

Definisce le richieste al sistema  
dipendenti dall'attore

# Definizione pratica di Use Case

---



- Sequenza di transazioni in dialogo col sistema
- Comporta Sempre uno o più Attori
- Rappresenta COSA (non come) il sistema offre all'attore
- Mappato alle attività di business

**Gli Use case definiscono fabbisogni di sistema  
“testabili” con una vista “fuori-dentro”**

# Cos'è un attore

---

- Un attore rappresenta un'entità esterna al sistema (una persona, un altro sistema software, un componente hardware) che interagisce col sistema
- Un attore individua un ruolo piuttosto che un'entità fisica



# Cos'è uno Use Case (caso d'uso)

---

- Uno use case rappresenta una situazione tipica di utilizzo del sistema e comprende in sé vari flussi possibili di esecuzione
- Uno use case rappresenta un'importante parte di funzionalità, completa dall'inizio alla fine

# Come si trova un attore?

---

- Un attore è un **ruolo**
- Usare le domande:
  - chi ha bisogno del sistema? (il caro vecchio: cui prodest?)
  - chi userà le funzionalità principali?
  - chi dovrà mantenere e amministrare il sistema?
  - di quali dispositivi hardware il sistema ha bisogno?
  - con quali altri sistemi il sistema dovrà comunicare?

# Come si trova uno use case?

---

Per ognuno degli attori precedentemente identificati, si può rispondere alle seguenti domande:

- quali funzioni l'attore richiede al sistema?
- l'attore ha bisogno di leggere o scrivere o immagazzinare informazioni nel sistema?
- l'attore deve ricevere notifiche di eventi dal sistema?

# Attori e Use Case

---

- Attori e use case sono sempre collegati fra loro
- un attore isolato non può interagire col sistema
- uno use case isolato non fornisce alcuna funzionalità all'esterno (stiamo parlando di funzionalità che abbiano un senso all'esterno)

# Attori e Use Case - 2

---

Un attore può essere

- **attivo**, ovvero inizia uno use case
- **passivo**, ovvero partecipa a uno use case, ma non lo inizia

# Perché usare gli Use Case?

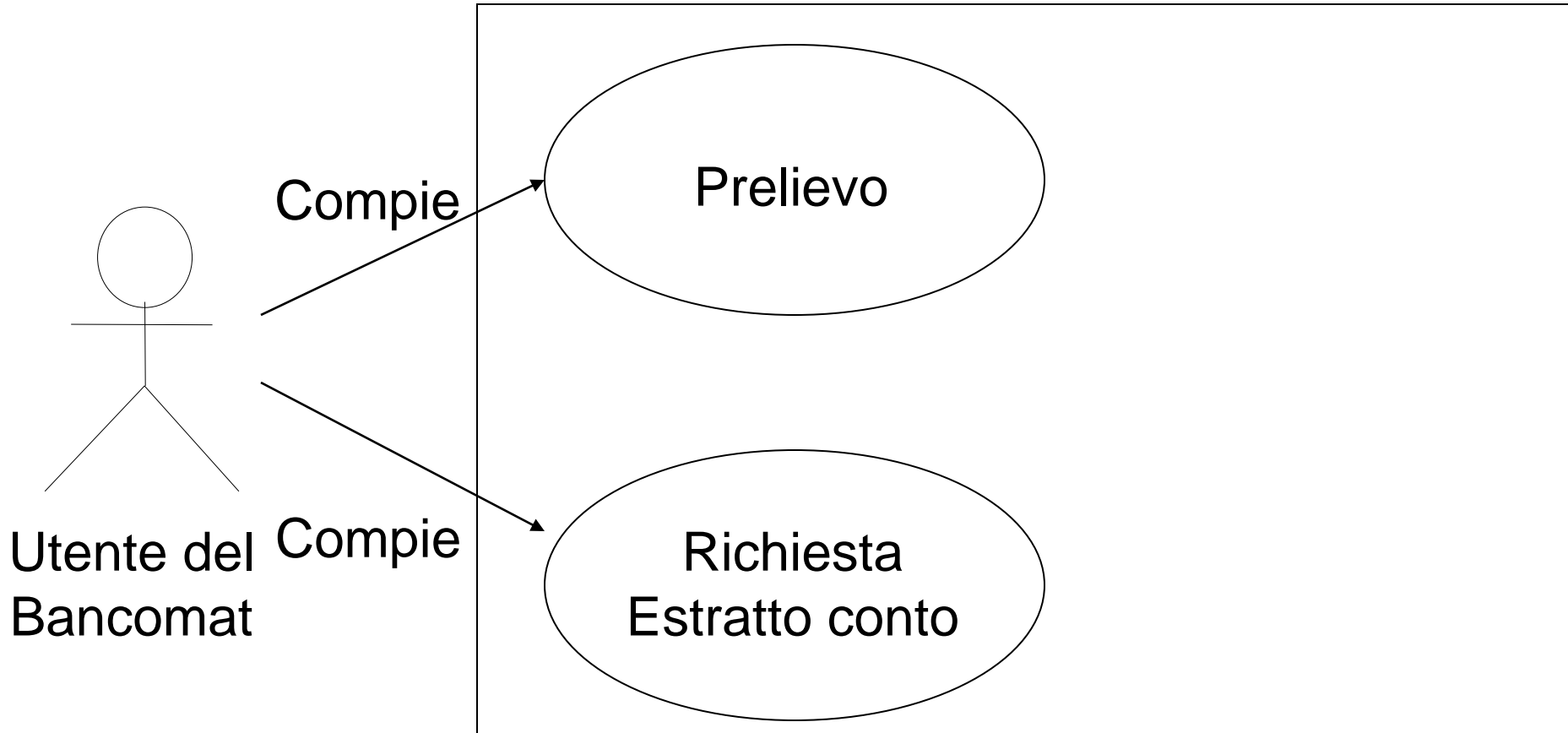
---

## PER EVITARE :

- *Utente*: non è ciò che volevo !
- *Analista*: ma come, fa tutto quello che mi hai chiesto...
- *Utente*: sì, ma non come intendevo !

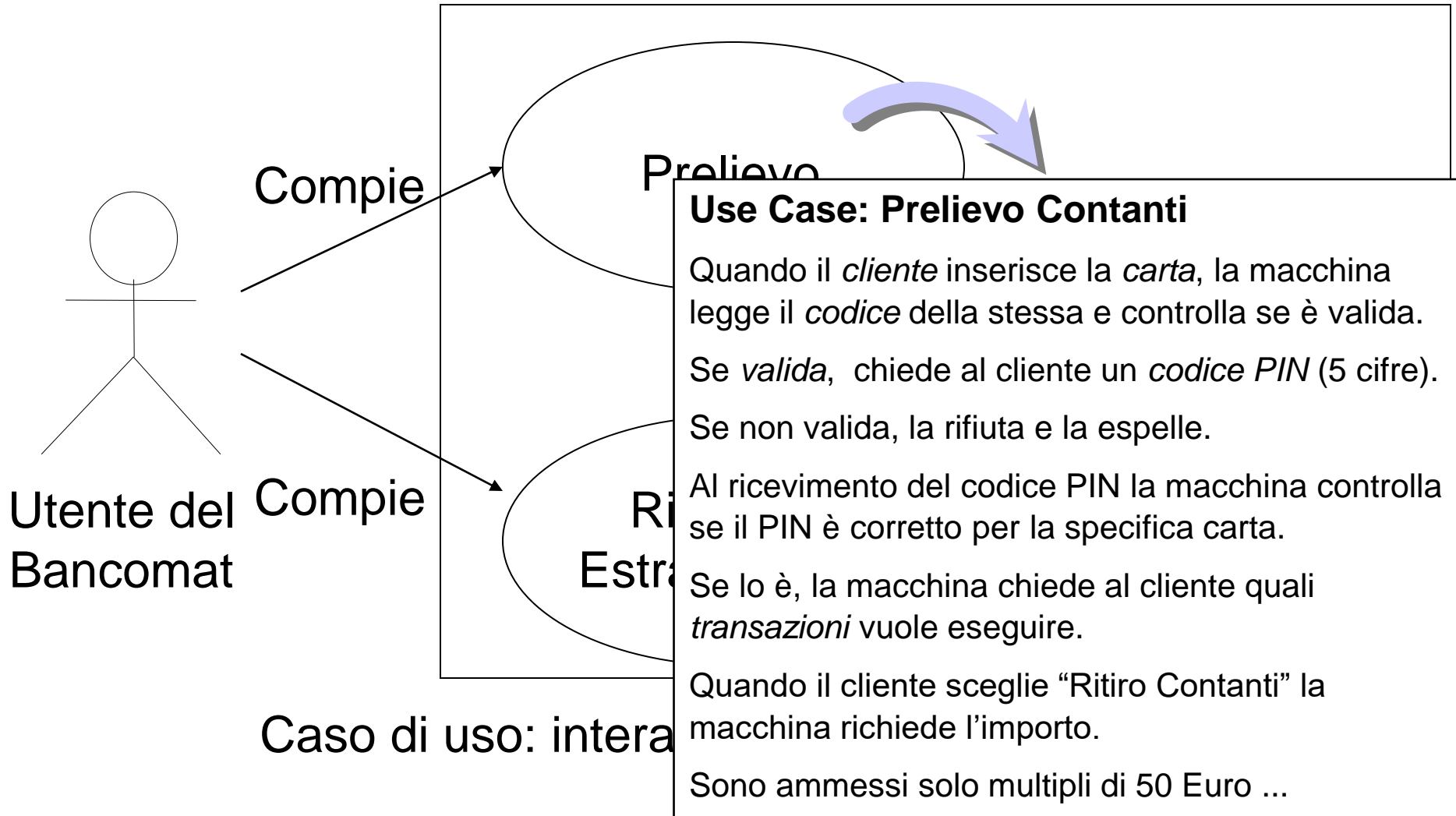
# Use Case Diagram: esempio

---



Caso di uso: interazioni con il bancomat

# Descrizioni Use Case: Specificano i Dettagli





# Use Case Diagram e Interfacce Utente

Compie

Prelievo

## Use Case: Prelievo Contanti

Quando il *cliente* inserisce la *carta*, la macchina legge il *codice* della stessa e controlla se è valida.

Se *valida*, chiede al cliente un *codice PIN* (5 cifre).

Se non valida, la rifiuta e la espelle.

Al ricevimento del codice PIN la macchina controlla se il PIN è corretto per la specifica carta.

Se lo è, la macchina chiede al cliente quali *transazioni* vuole eseguire.

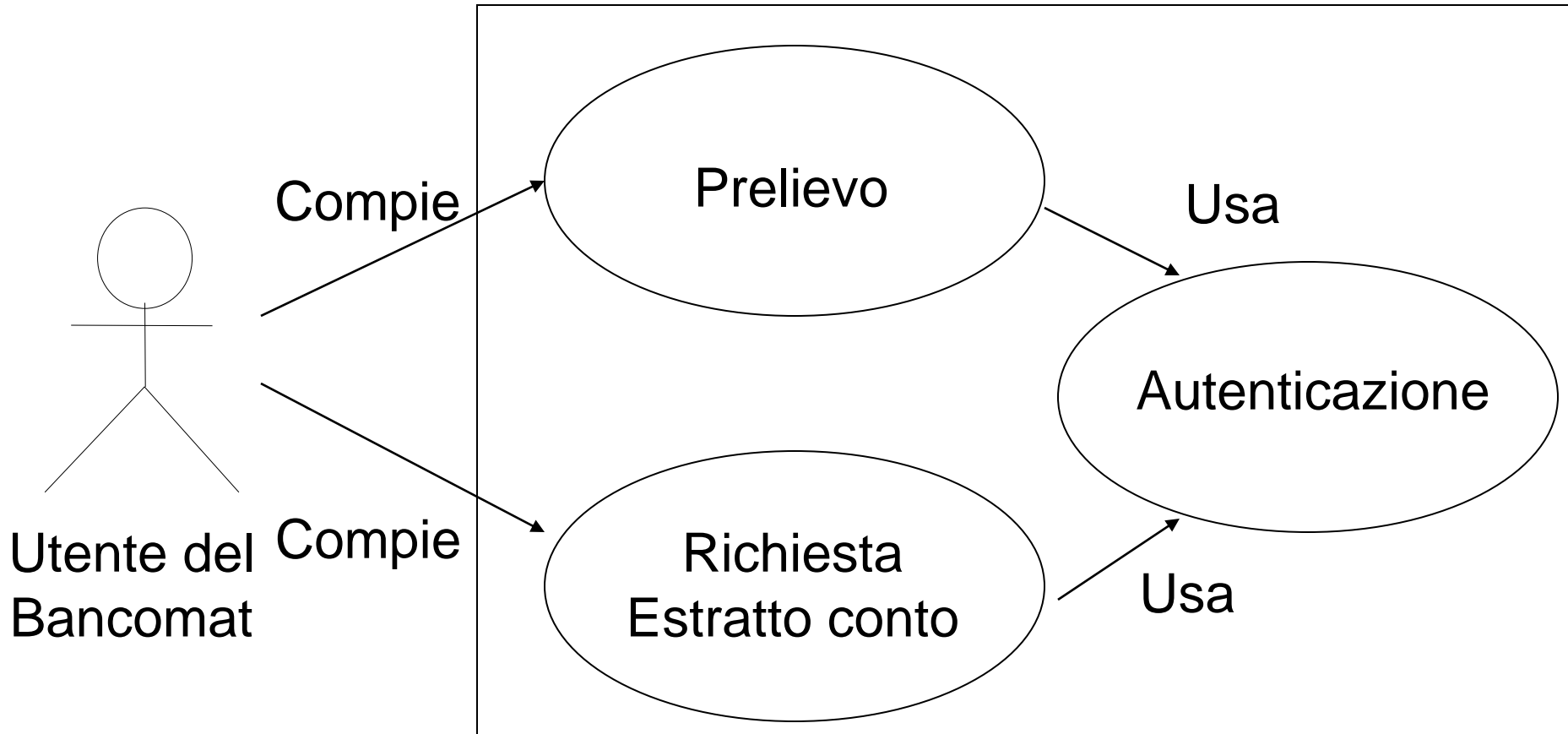
Quando il cliente sceglie “Ritiro Contanti” la macchina richiede l’importo.

Sono ammessi solo multipli di 50 Euro ...



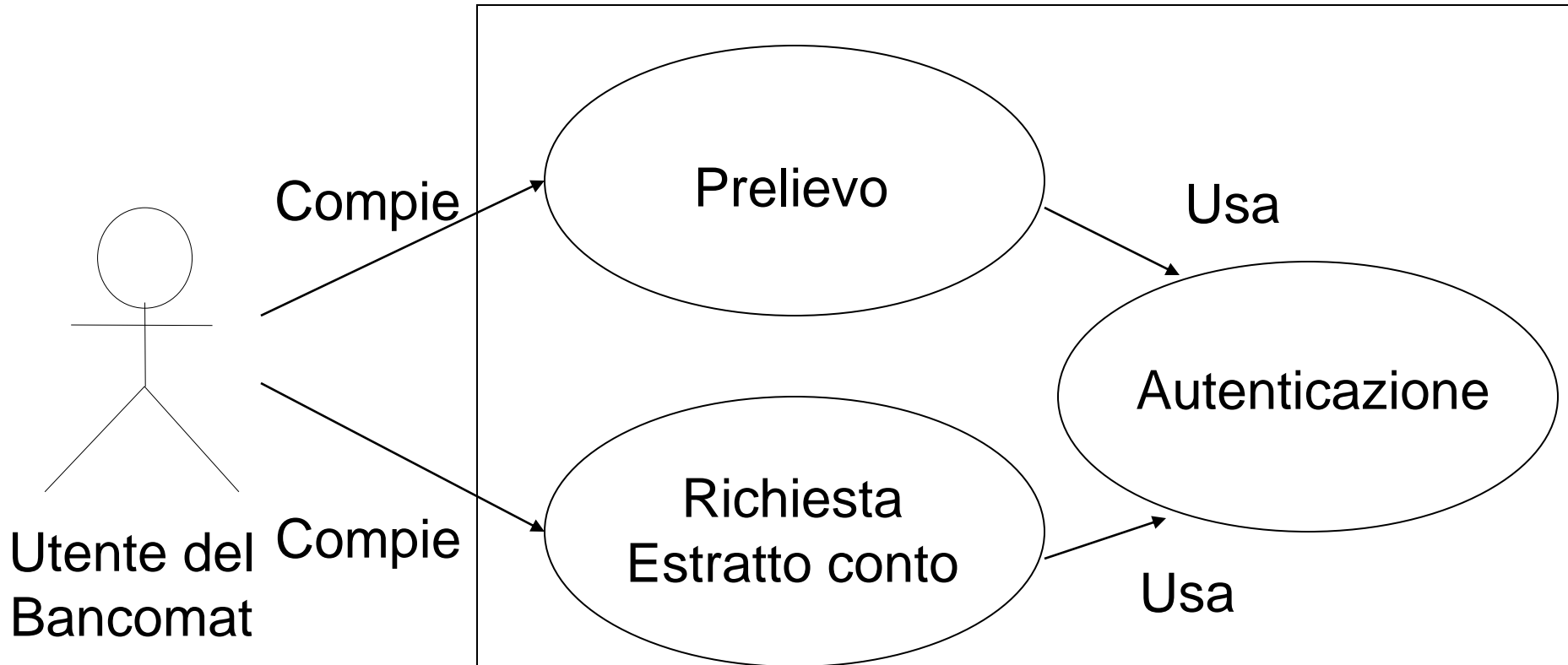
**Da uno Use Case può derivare una Interfaccia utente**

# Use Case Diagram: scomposizione



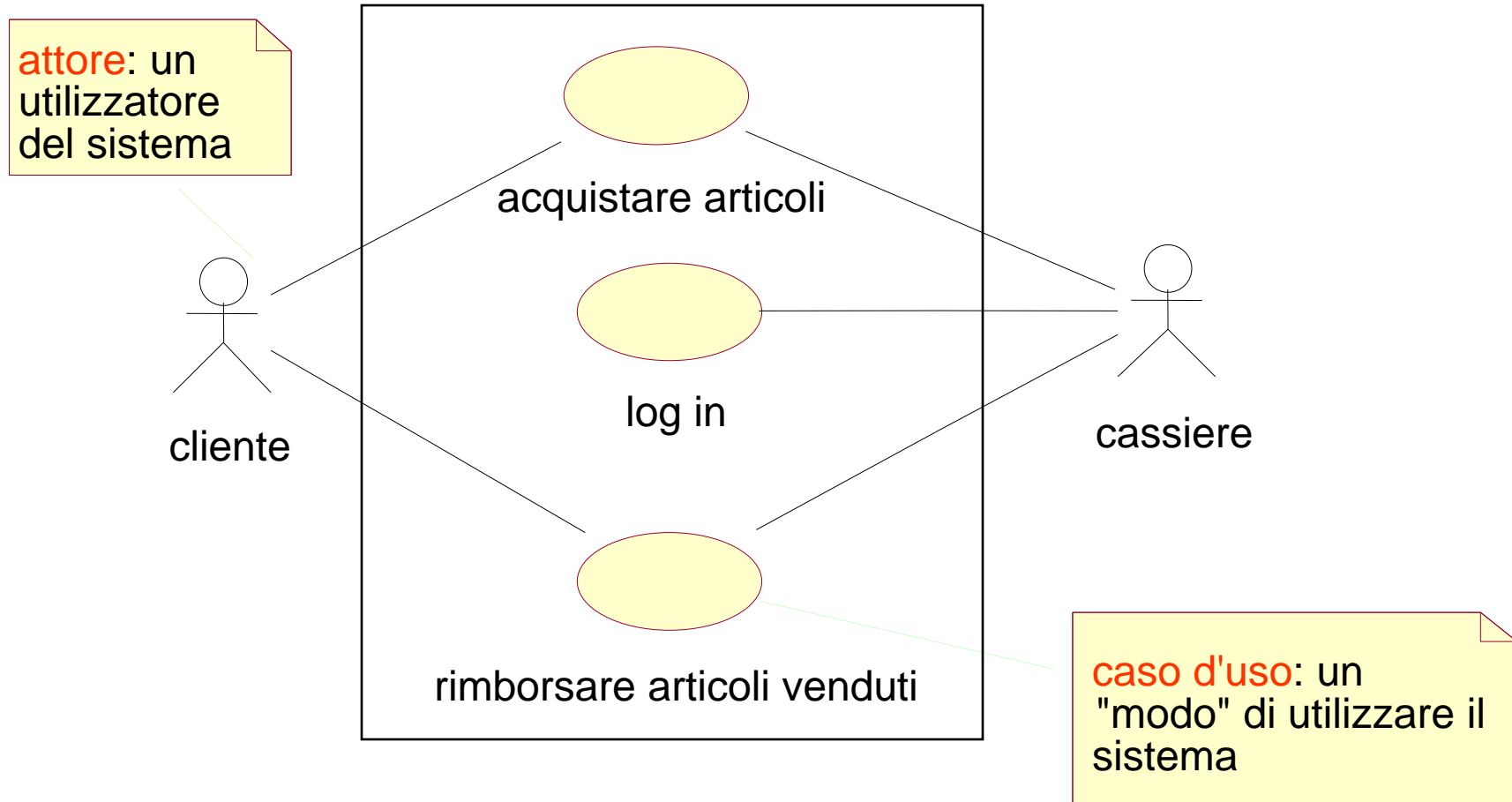
Caso di uso: interazioni con il bancomat

# Modello dei rapporti tra Attori e Use Case



- ◆ **Attori rappresentati da una persona**
- ◆ **Use case = ellissi**
- ◆ **Astrazione delle interazioni tra Attori ed uno use case**
- ◆ **Il sistema è circondato dal riquadro**

# Esempio di Use Case: la cassa



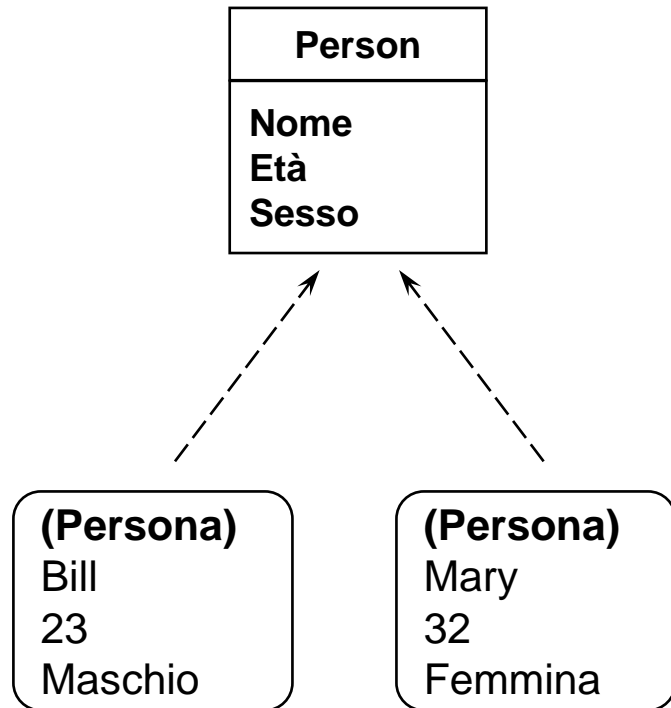
- **Activity Diagram**
- **Use Case Diagram**
- **Class Diagram**
- **Sequence Diagram**
- **Collaboration Diagram**
- **Statechart Diagram**

# Gli oggetti

---

- Gli oggetti sono proiezioni delle entità del mondo reale (del dominio del problema) entro il dominio dell'applicazione software
- In questa sede vengono usati come strumenti di modellizzazione per l'approccio OO

# Gli Oggetti sono Istanze delle Classi



**Classificazione: aspetto chiave  
dell'Object-Oriented**

## • Classe ...

- **Astrae le proprietà essenziali degli oggetti**
- **Definisce gruppi di oggetti con proprietà, comportamento, interazioni e semantica simili**
- **Fornisce un "modello" per costruire un oggetto** (è una "fabbrica delle istanze")
- **Definisce le operazioni fornite dalle istanze**

## • Oggetti ...

- **Sono definiti dalla loro classe**
- **Hanno un valore intrinseco per ogni proprietà definita dalla loro classe**
- **"Conoscono" la loro classe**
- **Possono essere creati "al run time"**

# I Servizi forniti dagli Oggetti sono le Operazioni (metodi)

---

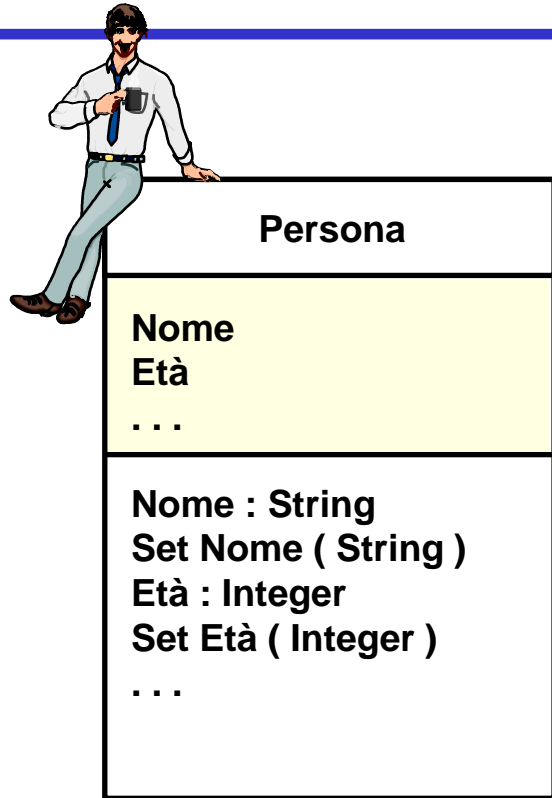
## Metodi (pubblici):

- Forniscono un comportamento osservabile
- I metodi definiscono una 'signature' (firma)
  - Nome
  - Parametri Opzionali
  - Valori ritorno Opzionali

Sega Circolare
Produttore ? Numero di Serie ?
Accendi Spegni Blocca Imposta Angolo( gradi ) Imposta Profondità ( cm ) ...



# Attributi: sono nascosti all'interno degli Oggetti



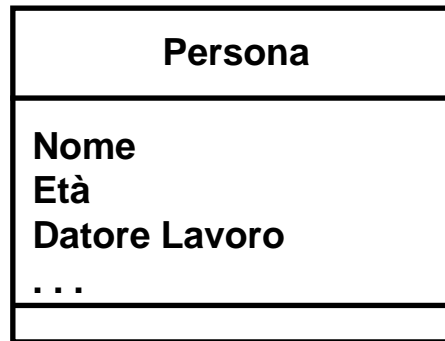
## Attributi

- Nascosti agli altri oggetti
- Vi si accede tramite le operazioni
- Riduzione della possibilità di corruzione casuale
- Ridotti gli effetti “ricaduta” delle modifiche

Le Operazioni di accesso all'attributo sono solitamente implicite, non evidenziate sui diagrammi delle classi

# Attributi sono generalmente 'primitivi'

---



L' Attributo 'Datore Lavoro' andrebbe modellato come oggetto 'Società' col quale 'Persona' interagisce?

- Attributi che hanno complessità intrinseca sono a loro volta Oggetti
  - Oggetti collegati, non più attributi
    - Gli Attributi non hanno struttura interna
- I Valori di Attributo non hanno identità unica
  - Età '23' NON è unica

# Le classi

---

- Una classe rappresenta una entità di business coinvolta nell'elaborazione (class diagram di business)
- L'individuazione delle classi consente di identificare chi fa che cosa e come all'interno del sistema/azienda
- La struttura delle classi aziendali relative ad un processo può essere confrontata con la definizione di strutture efficienti per la realizzazione (pattern)

# Le classi

---

- Una classe può rappresentare un oggetto concreto (un tornio) oppure immateriale (Iva) o una attività (spedizione)
- Una classe ha delle caratteristiche che la descrivono (attributi), ha delle elaborazioni che vengono eseguite (operazioni), rappresenta una definizione applicabile ad un insieme di elementi omogenei (oggetti)

# Le classi

---

- Le classi sono proiezioni delle entità del dominio del problema entro il dominio del modello
  - **Una classe definisce gruppi di entità con proprietà, comportamento, interazioni e semantica simili**
  - **Una classe fornisce un “modello” per “costruire” ciascuna di tali entità**

# Le classi

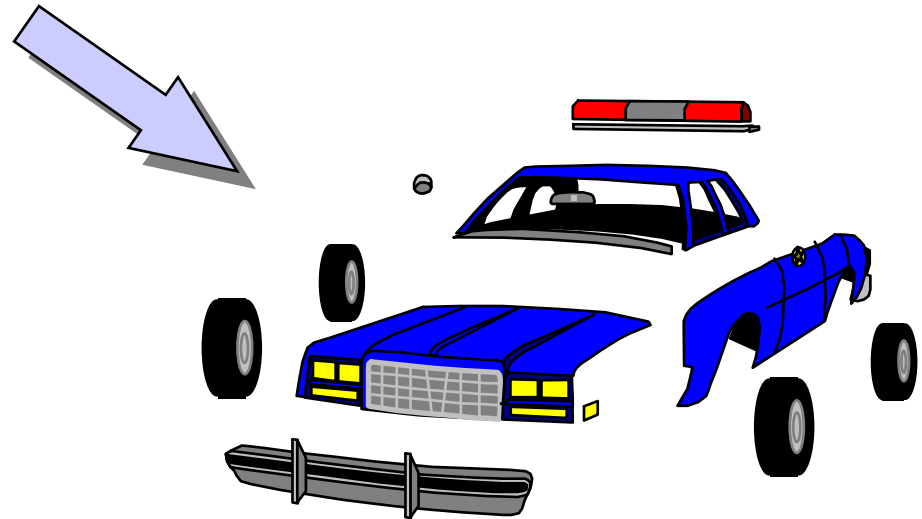
---

- Classe
  - Prototipo
  - Modello
  - Astrazione
  - Definisce Proprietà e comportamento degli oggetti

# Le classi hanno degli attributi

---

Colore cofano  
Tipo fanali...  
Portiera  
Pneumatico

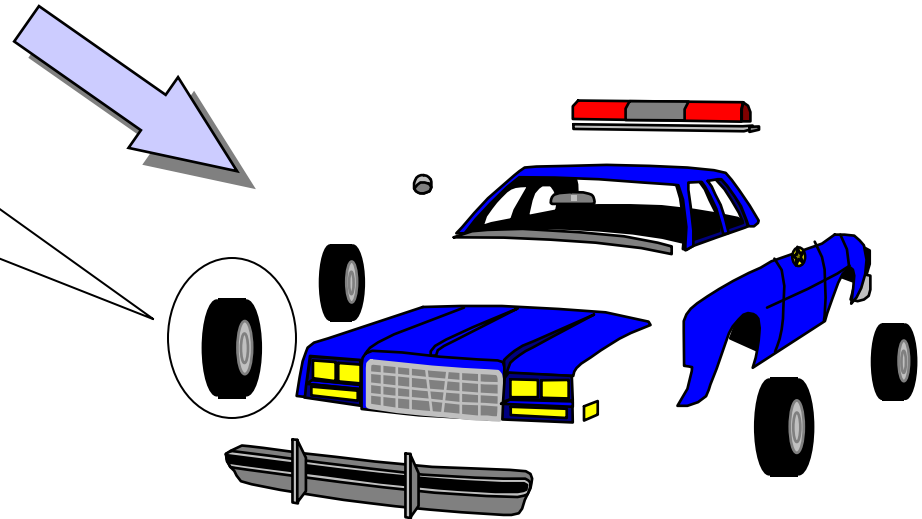


# Le classi possono avere componenti

---

## Pneumatico

Dimensione  
Marca  
Tipo (neve, bagnato...)





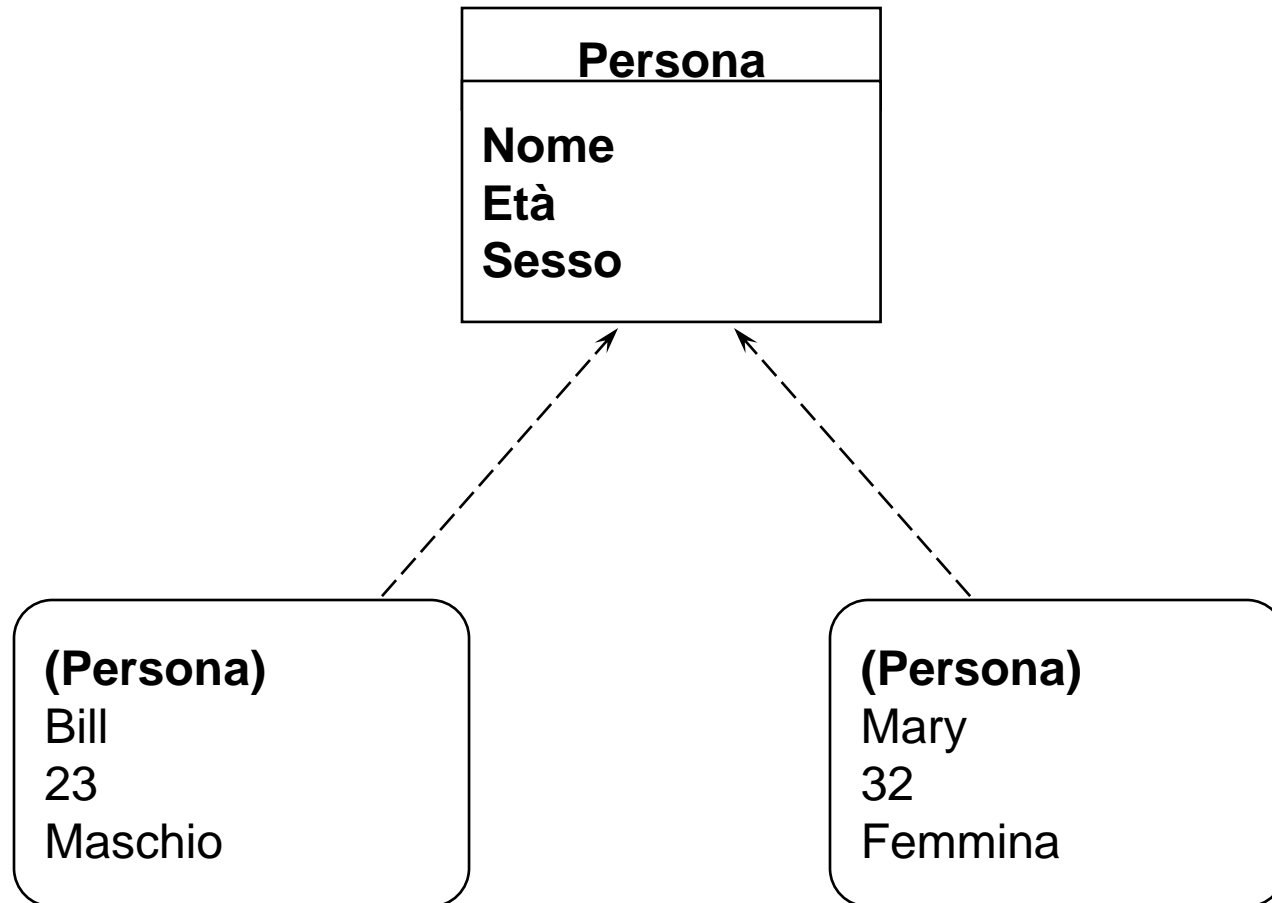
# Oggetti e classi

---

- **Classe ...**
  - **Analoga ad una definizione presente in un dizionario o un'enciclopedia**
  - **Esempio: l'auto della polizia**
- **Oggetti ...**
  - **Oggetto concreto**
  - **Esempio: l'automobile n.12 della polizia municipale di Parma**

# Gli Oggetti sono la “concretizzazione” delle Classi

---



# I class diagram

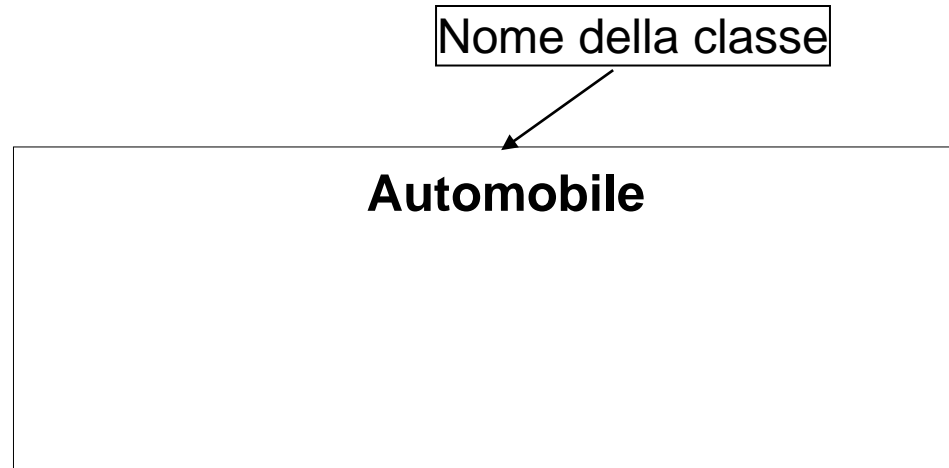
---

- Modellano la relazione fra le entità del sistema, rappresentate come classi
- Le classi possono avere relazioni fra loro, rappresentate con le *associazioni*
- Per default, un'associazione è bidirezionale, anche se può essere resa unidirezionale

# Rappresentazione di una classe:

## 1) come solo entità

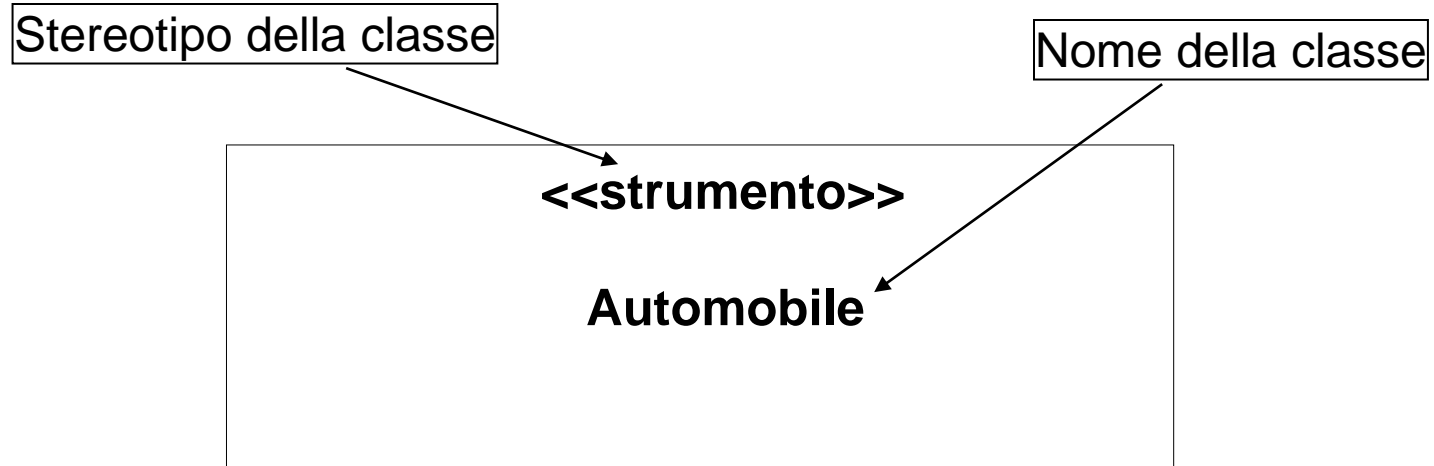
---



# Rappresentazione di una classe:

## 1a) come entità + stereotipo

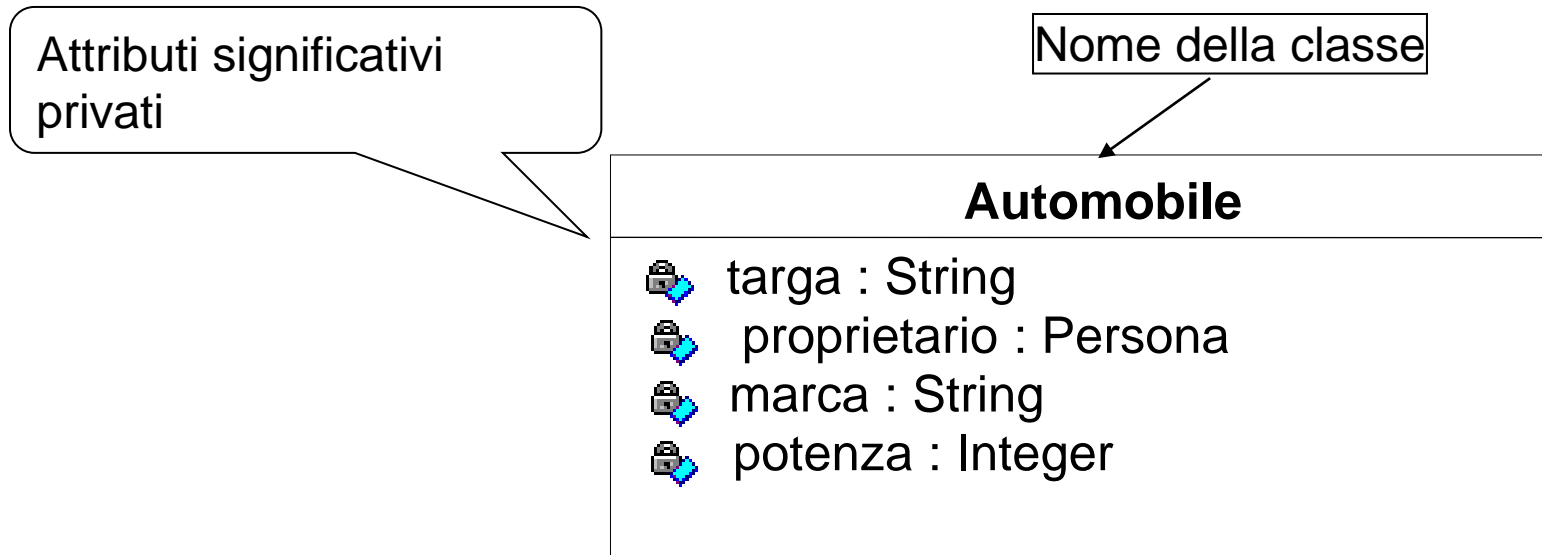
---



# Rappresentazione di una classe:

## 2) entità con attributi

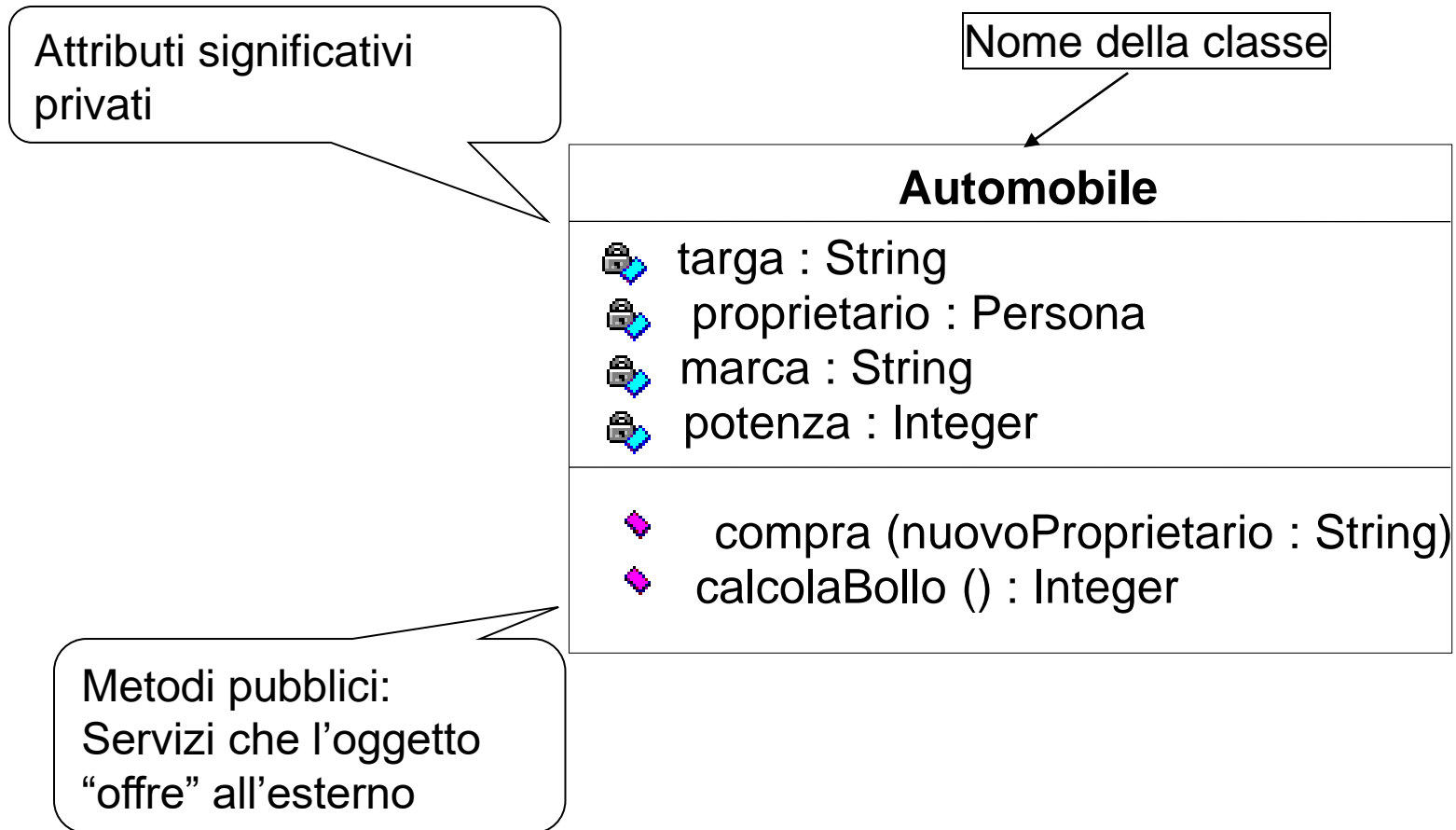
---



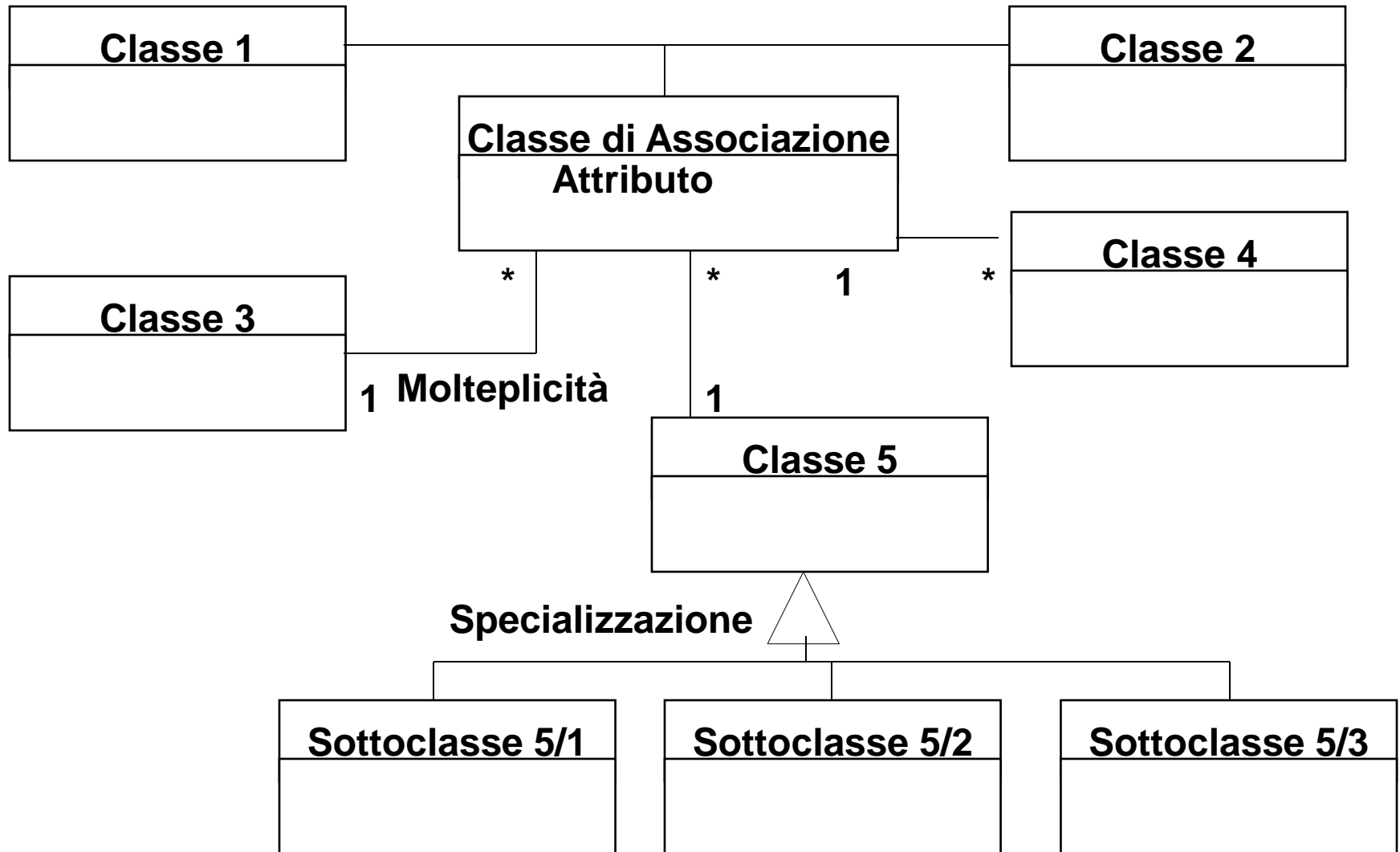
# Rappresentazione di una classe:

## 3) entità + attributi + metodi

---

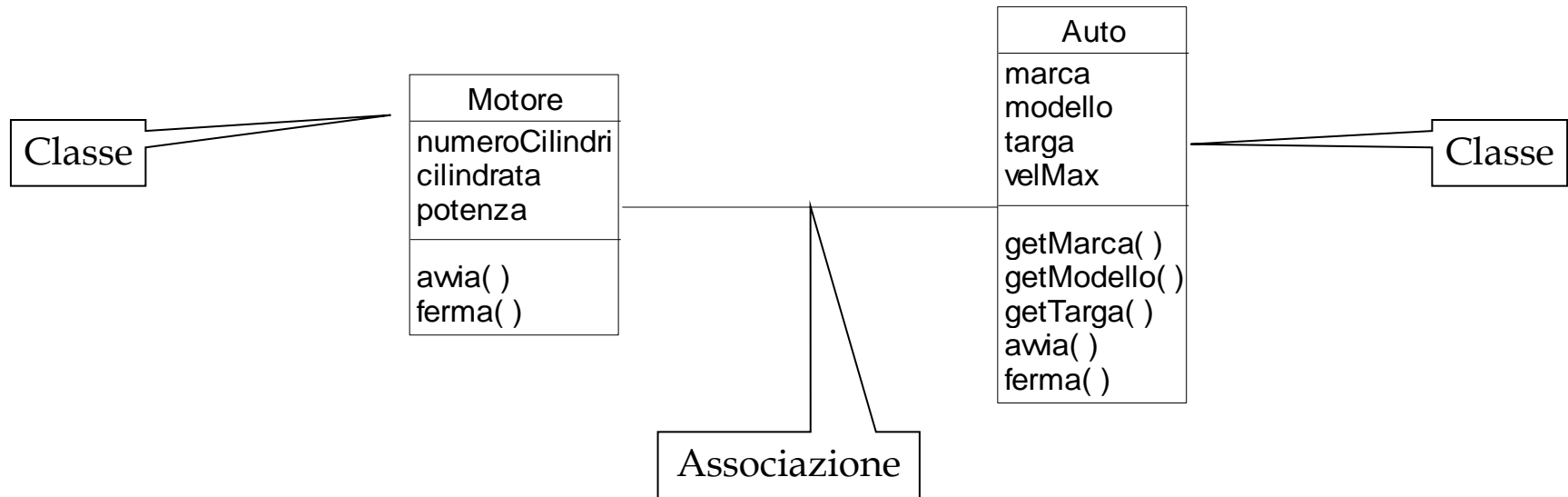


# Diagramma delle classi





# Un esempio di associazione

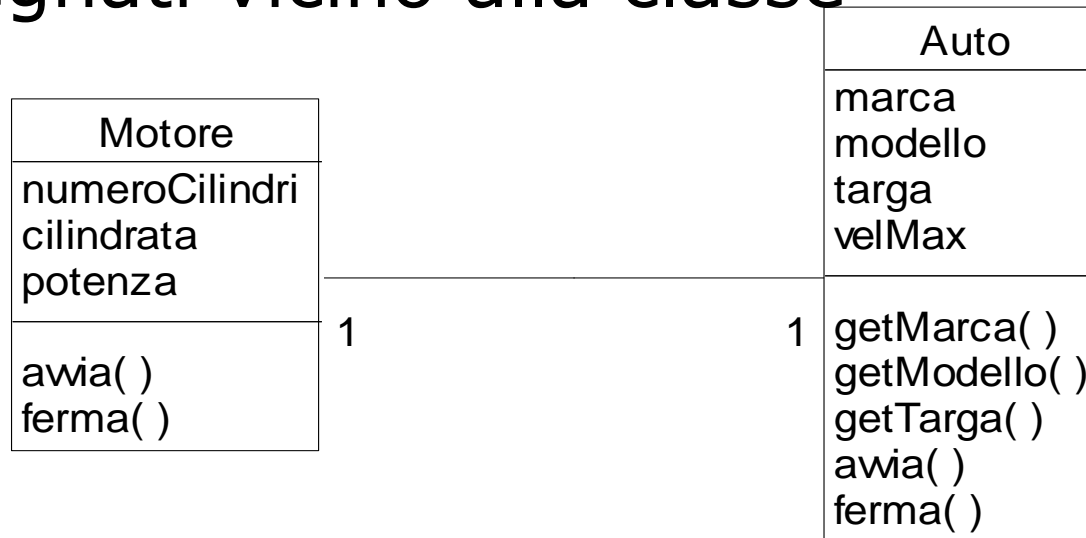


Fra le classi Auto e Motore c'è un'*associazione*: questo significa che fra ogni istanza della classe auto e ogni istanza della classe motore c'è un *link*

Oggetto : classe = link : associazione

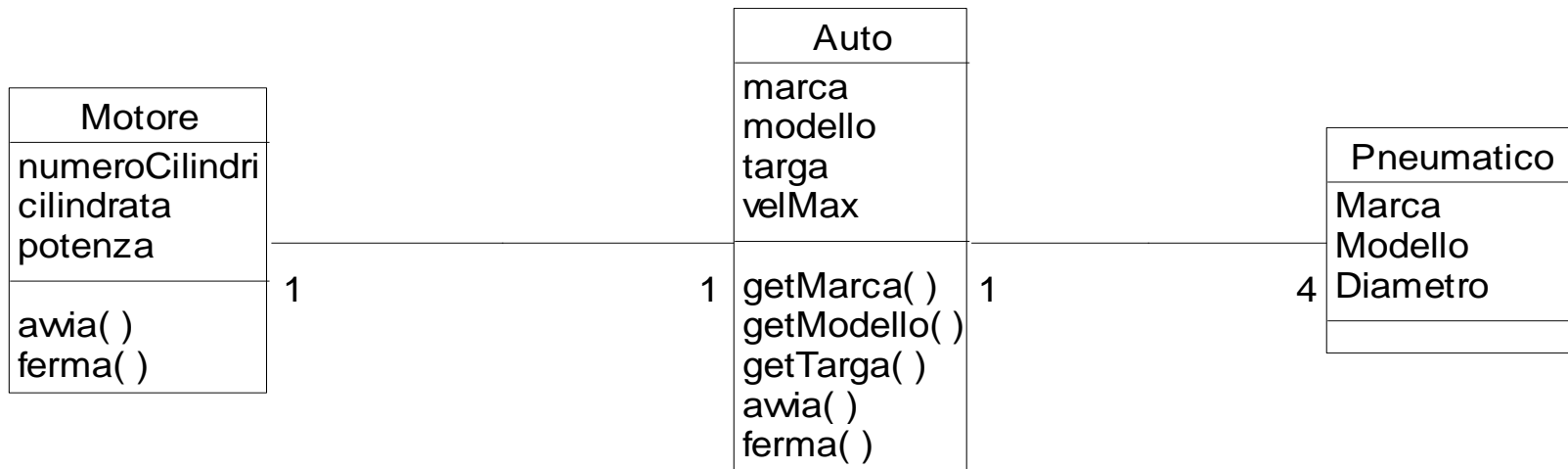
# Cardinalità

- La *cardinalità* di un'associazione esprime il numero di oggetti di una certa classe che prendono parte all'associazione
- Si esprime con un numero o un range disegnati vicino alla classe

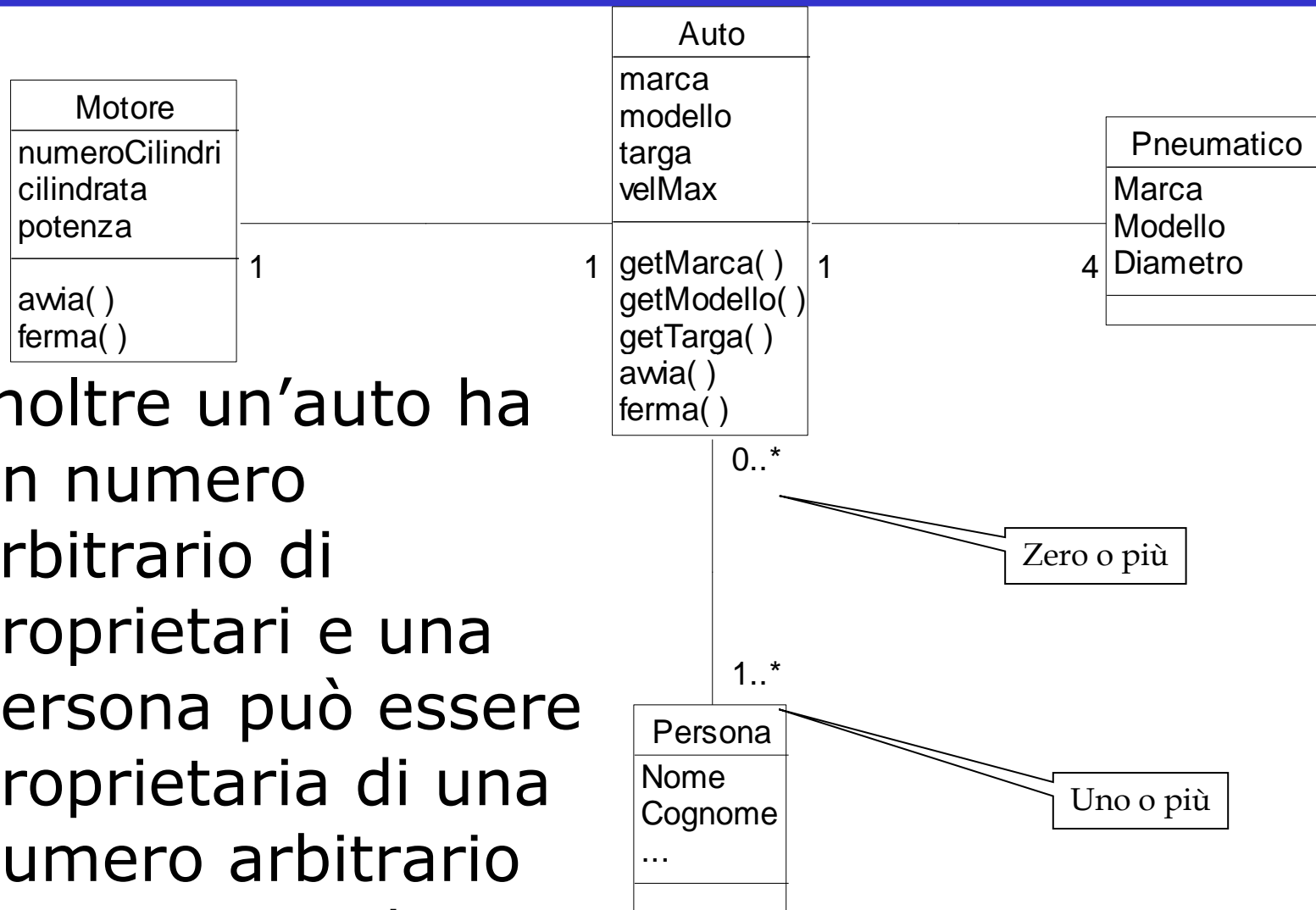


# Cardinalità - 2

- Un auto ha 4 pneumatici, per cui il diagramma diventa

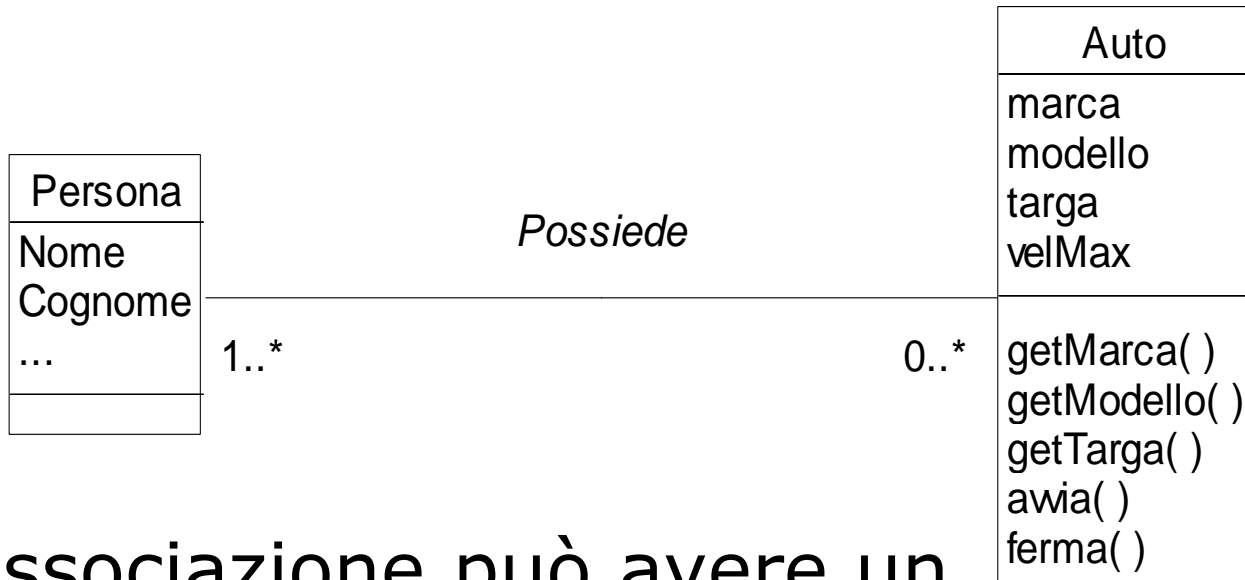


# Cardinalità - 3



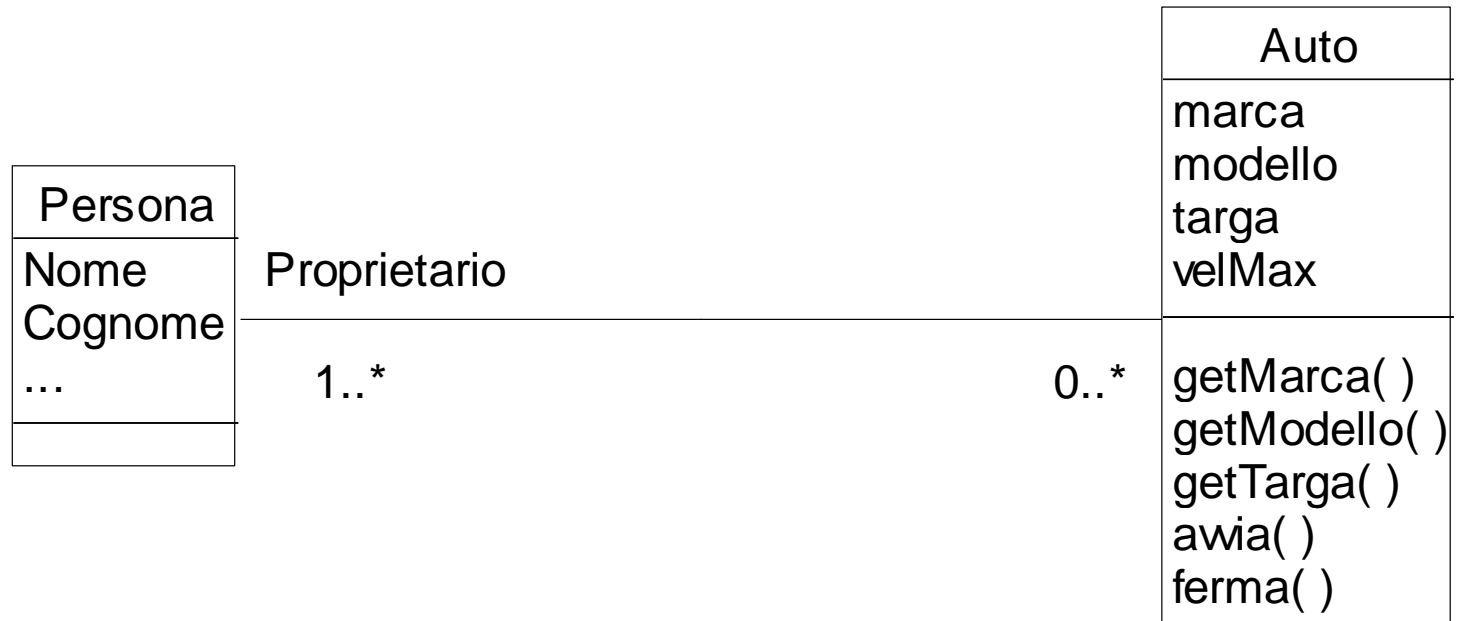
- Inoltre un'auto ha un numero arbitrario di proprietari e una persona può essere proprietaria di un numero arbitrario di auto, quindi...

# Nomi per le associazioni



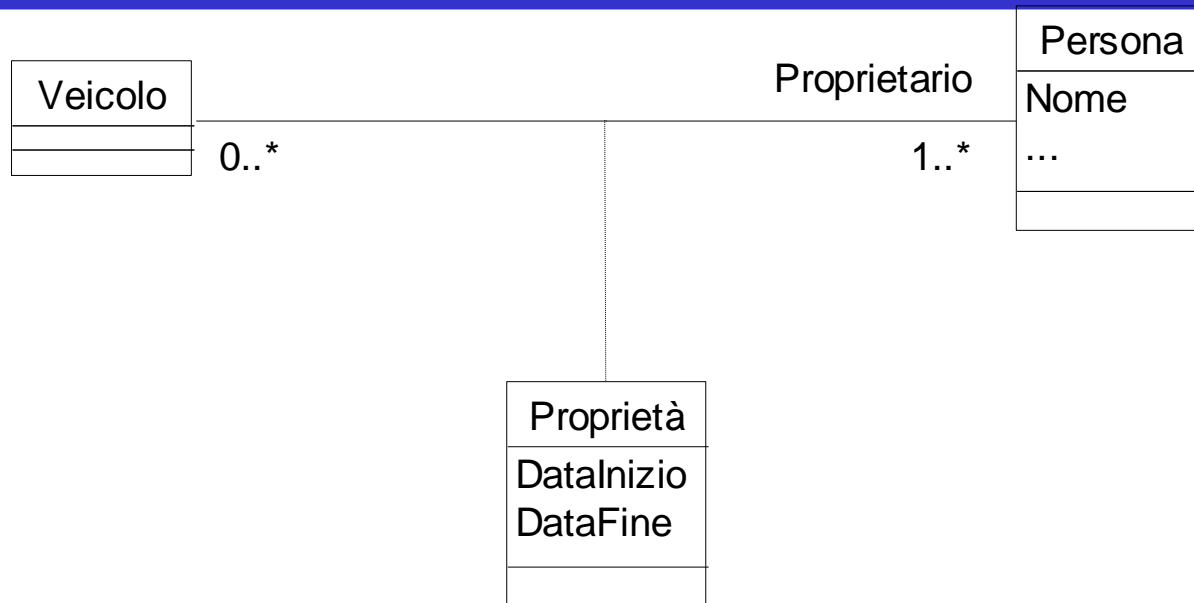
- Un'associazione può avere un nome
- Il concetto di "possiede" è però unidirezionale

# Ruoli per le classi



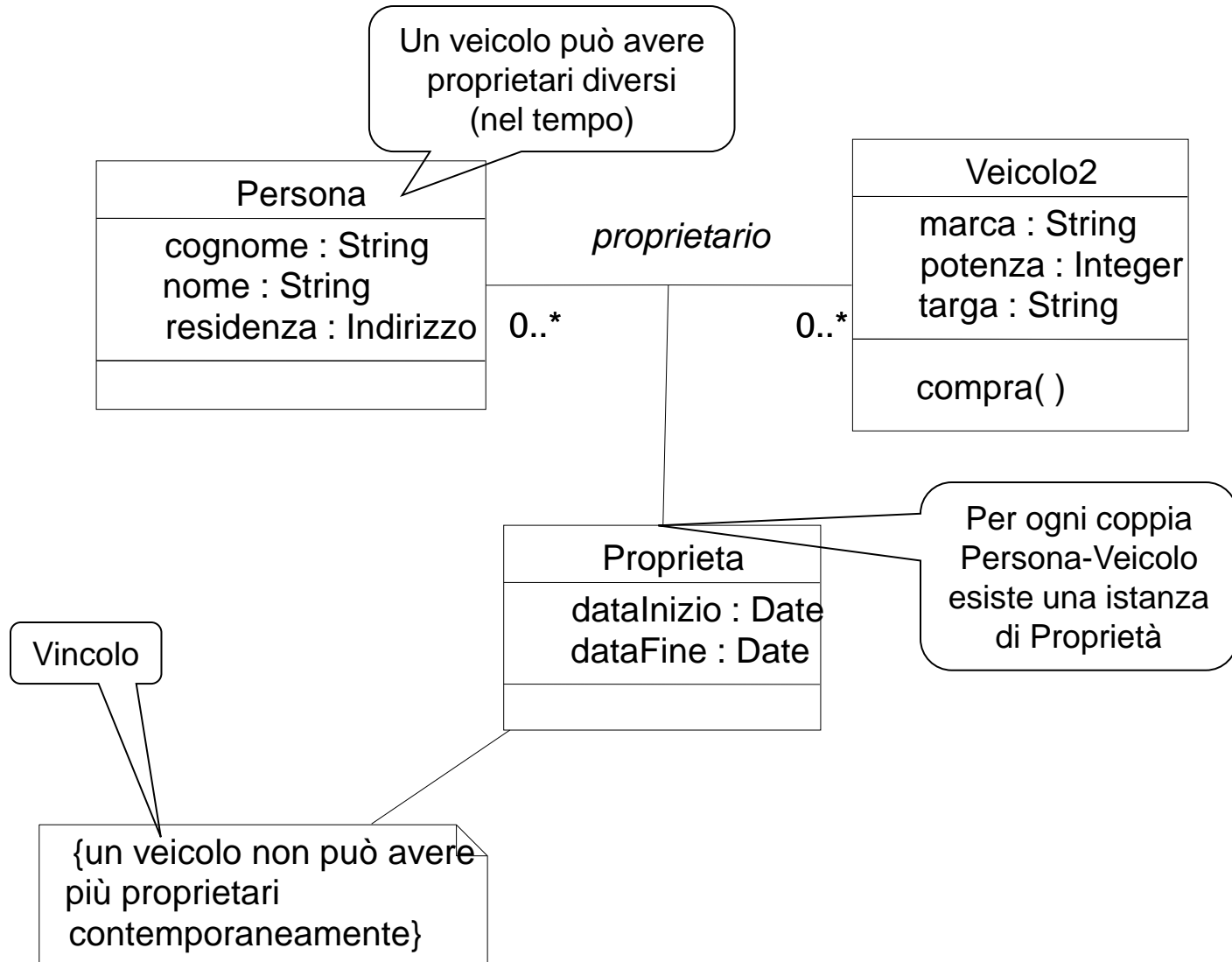
In genere si preferisce usare  
Ruoli associati a una delle classi

# Classi di associazione



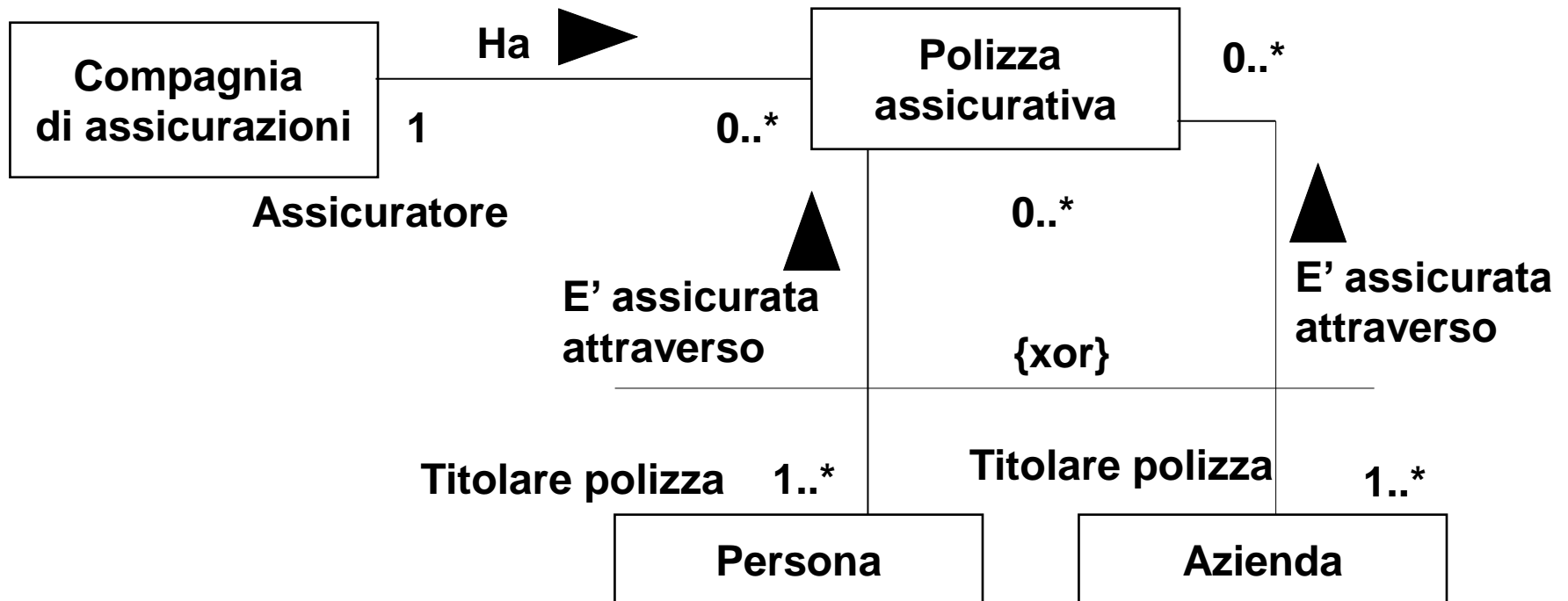
- Servono ad esprimere attributi e/o metodi propri dell'associazione
- I concetti espressi non appartengono agli oggetti associati ma all'associazione in quanto tale

# Classi di associazione e vincoli

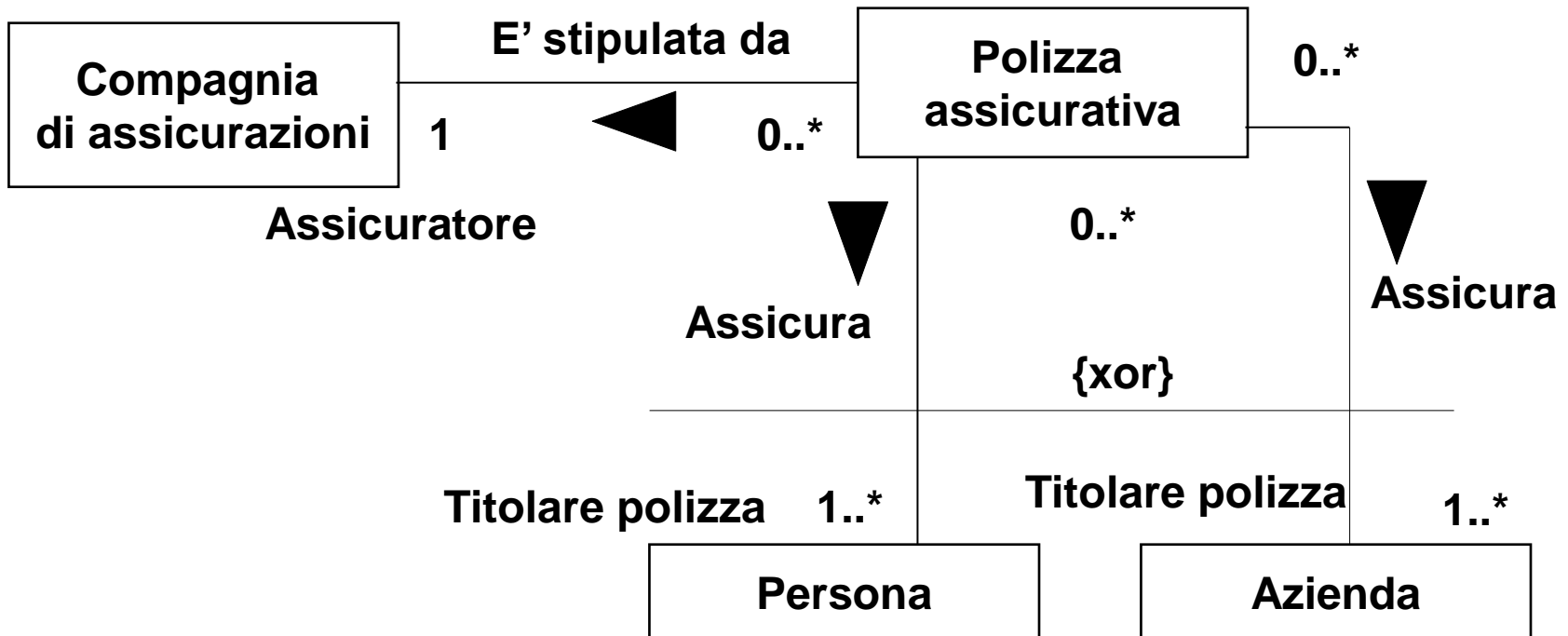




# Dettagli delle associazioni: le relazioni



# Dettagli delle associazioni: le relazioni

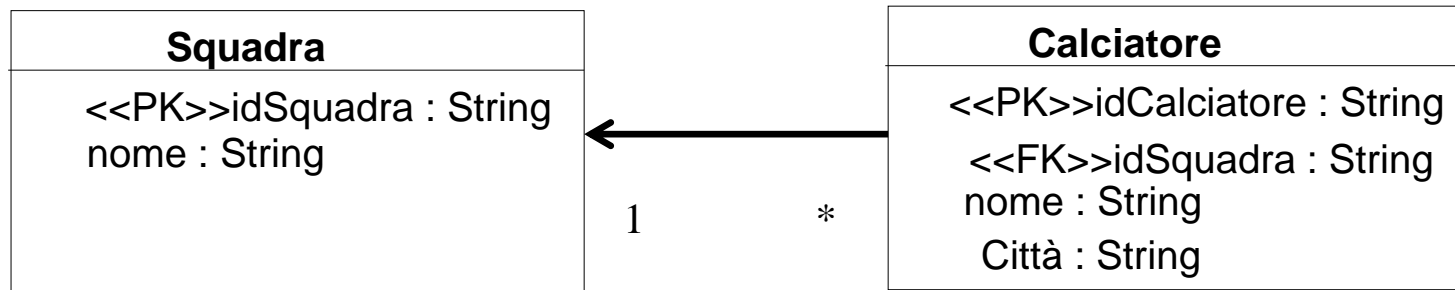
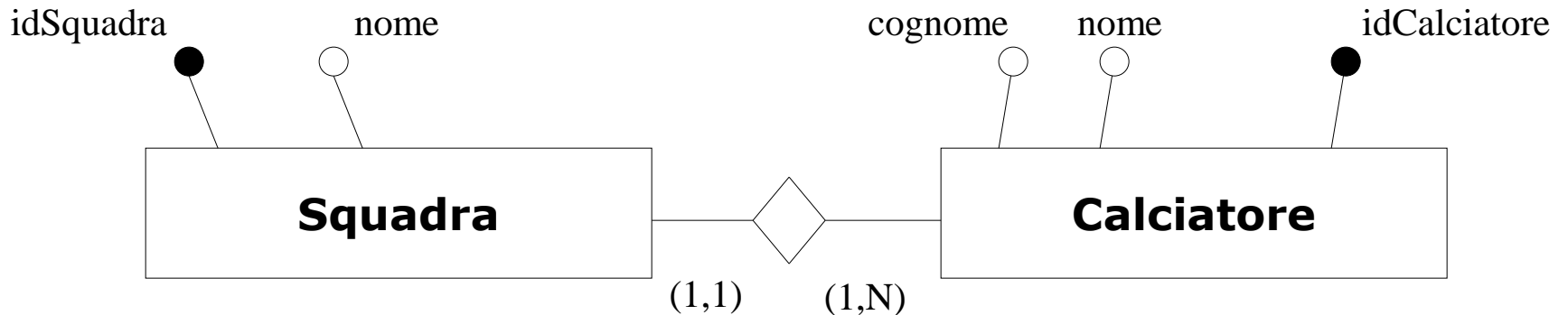


# Class diagram e diagrammi E-R

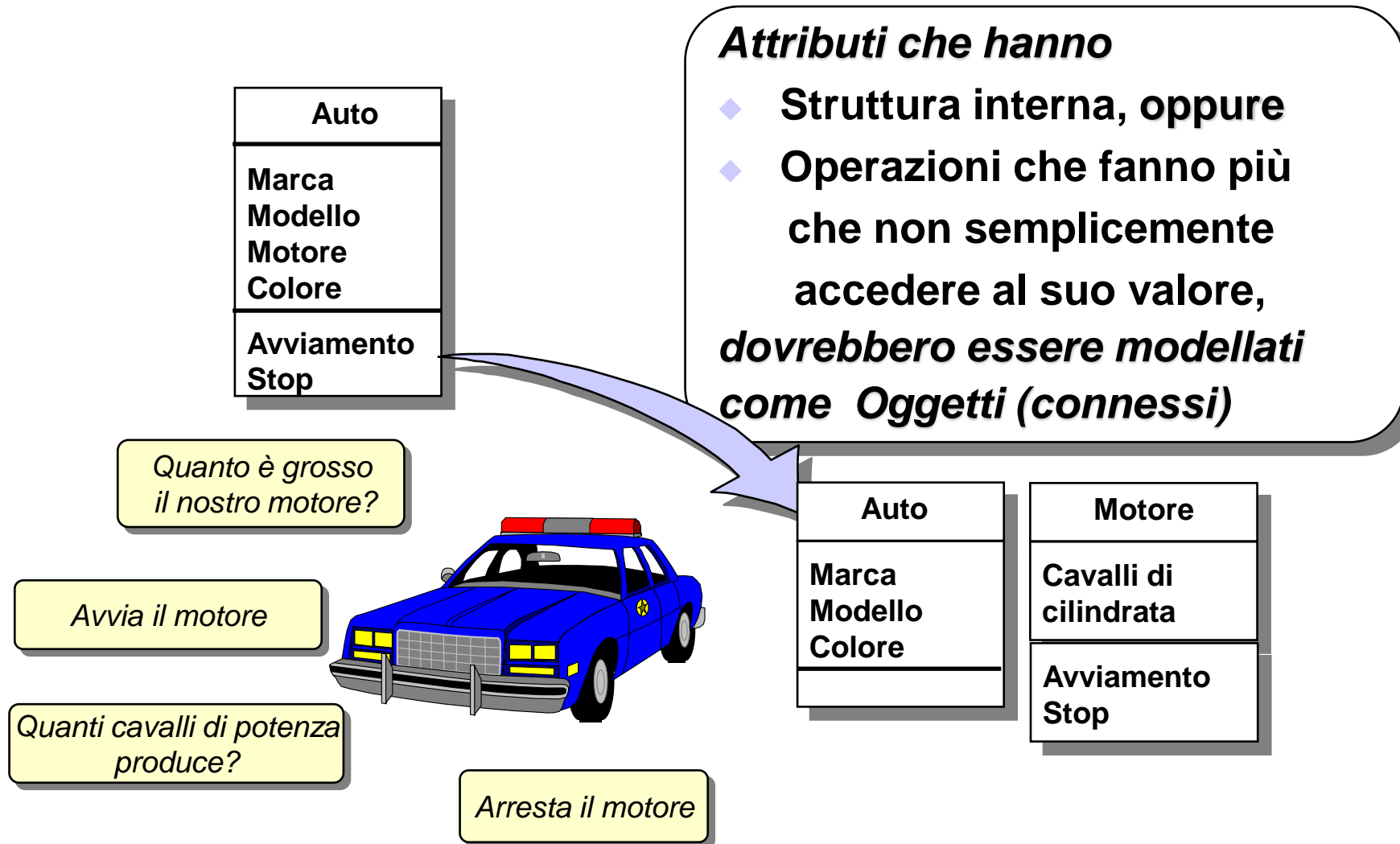
---

- I diagrammi di classe rappresentano una evoluzione dei diagrammi E-R
- Vengono inseriti concetti non direttamente rappresentabili negli E-R, quali:
  - Inclusione
  - Ereditarietà
  - Operazioni (metodi) associate alle classi

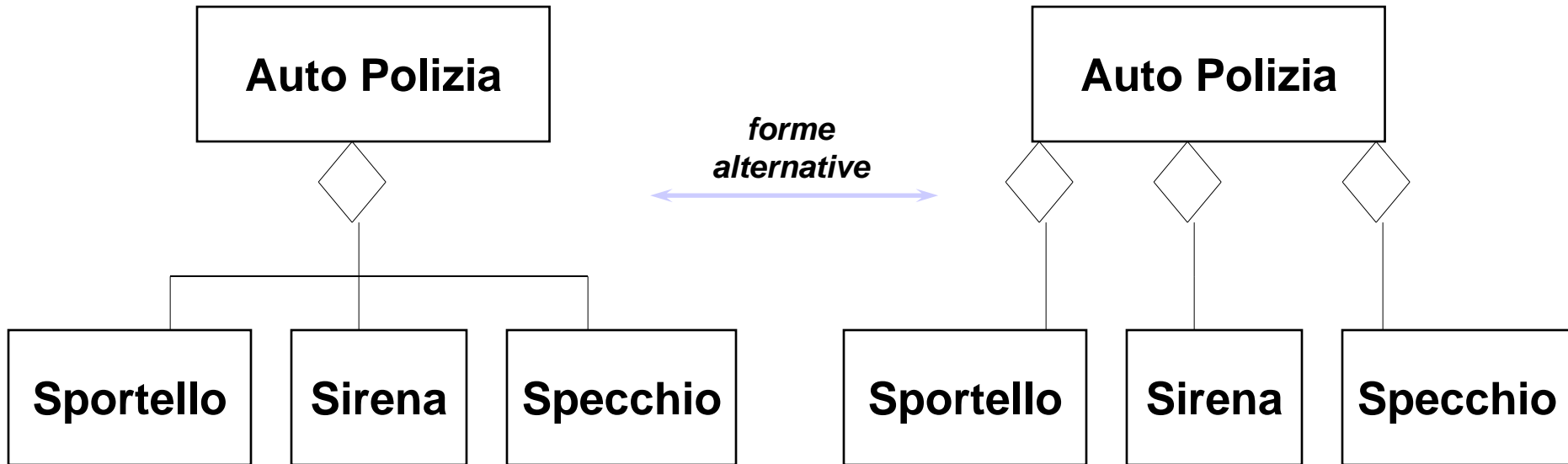
# Class diagram e diagrammi E-R: esempio



# Cosa accade quando gli Attributi diventano complessi?



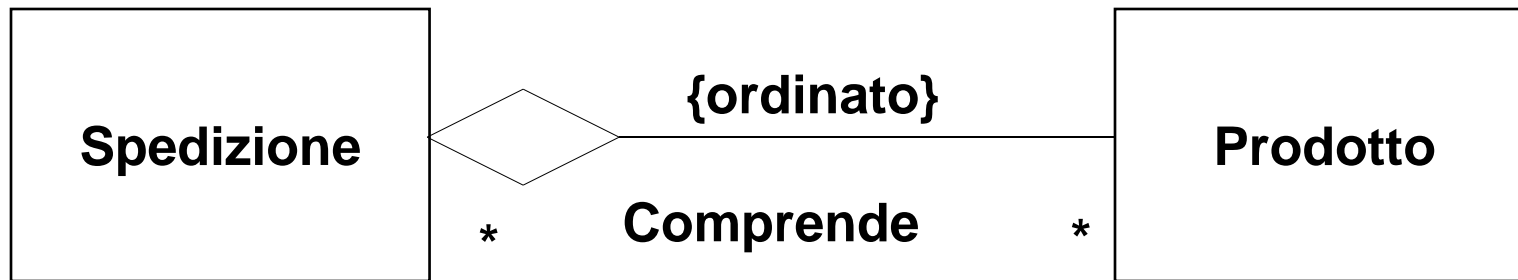
# Aggregazione di classi



- Il diamante o rombo che tocca il simbolo di una classe indica che essa contiene gli altri elementi (è un aggregato)
- La linea di relazione conduce dal diamante alle classi componenti
- Forme gerarchiche e multi-diamante hanno lo stesso significato

# Diagramma delle classi: l'aggregazione

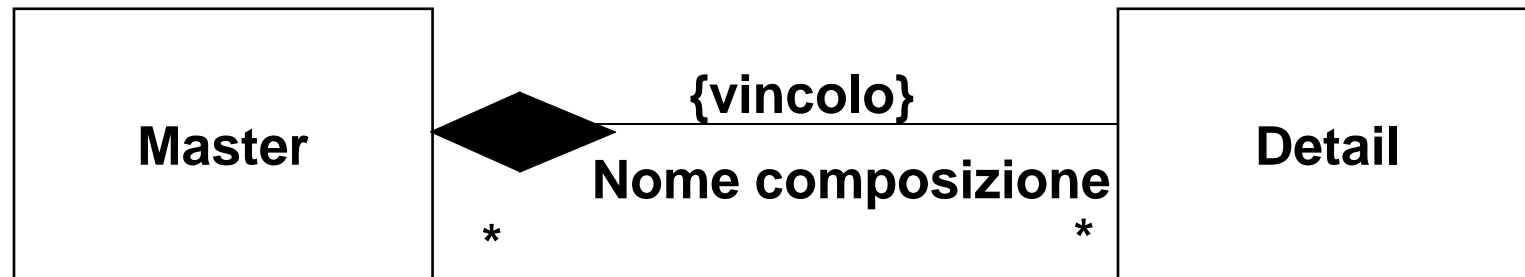
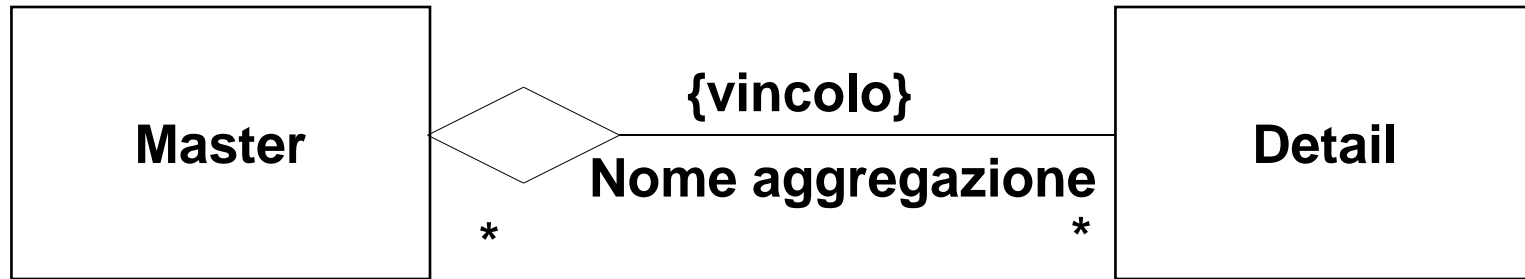
---



**Una spedizione comprende un certo numero di prodotti (quelli ordinati)**

**Un prodotto può essere compreso in più di una spedizione**

# Aggregazione e composizione

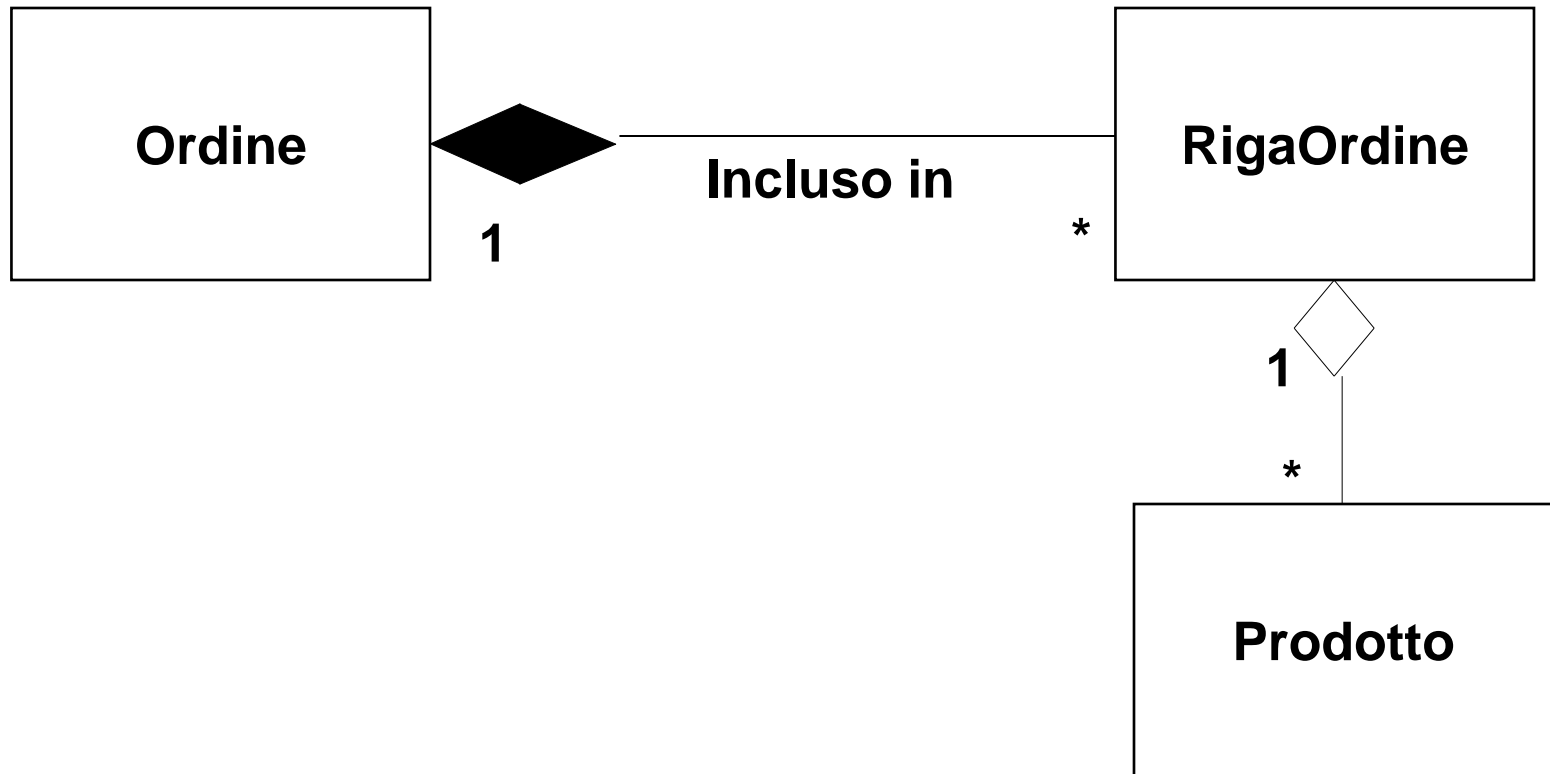


- L'eliminazione di una composizione (diamante nero) elimina anche tutti i suoi elementi componenti
- L'eliminazione di una aggregazione (diamante bianco) invece no (i componenti hanno anche una natura Indipendente)



# Aggregazione e composizione: esempio

---



**L'eliminazione di un ordine elimina anche le righe ordine ma non elimina i prodotti che esse comprendono**

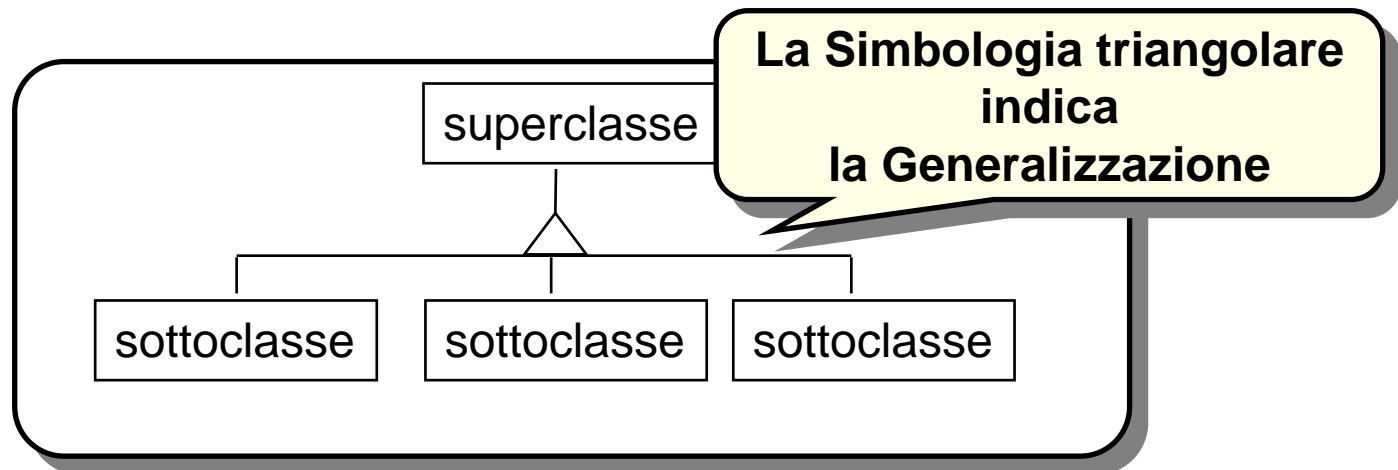
# Le Generalizzazioni

---

- Le Generalizzazioni sfruttano le Analogie delle Classi, proprio come le Classi sfruttano le Analogie degli Oggetti
- Le Generalizzazioni sfruttano gli elementi in comune delle Classi
- Le Specializzazioni definiscono le Differenze
- Le Gerarchie di Generalizzazione definiscono una “specie” di relazione

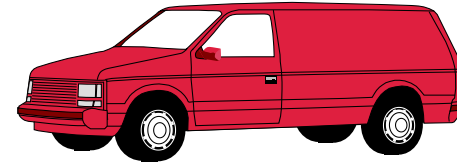
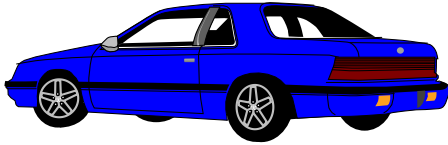
# La Generalizzazione ha una Simbologia Speciale

- Rapporti delle Generalizzazioni
  - La classe da specificare è una superclasse
  - La classe specificata è una sottoclasse
  - I genitori si riferiscono all'insieme di tutte le superclassi
  - I figli si riferiscono all'insieme di tutte le sottoclassi

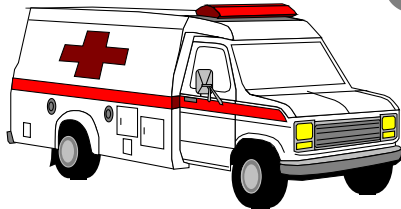


# Cosa è analogo in questi Oggetti?

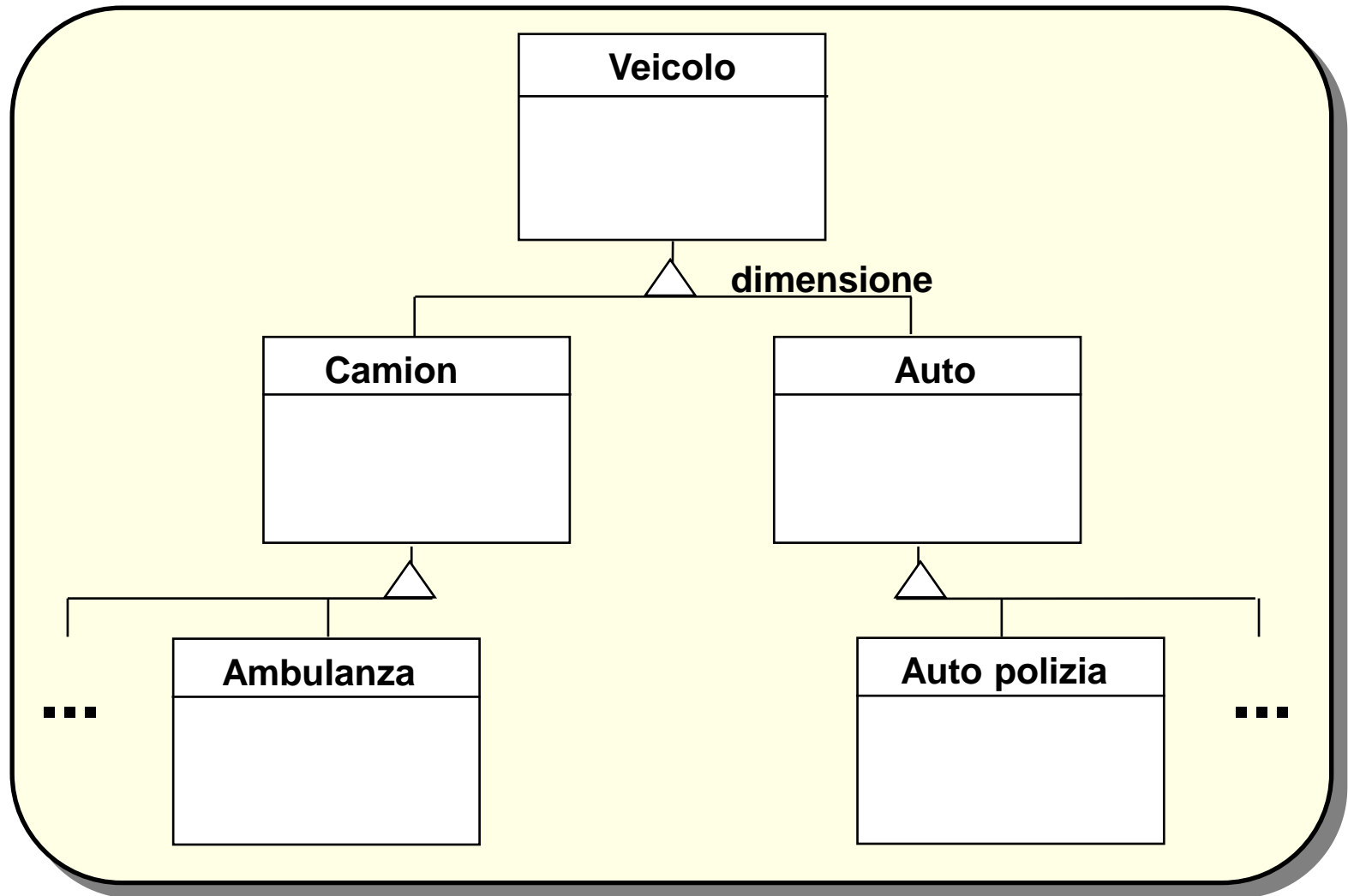
---



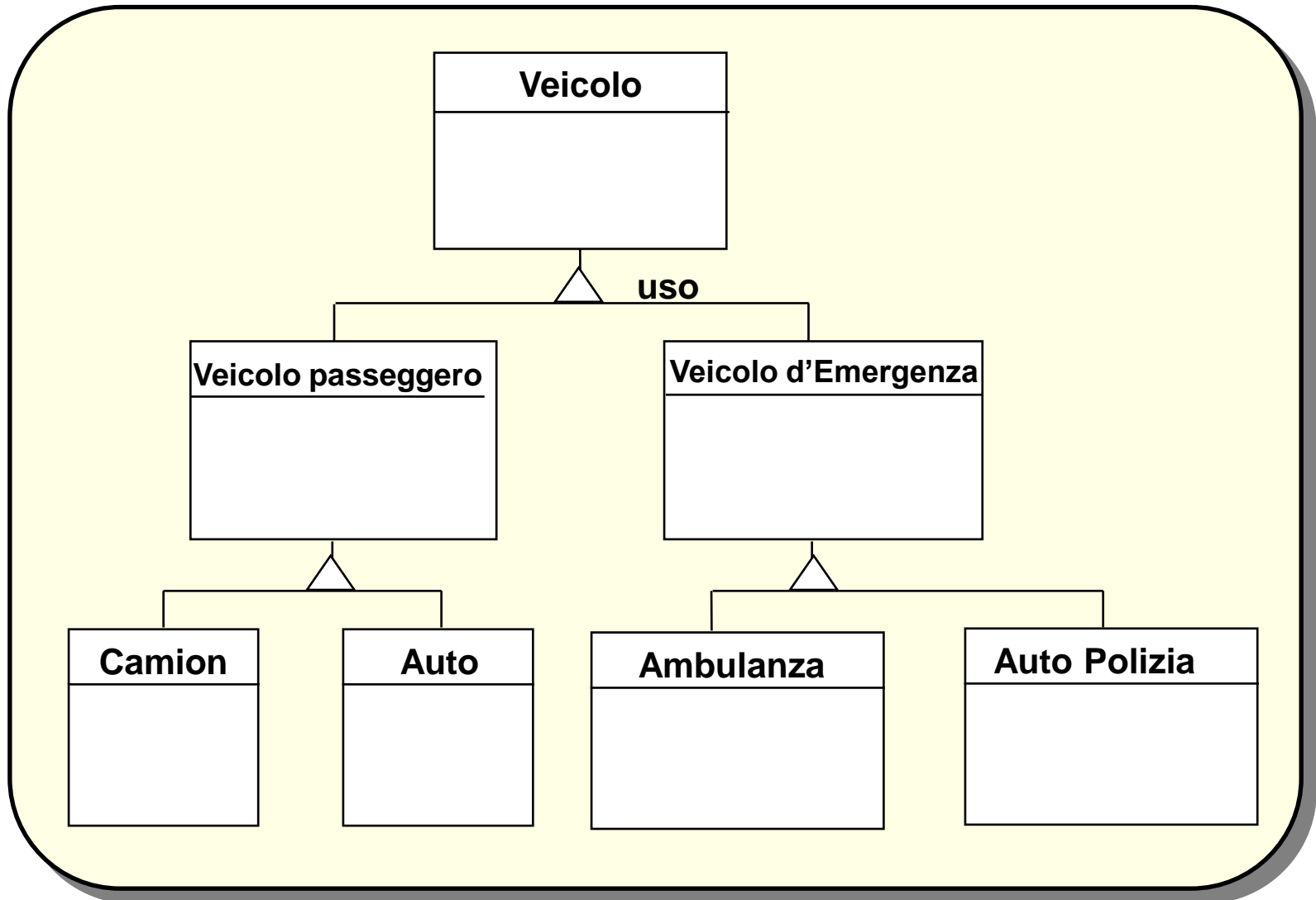
**Modella la gerarchia di  
questi veicoli**



# Una possibile gerarchia



# Un'altra gerarchia



# Il Discriminante

---

- Fornisce le Basi per la Specializzazione specificando quale proprietà della “superclasse” è da astrarre
- Valori opzionali, enumerati
- Attributo speciale che in altro modo non appare sul diagramma

# Il Discriminante

---

- Solo una proprietà alla volta dovrebbe essere distinta
- Ogni sottoclasse ha un valore unico per il discriminante
  - Si distingue dalle altre sottoclassi



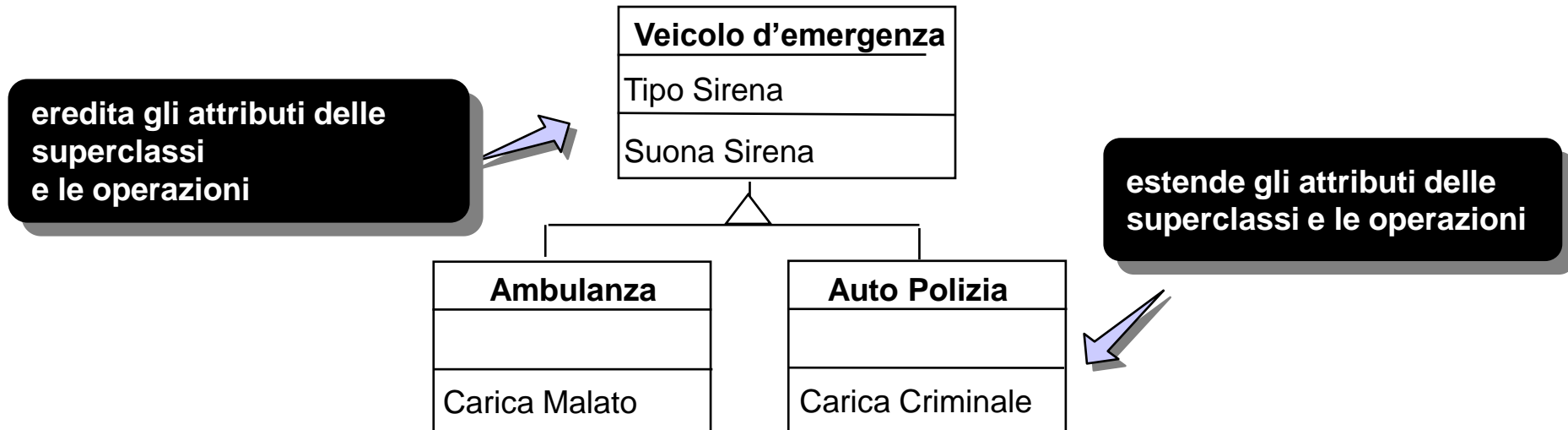
# Le Sottoclassi ereditano Proprietà dai Genitori

---

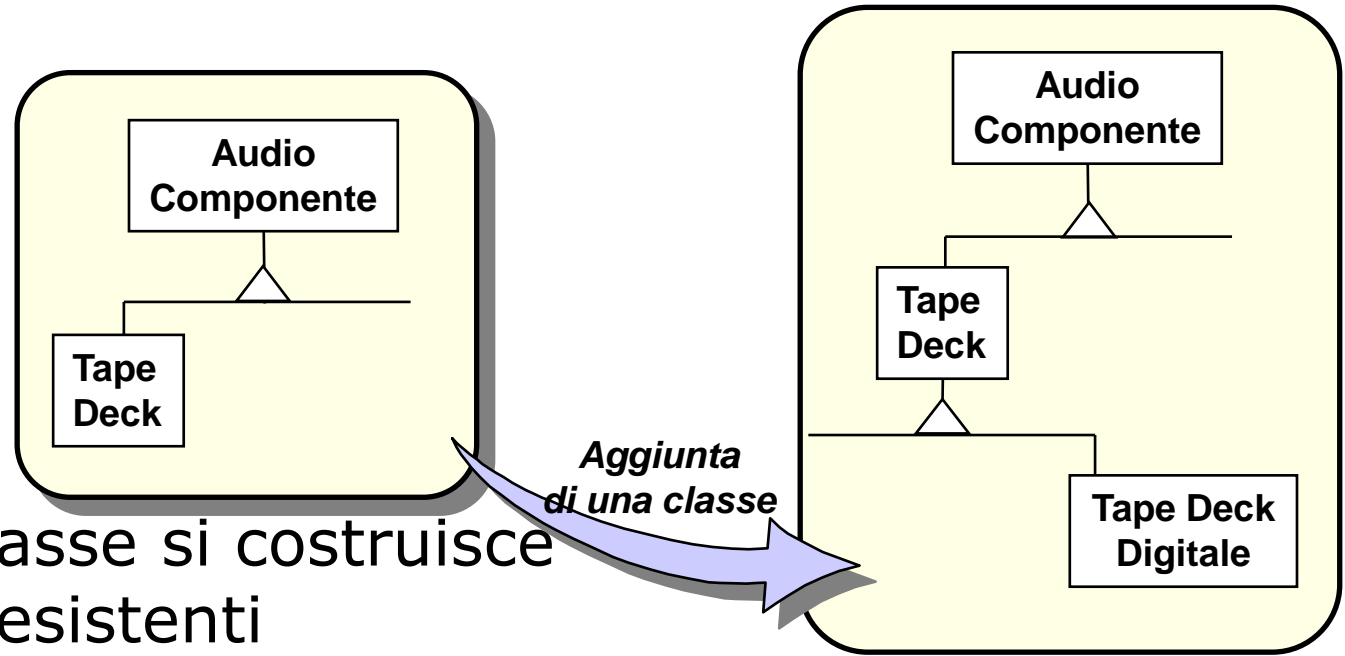
- Le Sottoclassi ereditano gli Attributi, le operazioni e le Associazioni delle loro superclassi
  - Entrambe le sottoclassi ereditano il Tipo Sirena e il Suono Sirena
- L'esempio di una sottoclasse è un esempio di tutte le classi genitrici
- Specializzazioni delle sottoclassi tramite l'aggiunta di proprietà uniche
  - L'ambulanza aggiunge il carico della vittima, il veicolo della Polizia aggiunge il carico del criminale

# Le Sottoclassi ereditano Proprietà dai Genitori - 2

---

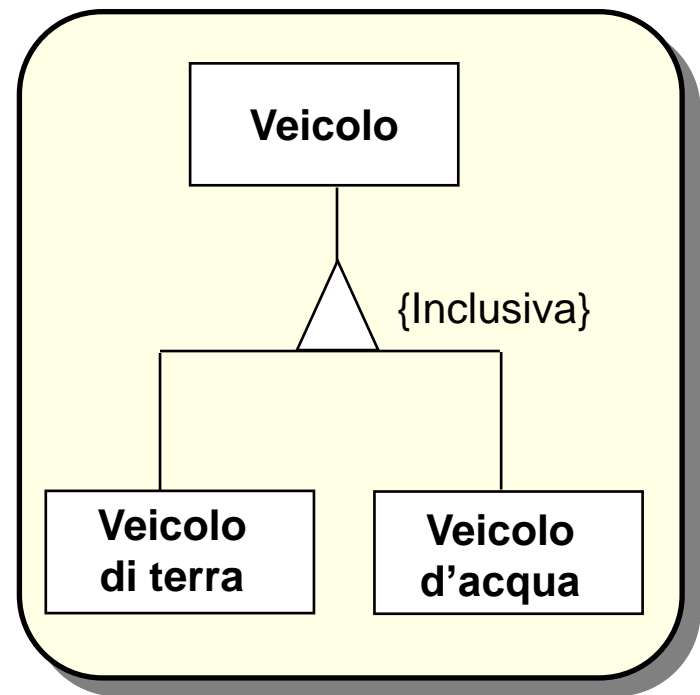
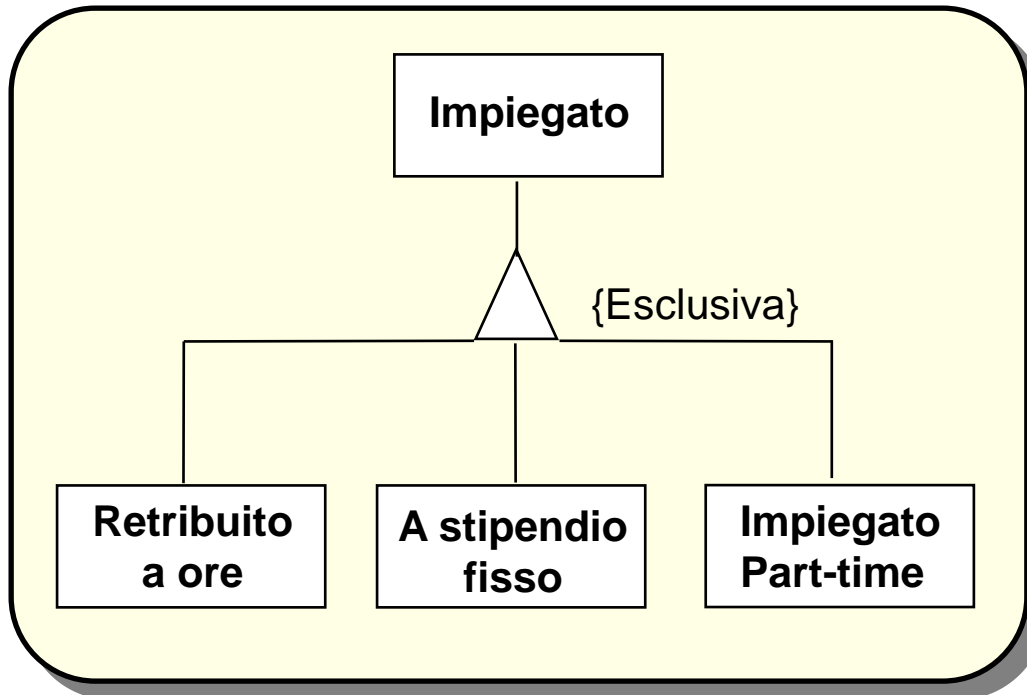


# L'ereditarietà ci consente di costruire da Componenti Simili



- La nuova classe si costruisce sulle classi esistenti ereditando le proprietà dei genitori nella gerarchia di generalizzazione
- La nuova classe ha bisogno solo d'implementare le estensioni e le differenze

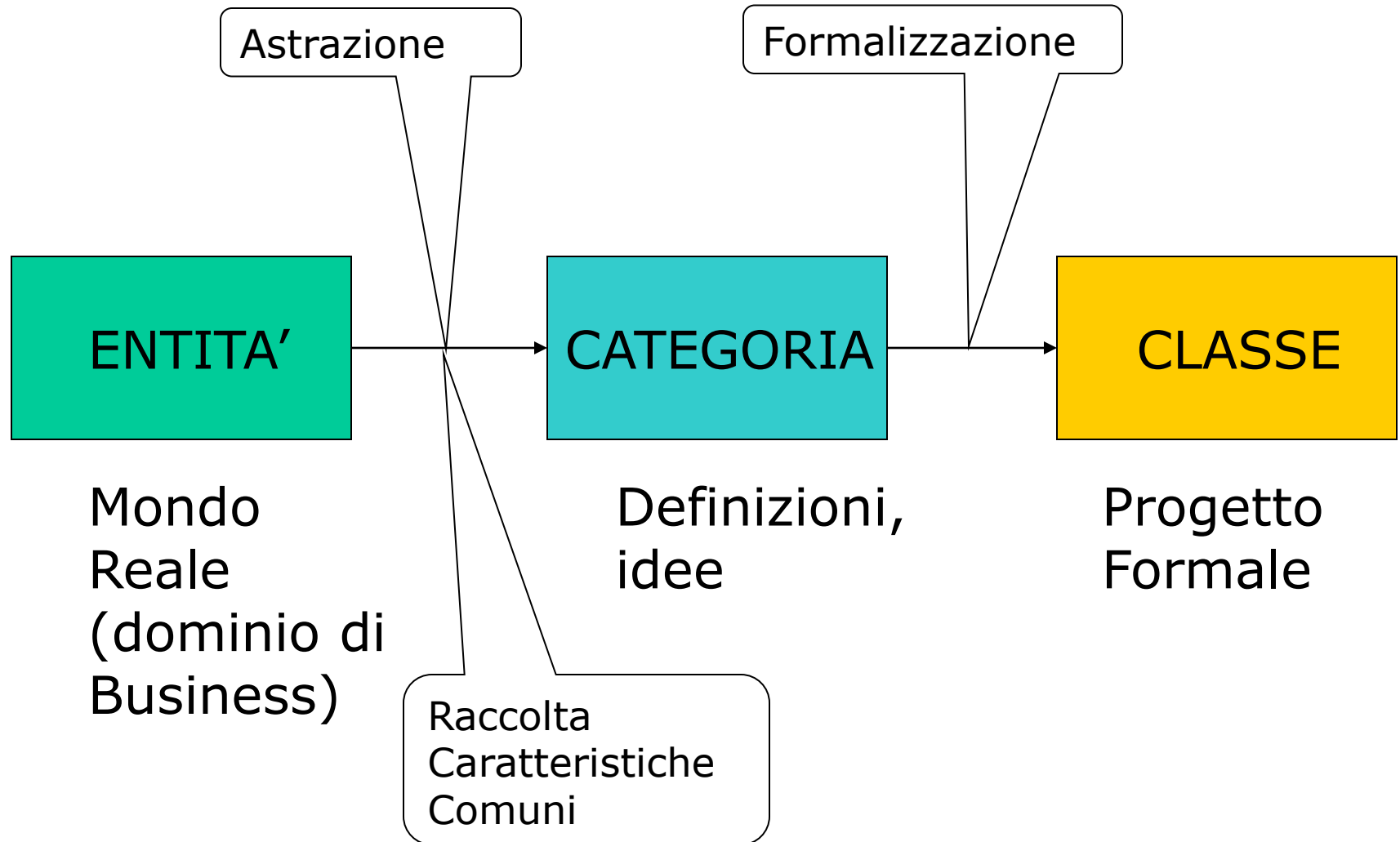
# La suddivisione in Sottoclassi può essere Esclusiva o Inclusiva



- L'impiegato è retribuito a ore, a stipendio fisso o part time
- Un veicolo è un veicolo di terra, un veicolo d'acqua o entrambi.

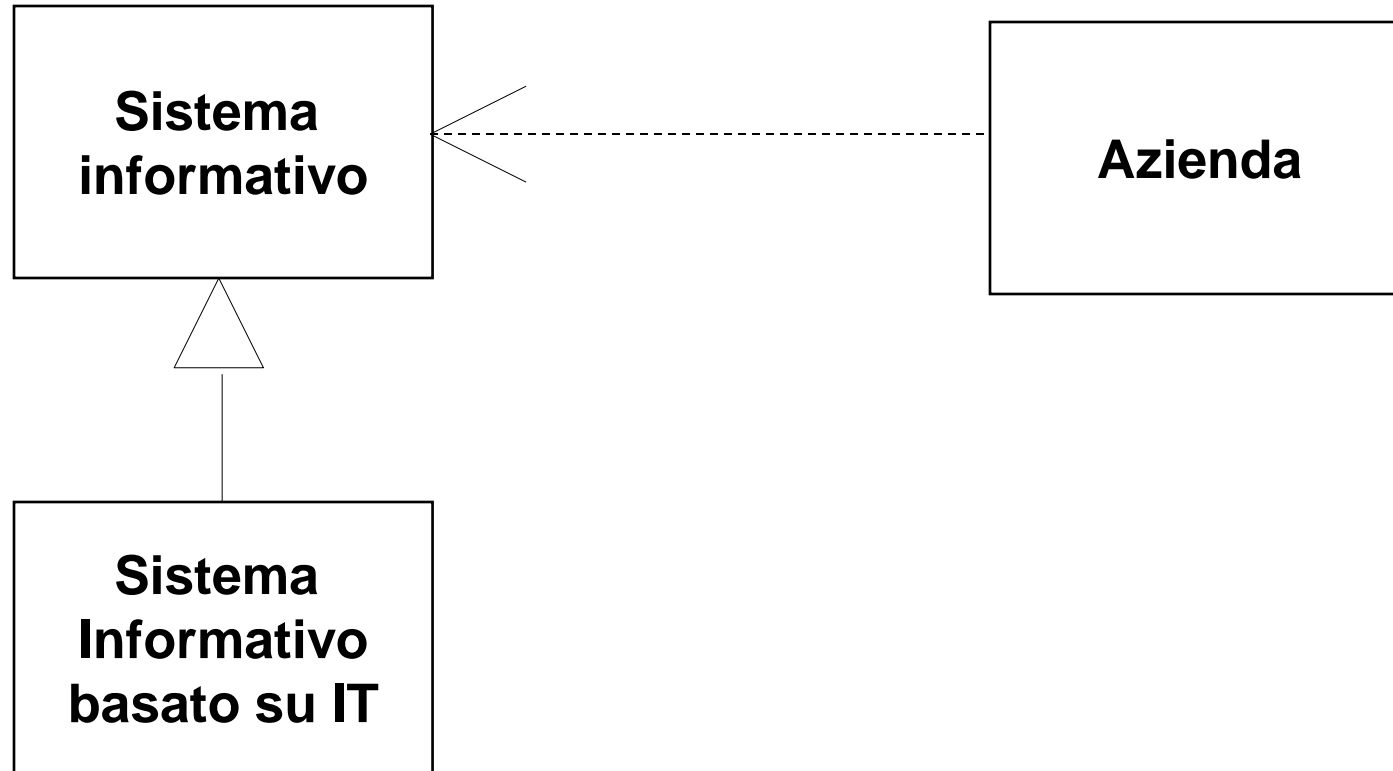
# Il passaggio da entità a classi

---



# Diagramma delle classi: dipendenza

---



**L'Azienda dipende dal sistema informativo, che viene Realizzato attraverso un sistema informativo basato sull'information technology**

# Classi e risorse

---

**<<Abstract>>  
Azione**

**<<Physical>>  
Trapano**

**<<People>>  
Venditore**

**<<Information>>  
Business news**

**Gli stereotipi definiscono concetti che personalizzano le entità modellizzate con le classi**

# Classi e risorse: metamodelli

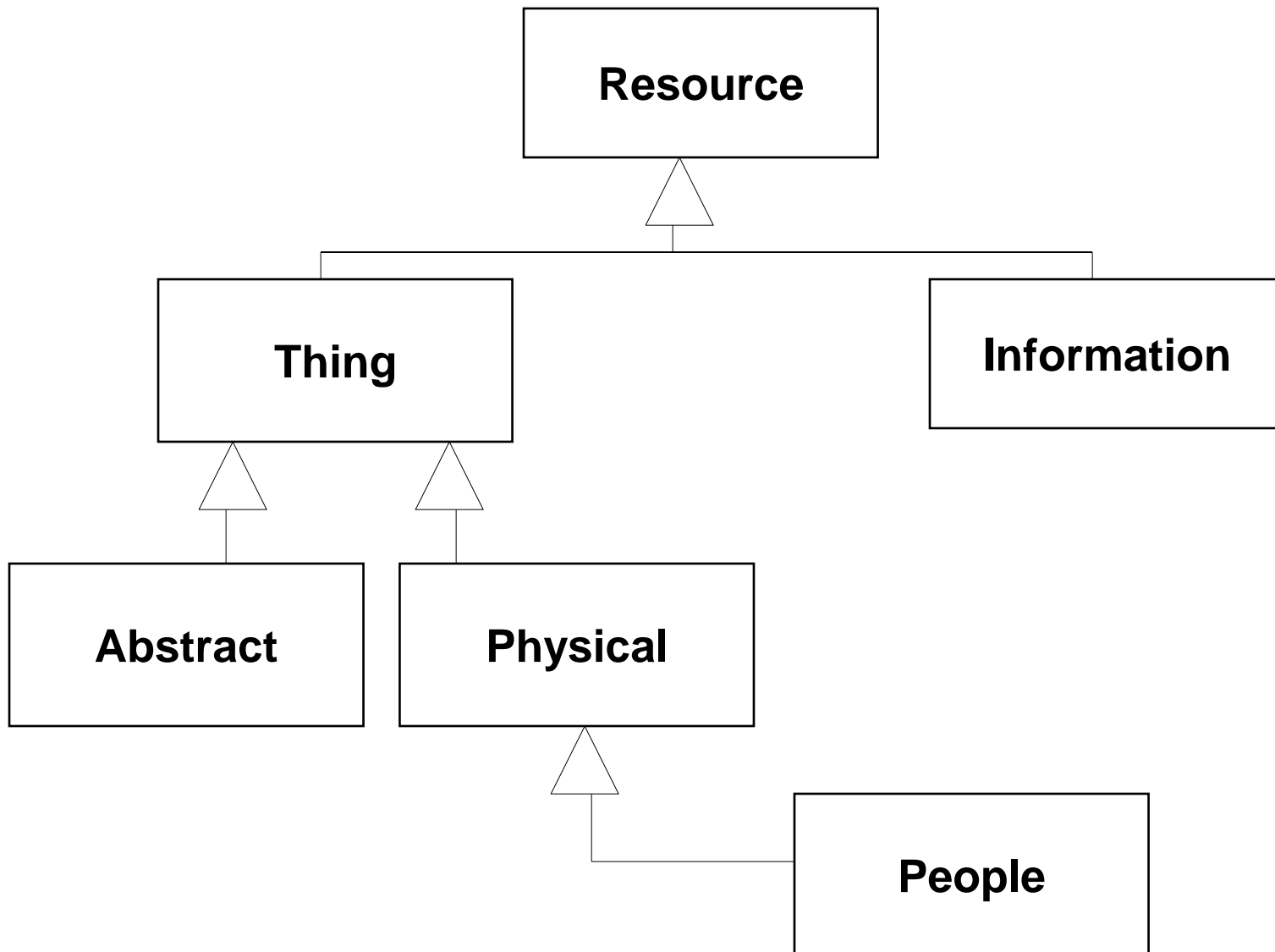
---

- Attraverso i class diagram è possibile esprimere anche concetti costitutivi di altre classi e/o categorie
- Si possono costruire quindi dei metamodelli

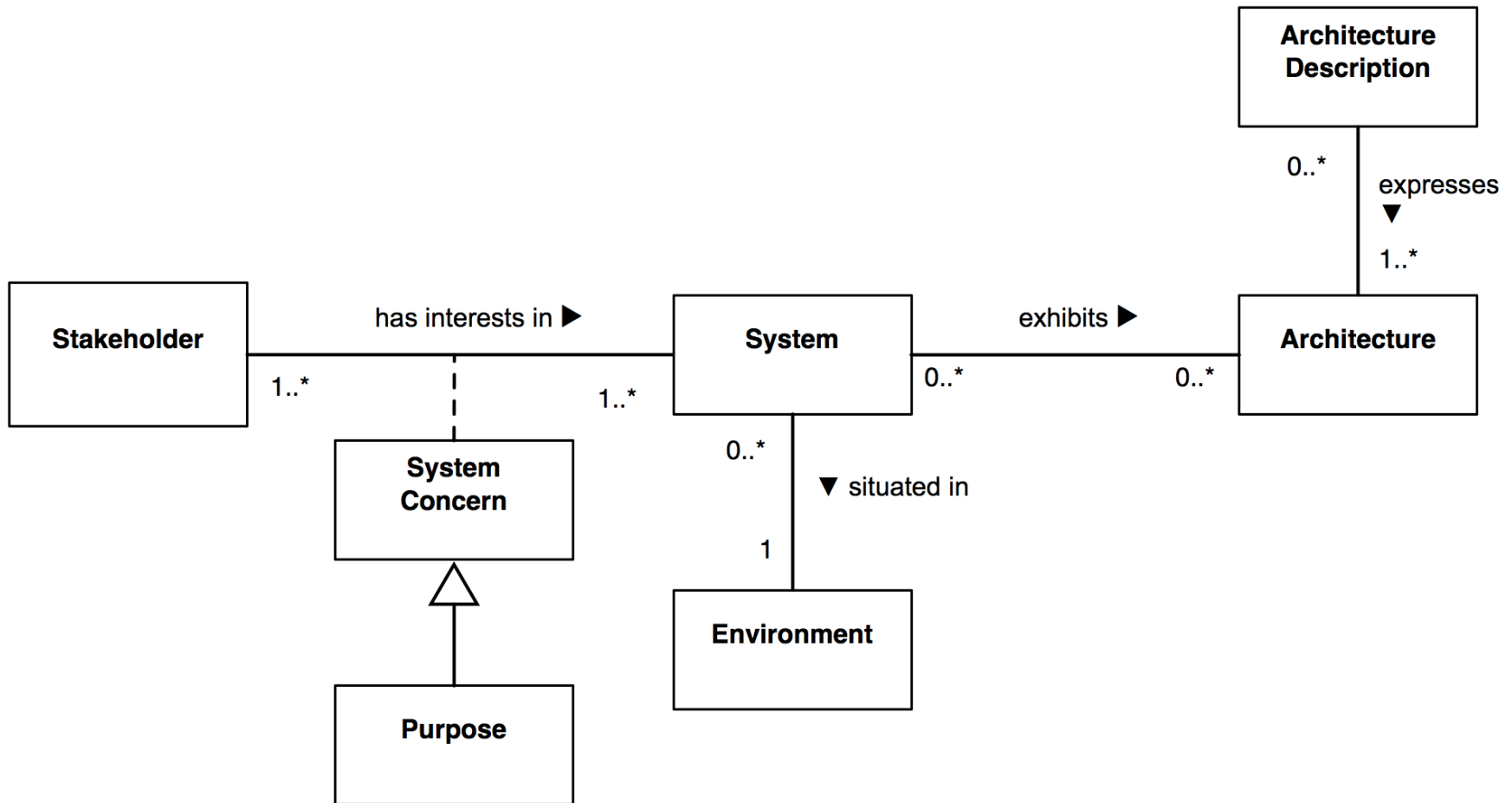


# Metamodelli: esempio

---



# Business Architecture Class Diagram



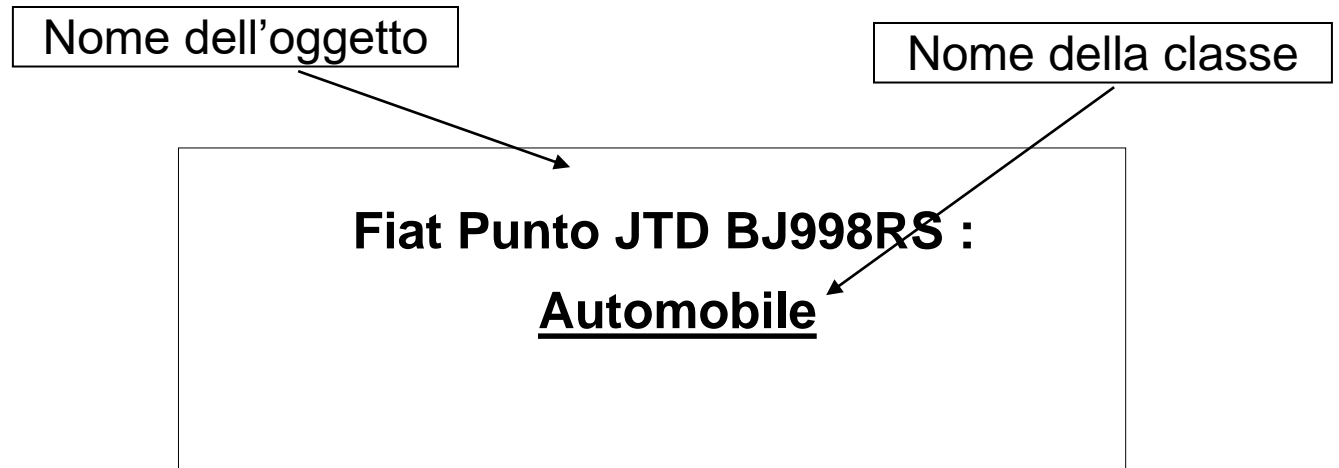
# Class diagram ed object diagram

---

- Gli object diagram sono simili ai class diagram, ma i loro componenti sono gli oggetti, ossia istanze ben definite delle classi
- In pratica quindi un object diagram rappresenta un insieme di legami logici caratteristici di di entità concrete e non astratte

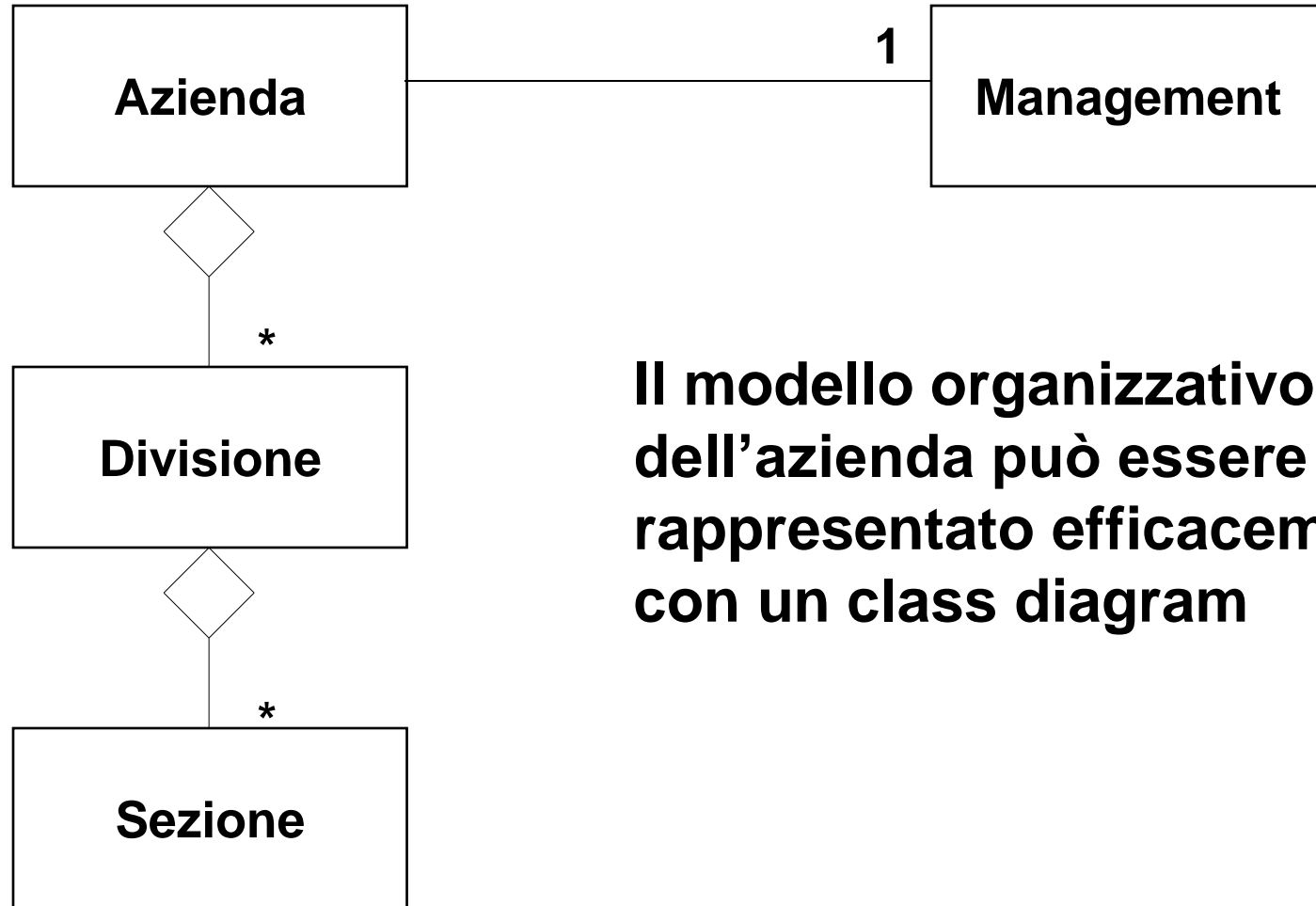
# Object diagram: sintassi

---



# Classi ed oggetti: organizzazione

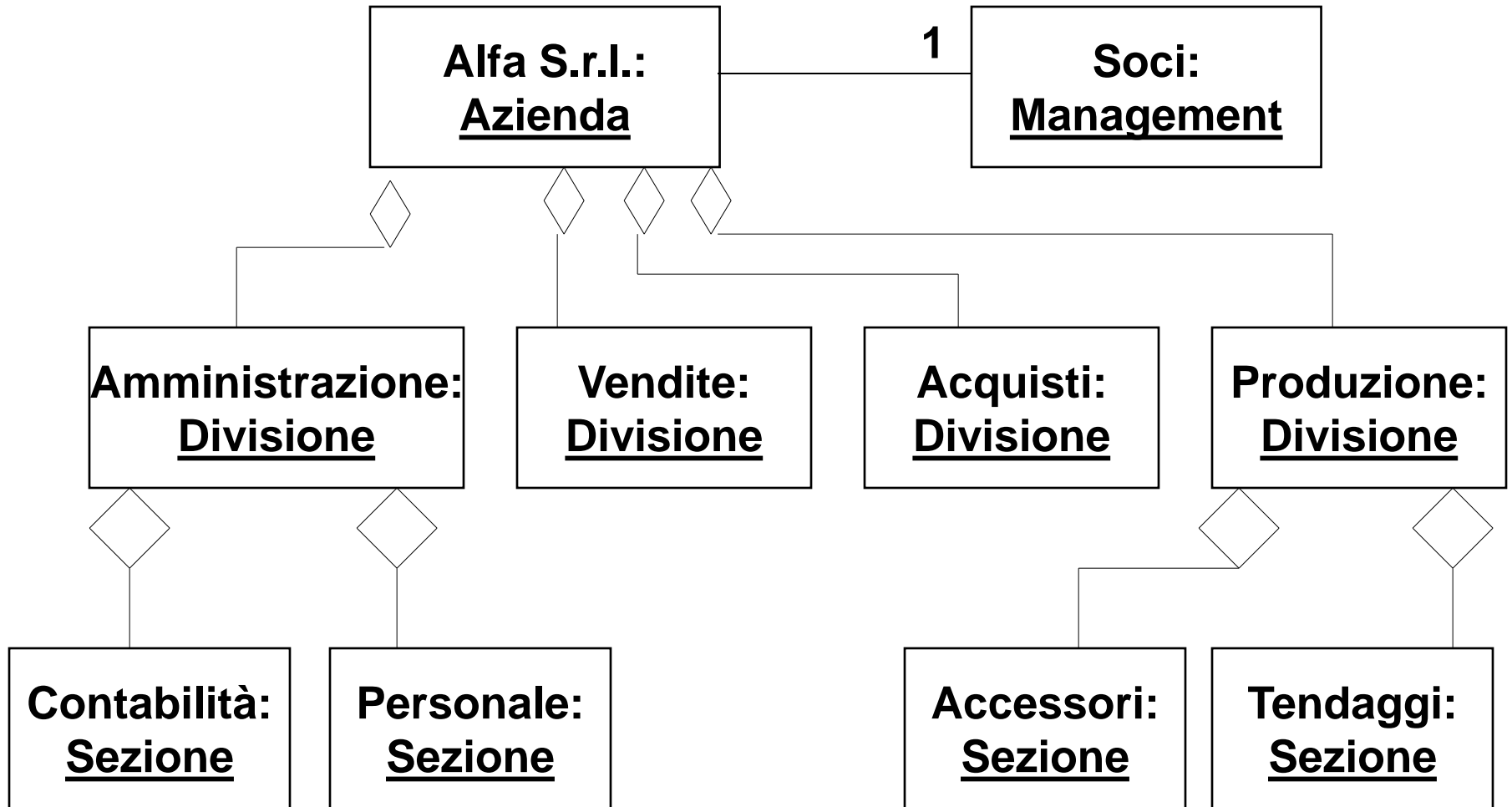
---



**Il modello organizzativo dell'azienda può essere rappresentato efficacemente con un class diagram**

# Organizzazione effettiva: object diagram

---



- **Activity Diagram**
- **Use Case Diagram**
- **Class Diagram**
- **Sequence Diagram**
- **Collaboration Diagram**
- **Statechart Diagram**

# Diagrammi di sequenza

---

- Descrivono le interazioni fra gli oggetti organizzate in sequenza temporale
- Uno use case contiene al suo interno vari diagrammi di sequenza
- Salvo casi banali, non si rappresentano all'inizio tutte le possibili sequenze, ma solo le principali



# Diagrammi di sequenza: gli elementi

---

- Elementi costitutivi di un diagramma di sequenza:
  - gli oggetti
  - i messaggi attraverso cui essi interagiscono.
- Lo scambio di messaggi è rappresentato da frecce con un nome.

# Diagrammi di sequenza: i messaggi

---

In base al livello di dettaglio

- I messaggi possono esplicitamente far riferimento ai metodi (operazioni) effettivamente coinvolti, ossia richiamati;
- Possono essere specificati anche i parametri e/o i risultati;
- Oppure viene descritta un'azione generica.

# Diagrammi di sequenza: il controllo

---

- Ripetizioni cicliche (for, while)
- Condizioni (if, case)

# Diagrammi di sequenza: messaggi e azioni

---

- **Call**: invoca un metodo di un oggetto; un oggetto può inviare un messaggio a se stesso, invocando un proprio metodo
- **Return**: restituisce un valore al chiamante
- **Send**: invia un signal ad un oggetto
- **Create**: crea un oggetto
- **Destroy**: distrugge un oggetto; un oggetto può distruggere se stesso

# Diagrammi di sequenza: sequenze di messaggi

---

- Uno scambio di messaggi può dare origine ad una sequenza
- La sequenza deve avere un punto d'inizio ("evento scatenante")
- Può essere utile anteporre numeri ai nomi dei messaggi

# Sequence Diagram: simboli base

---



Oggetto1

Oggetto di riferimento con nome “oggetto1”

Asse dei tempi diretto verso il basso

Richiesta fatta



Messaggio in partenza (comando inviato)

Richiesta ricevuta/  
risposta



Messaggio (comando) ricevuto



Durata temporale (opzionale) di un'azione, ovvero periodo durante il quale l'oggetto controlla il flusso

**(Focus of control)**

Nome del diagramma di sequenza (di solito in fondo)

# Sequence Diagram: tipi di entità

---

**Oggetto1:**

Oggetto di riferimento con nome “oggetto1”

**Oggetto1:  
Classe1**

Oggetto di riferimento con nome “oggetto1”,  
istanza della classe “classe1”

**[:]Classe1**

Oggetto di riferimento,  
istanza generica della classe “classe1”  
(tutti gli oggetti di quella classe)

# Sequence Diagram: tipi di messaggi

---

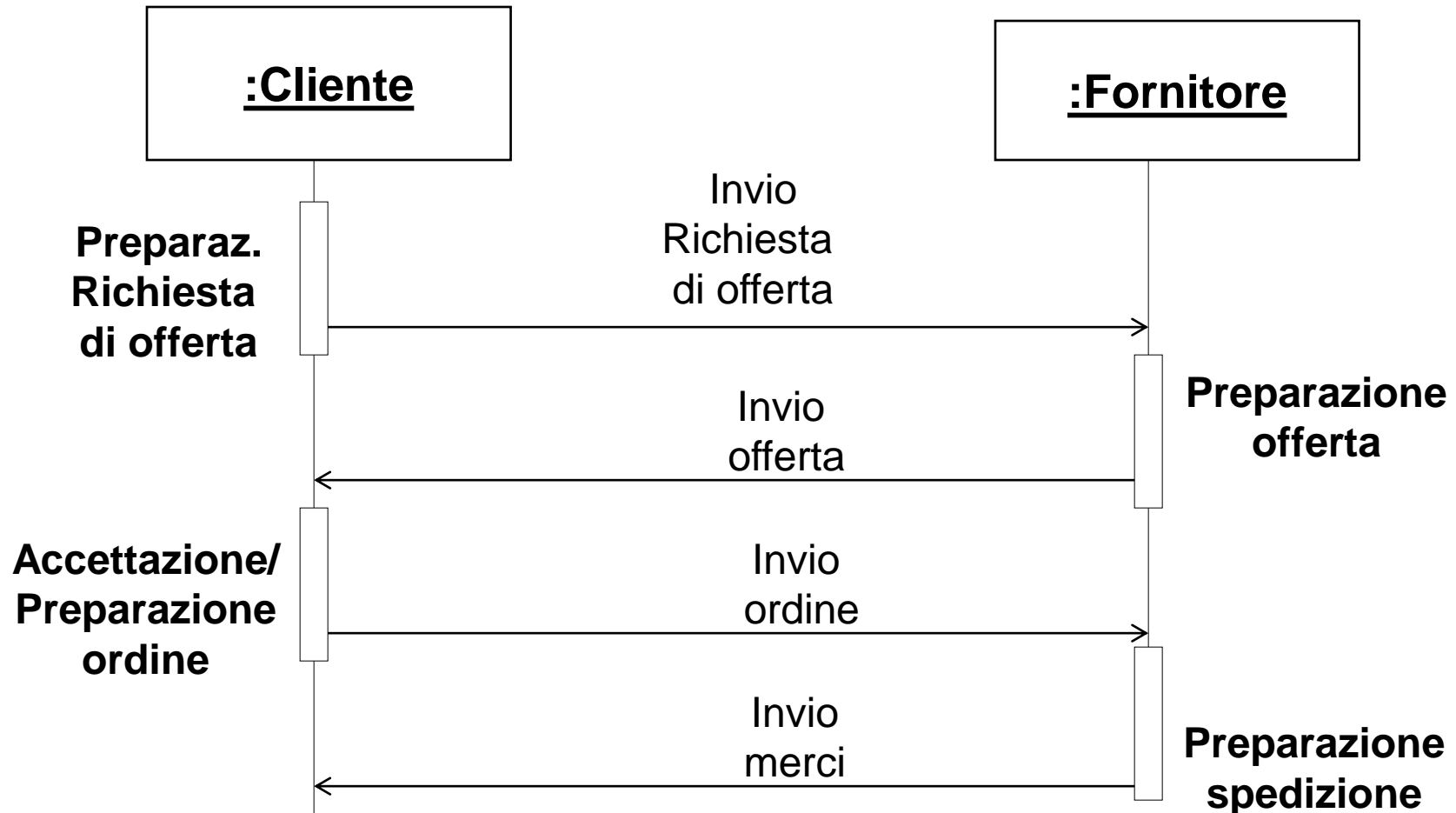
————→ **Semplice:** il controllo è passato dal chiamante al ricevente

————→ **Sincrono:** il controllo è passato dal chiamante al ricevente ed il primo attende che il secondo gli restituisca il controllo

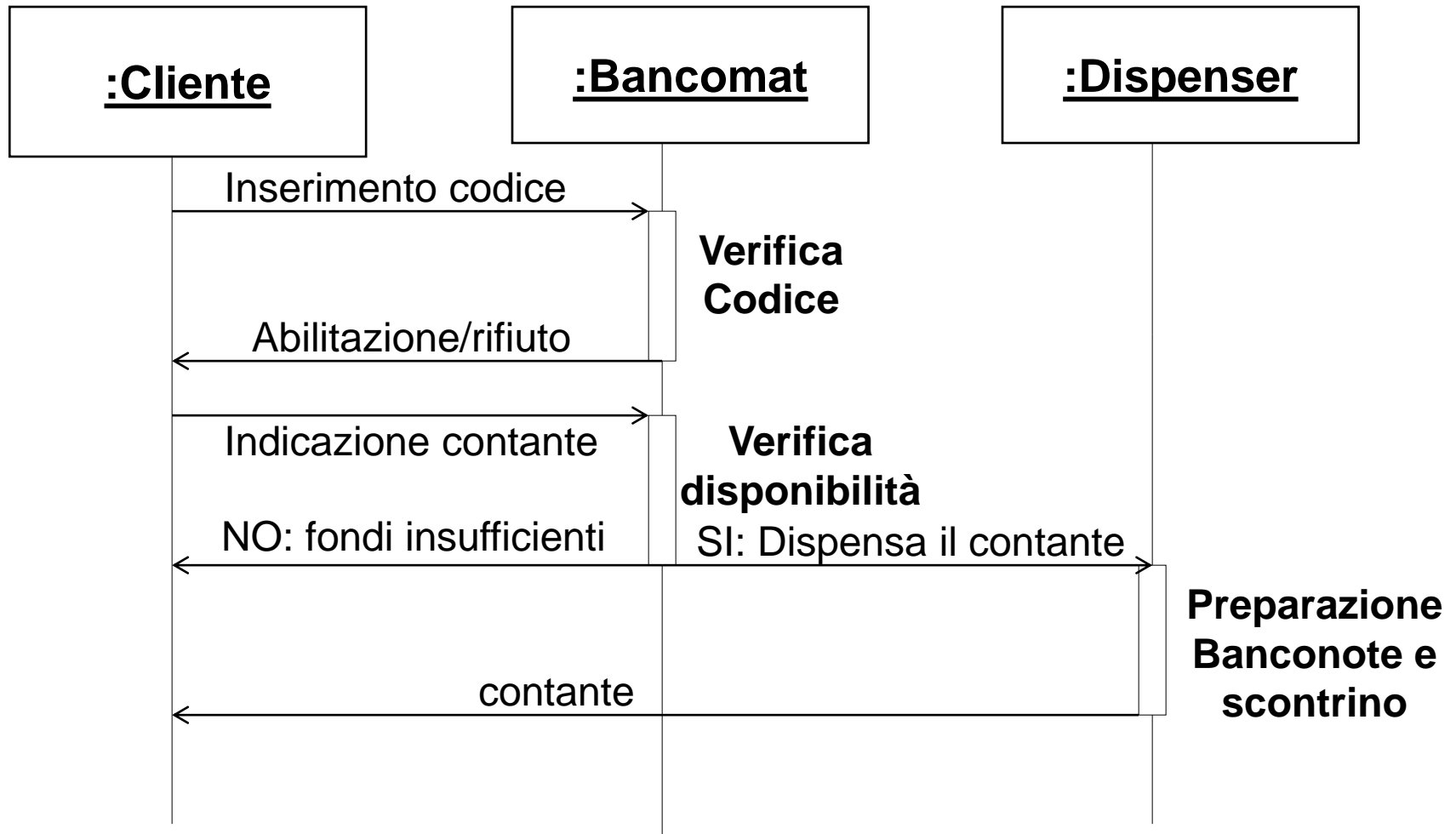
————> **Asincrono:** il chiamante trasmette un segnale al ricevente ma prosegue poi nelle proprie azioni senza attendere il secondo che può o meno ritornare informazioni



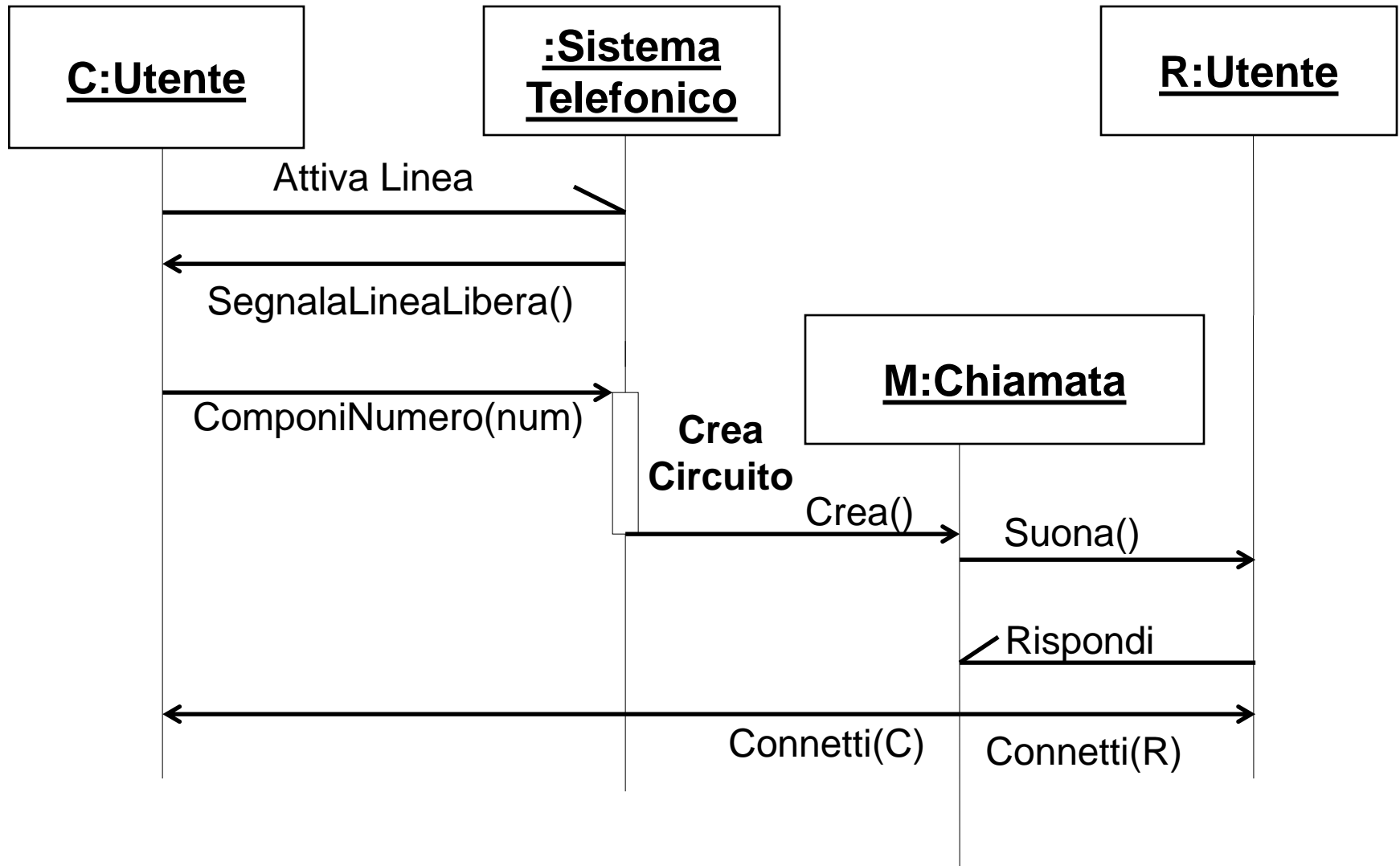
# Esempio di diagramma di sequenza: offerta



# Esempio di diagramma di sequenza: bancomat



# Esempio di diagramma di sequenza: telefono





- 
- **Activity Diagram**
  - **Use Case Diagram**
  - **Class Diagram**
  - **Sequence Diagram**
  - **Collaboration Diagram**
  - **Statechart Diagram**

# Diagrammi di collaborazione

---

- Sono semanticamente equivalenti ai diagrammi di sequenza: entrambi sono visualizzazioni di scenari
- I diagrammi di collaborazione enfatizzano le relazioni fra oggetti (ovvero l'organizzazione strutturale), i diagrammi di sequenza enfatizzano la sequenza temporale delle comunicazioni

# Diagrammi di collaborazione - 2

---

- La sequenza dei messaggi è meno evidente che nel diagramma di sequenza, mentre sono più evidenti i legami tra gli oggetti
- I messaggi hanno sempre espresso l'ordine di sequenza

# Diagrammi di collaborazione - 3

---

- I diagrammi di collaborazione vengono usati prevalentemente in fase di progetto, quelli di sequenza in fase di analisi, perché sono più comprensibili da parte del committente (cliente, esperto del dominio)
- I due diagrammi sono isomorfi, è possibile cioè trasformare uno nell'altro

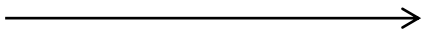
# Collaboration Diagram: simboli base

---

**Oggetto1**

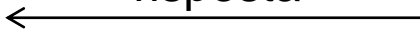
Oggetto di riferimento con nome “oggetto1”

1: Richiesta fatta



Messaggio in partenza (comando inviato),  
con numero di sequenza

2: Richiesta ricevuta/  
risposta



Messaggio (comando) ricevuto,  
con numero di sequenza

Nome del diagramma di collaborazione (di solito in fondo)



# Collaboration Diagram: tipi di entità

---

**Oggetto1:**

Oggetto di riferimento con nome “oggetto1”

**Oggetto1:**  
**Classe1**

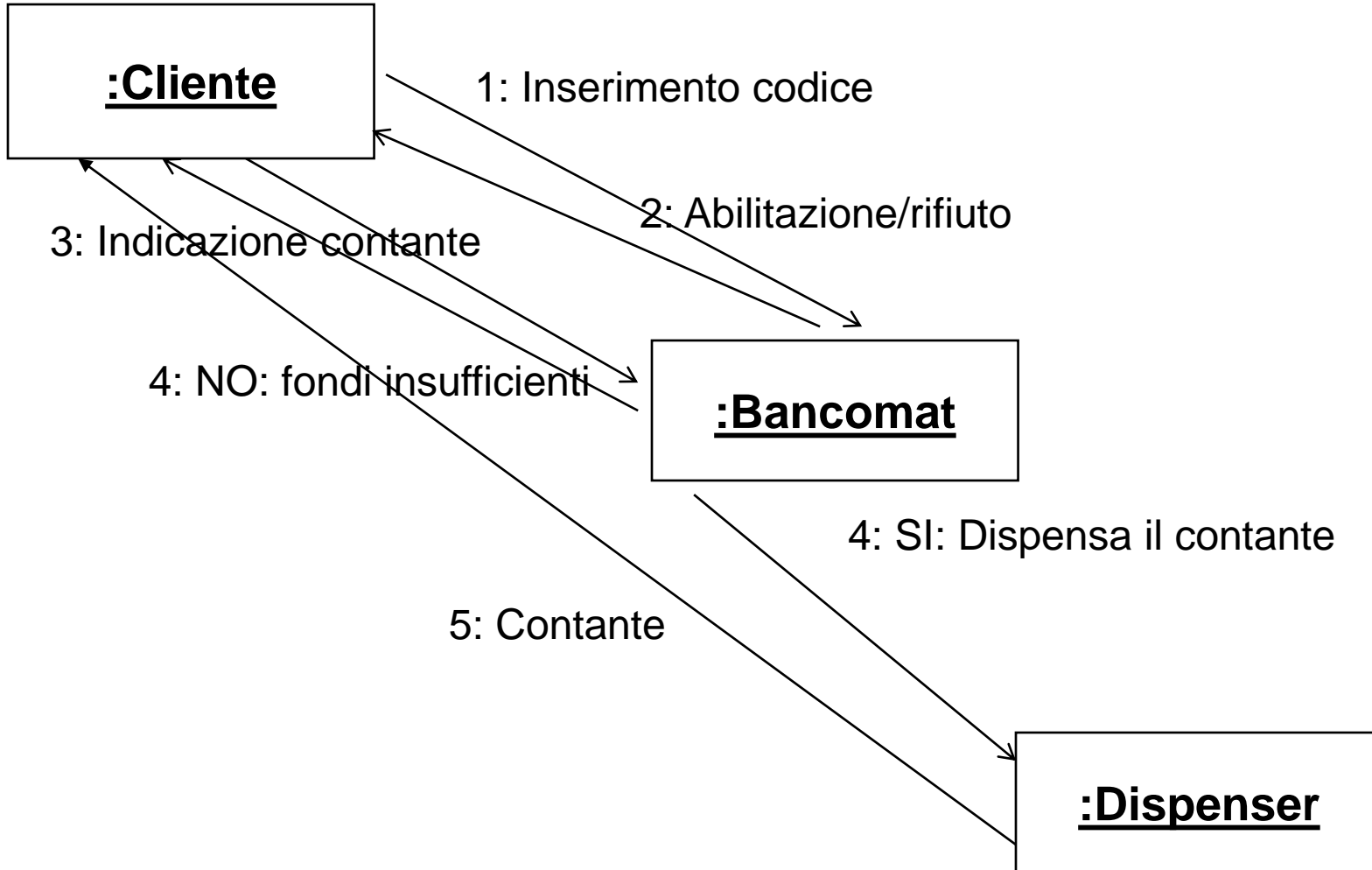
Oggetto di riferimento con nome “oggetto1”,  
istanza della classe “classe1”

**[:]Classe1**

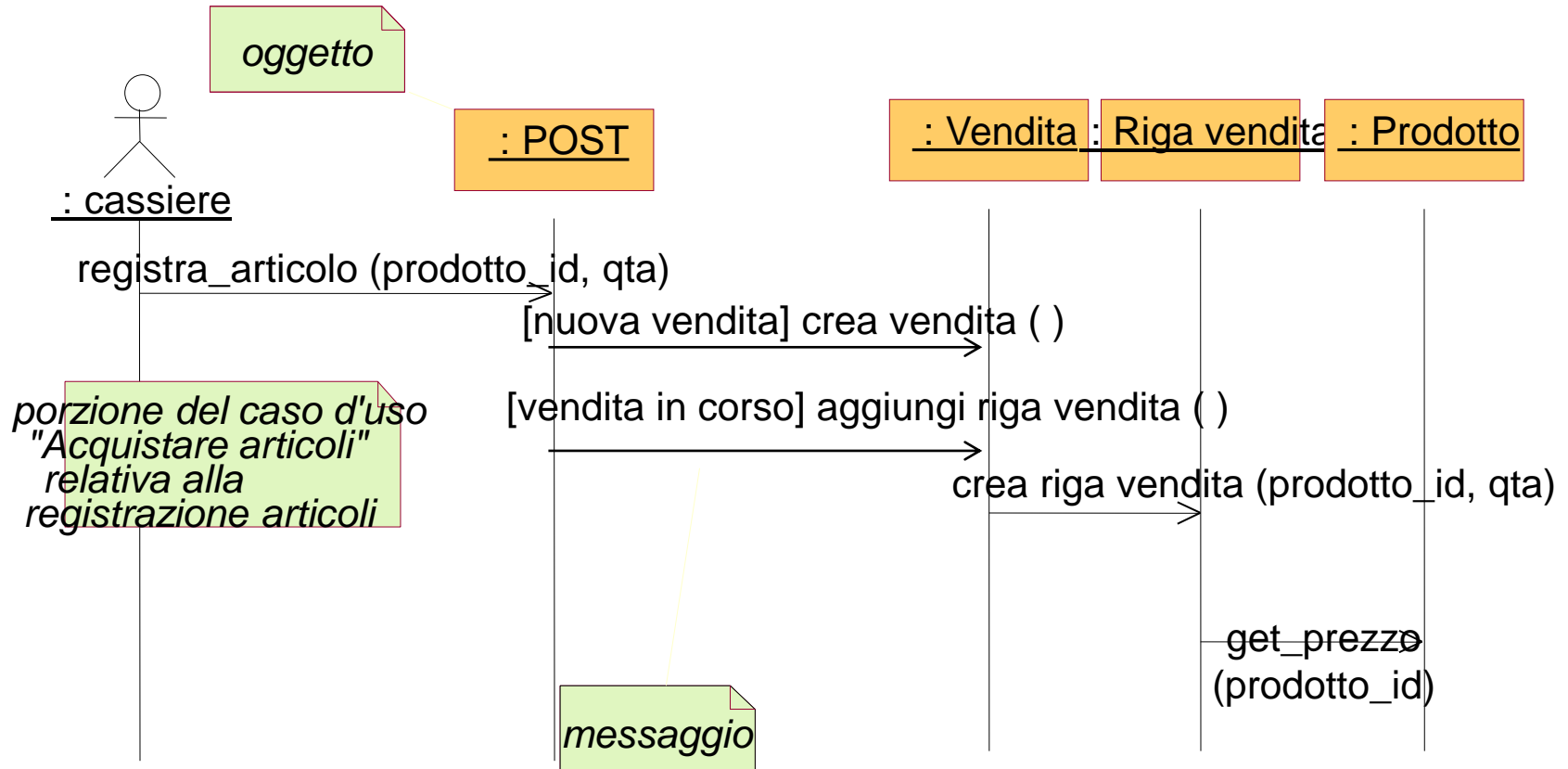
Oggetto di riferimento,  
istanza generica della classe “classe1”  
(tutti gli oggetti di quella classe)

# Esempio di diagramma di collaborazione: bancomat

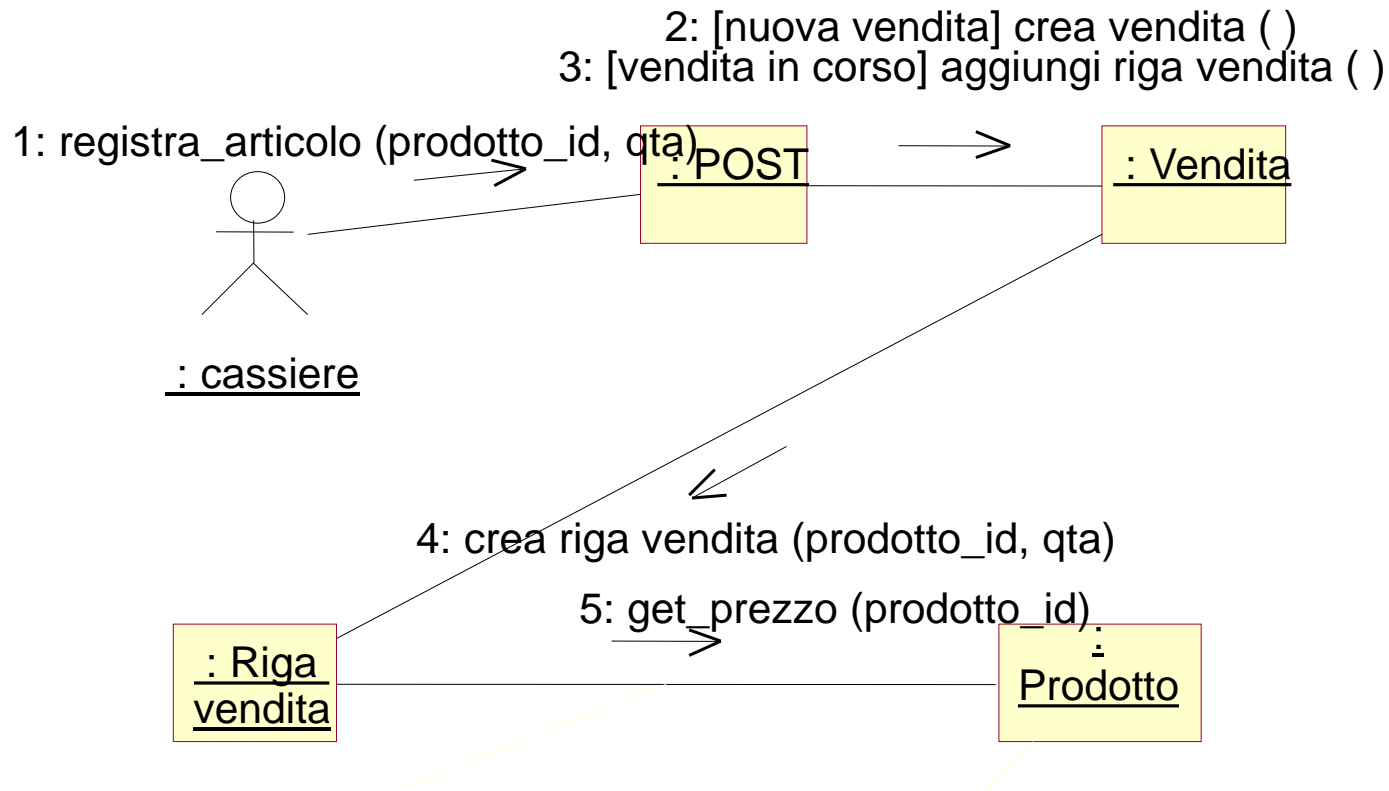
---



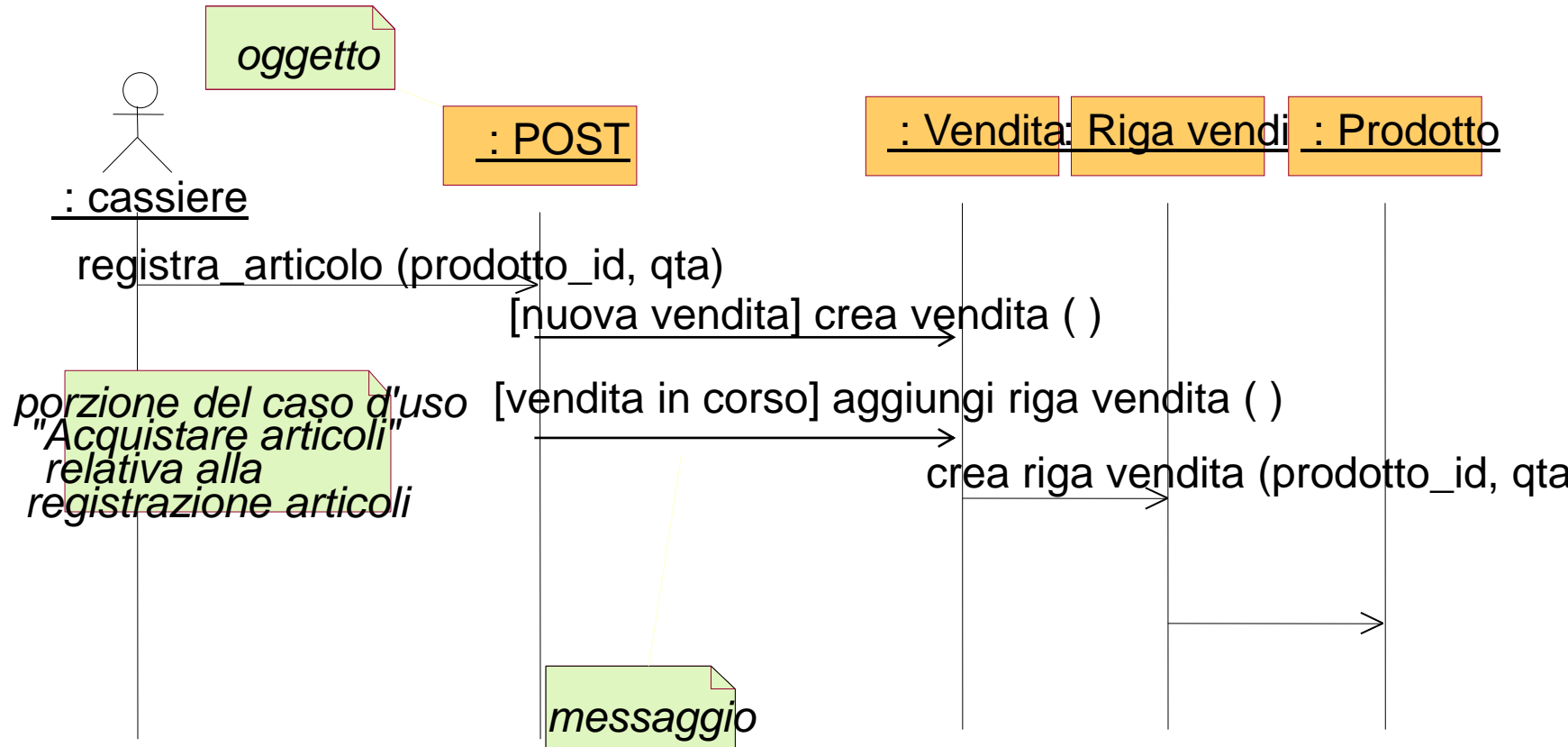
# Esempio completo: sequenza



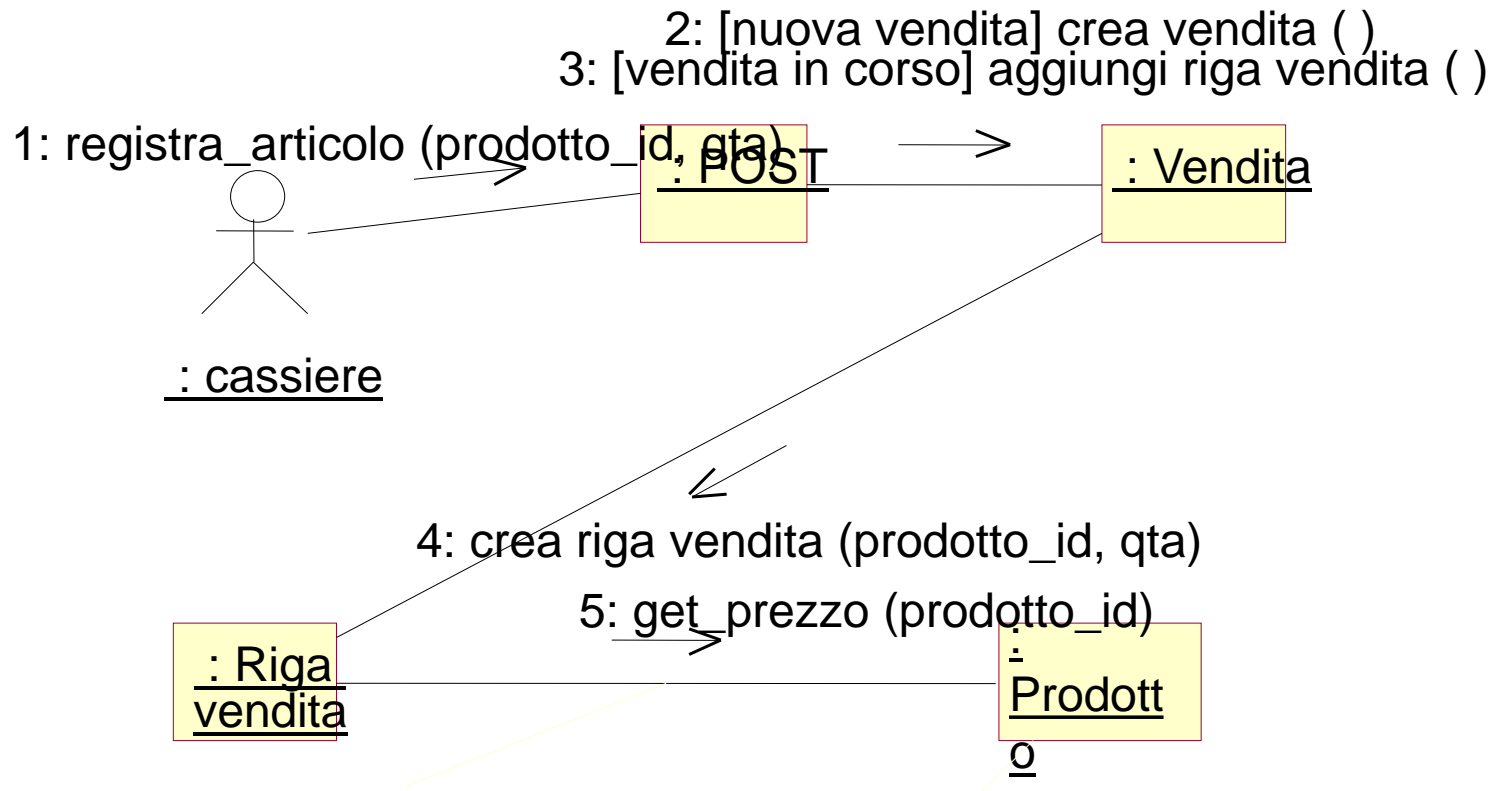
# Esempio completo: collaborazione



# Esempio completo: sequenza



# Esempio completo: collaborazione





- 
- **Activity Diagram**
  - **Use Case Diagram**
  - **Class Diagram**
  - **Sequence Diagram**
  - **Collaboration Diagram**
  - **Statechart Diagram**

# Statechart Diagram: diagrammi di stato

---

- Possono essere usati per descrivere il comportamento nel tempo di un particolare elemento come
  - un oggetto (ovvero una singola entità)
  - un intero sottosistema
- ovvero l'evoluzione di una interazione.



# Statechart Diagram: diagrammi di stato

---

In pratica essi descrivono

- sequenze di **stati** ed **azioni** attraverso cui l'elemento considerato passa durante la propria vita
- reagendo a eventi discreti (segnali, chiamate a funzionalità...).

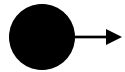
# Statechart Diagram: diagrammi di stato

---

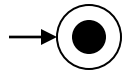
- Si possono pensare come “il contrario” degli Activity Diagram
- Enfasi posta sugli **stati** e non sulle **azioni**

# Statechart Diagram: simboli base

---



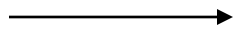
Inizio di un diagramma di stato



Termine di un diagramma di stato

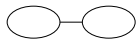
Stato 1

Singolo stato generico con nome “Stato 1”



Connessione fra stati

Stato 2



Stato scomponibile in un ulteriore diagramma di stati

Nome del diagramma di stato (in fondo)

# Statechart Diagram: simboli base

---

Stato 1

Do: attività

Stato con attività in corso per tutta la sua durata

Stato 1

Enter: attività

Stato con attività che accade all'ingresso in esso

Stato 1

Exit: attività

Stato con attività che accade all'uscita da esso

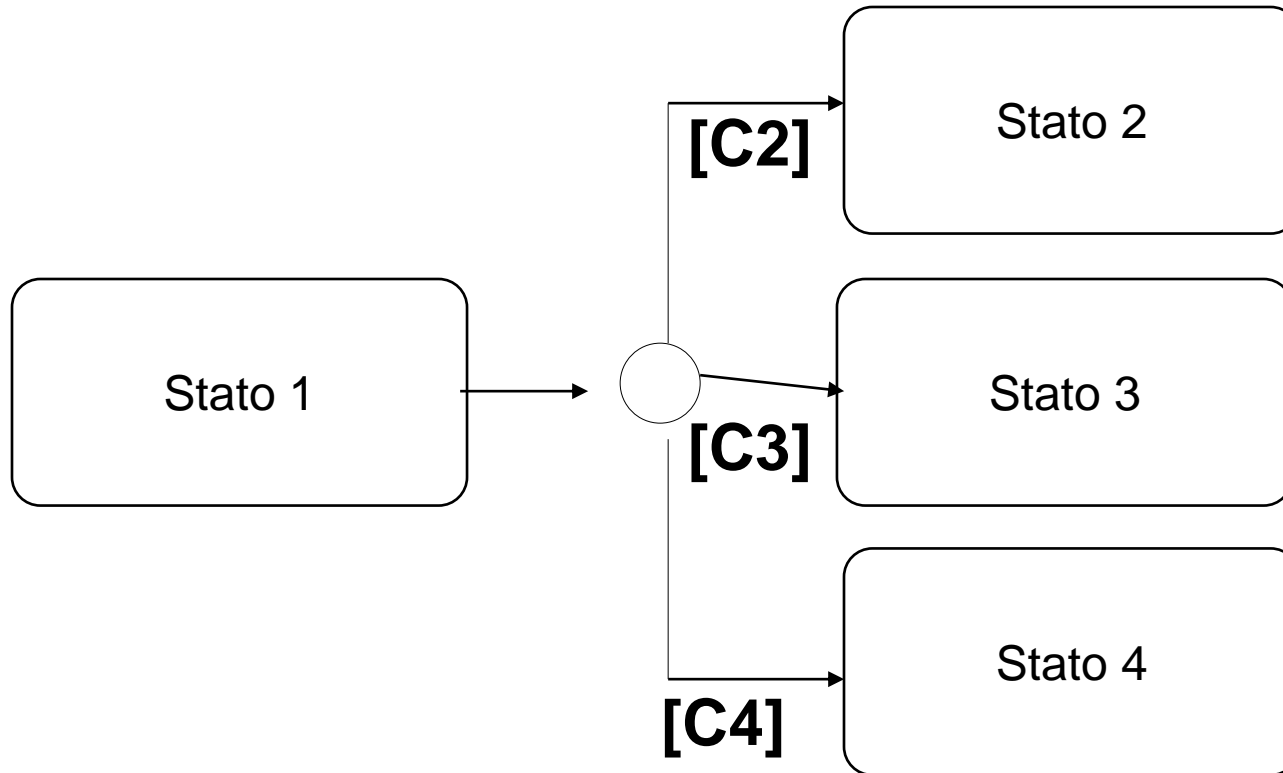
Stato 1

Include: attività

Stato che include un'attività compiuta su un altro diagramma

# Tipi di SD: punto di scelta dinamica

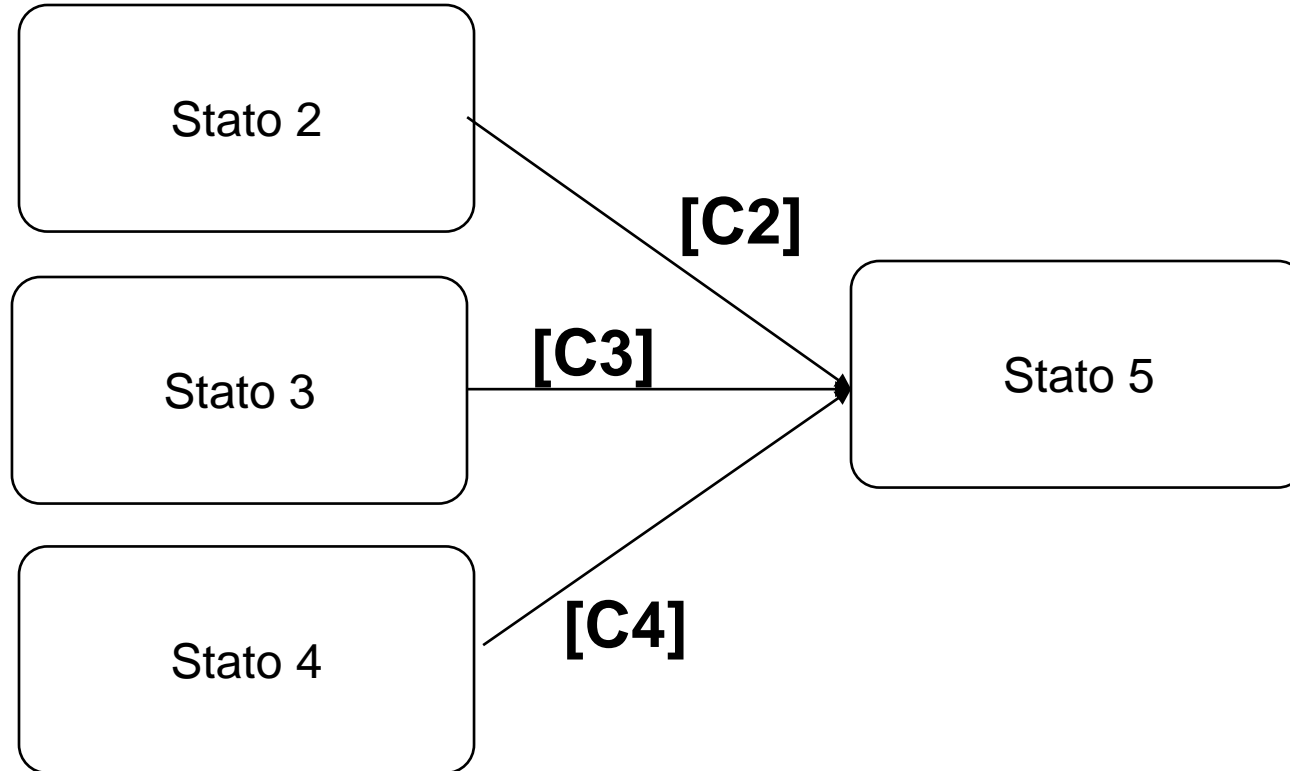
---



In base alle condizioni si passa ad uno degli stati.  
Forma equivalente: le frecce partono direttamente da  
Stato 1

# Tipi di SD: punto di giunzione

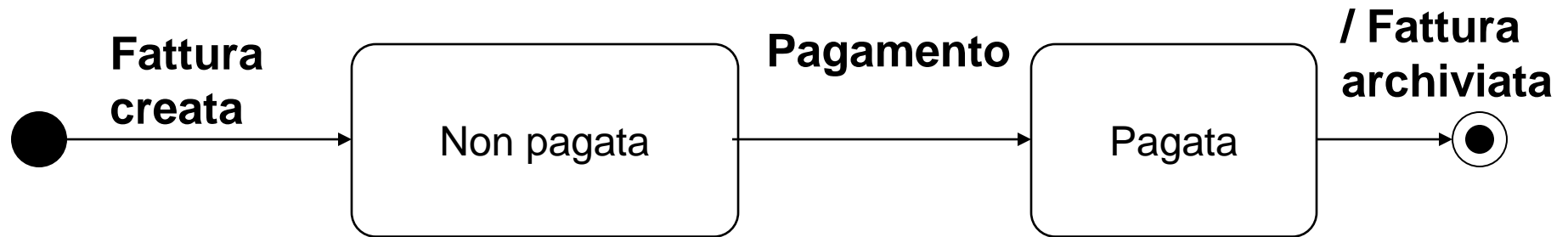
---



Il successivo di tutti e 3 gli stati, seguendo le condizioni, è Stato 5

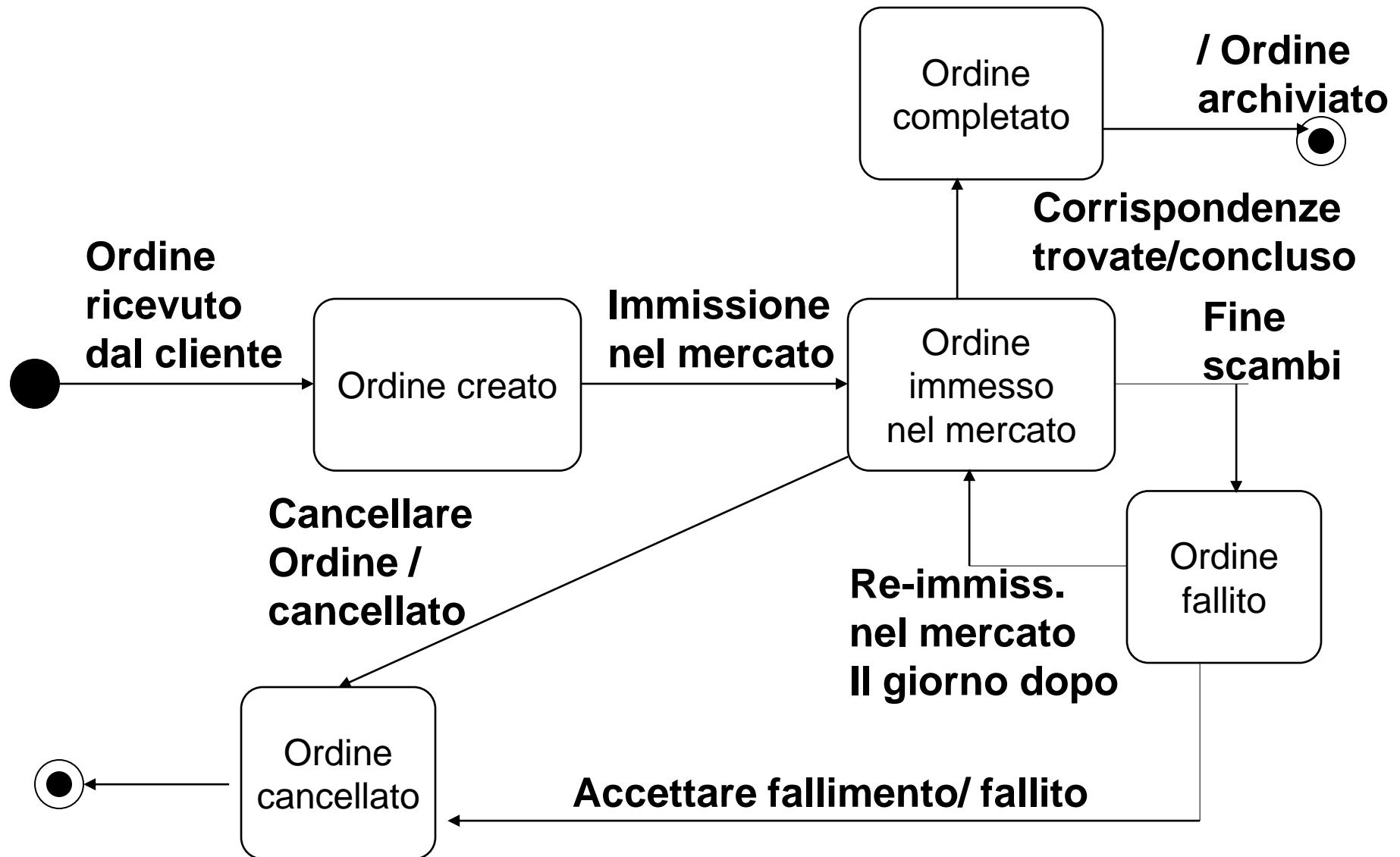
# Statechart Diagram: esempio

---



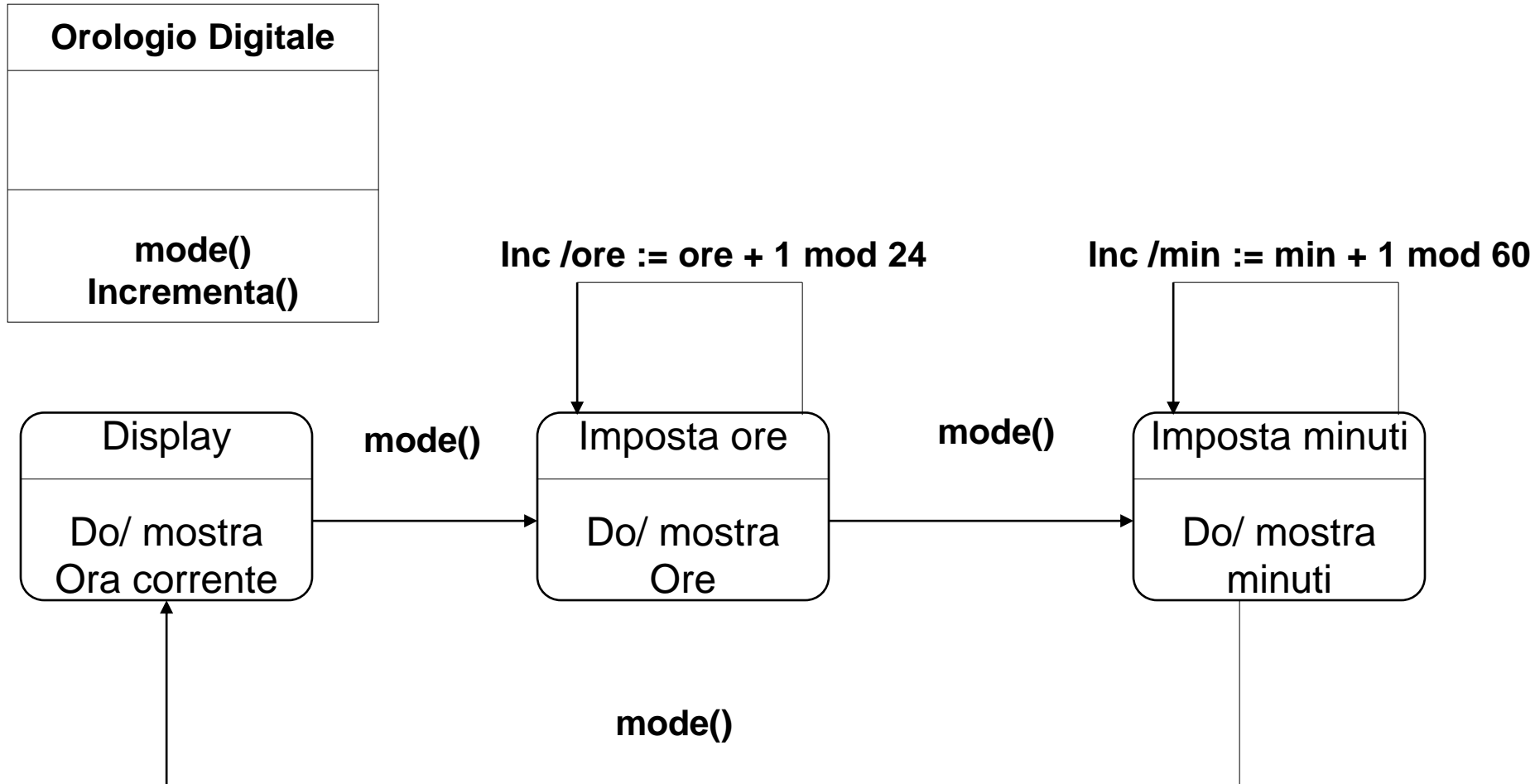
I rettangoli rappresentano uno stato per le entità in gioco

# Statechart Diagram: terminazione a due possibilità





# Statechart Diagram: inserimento di oggetti



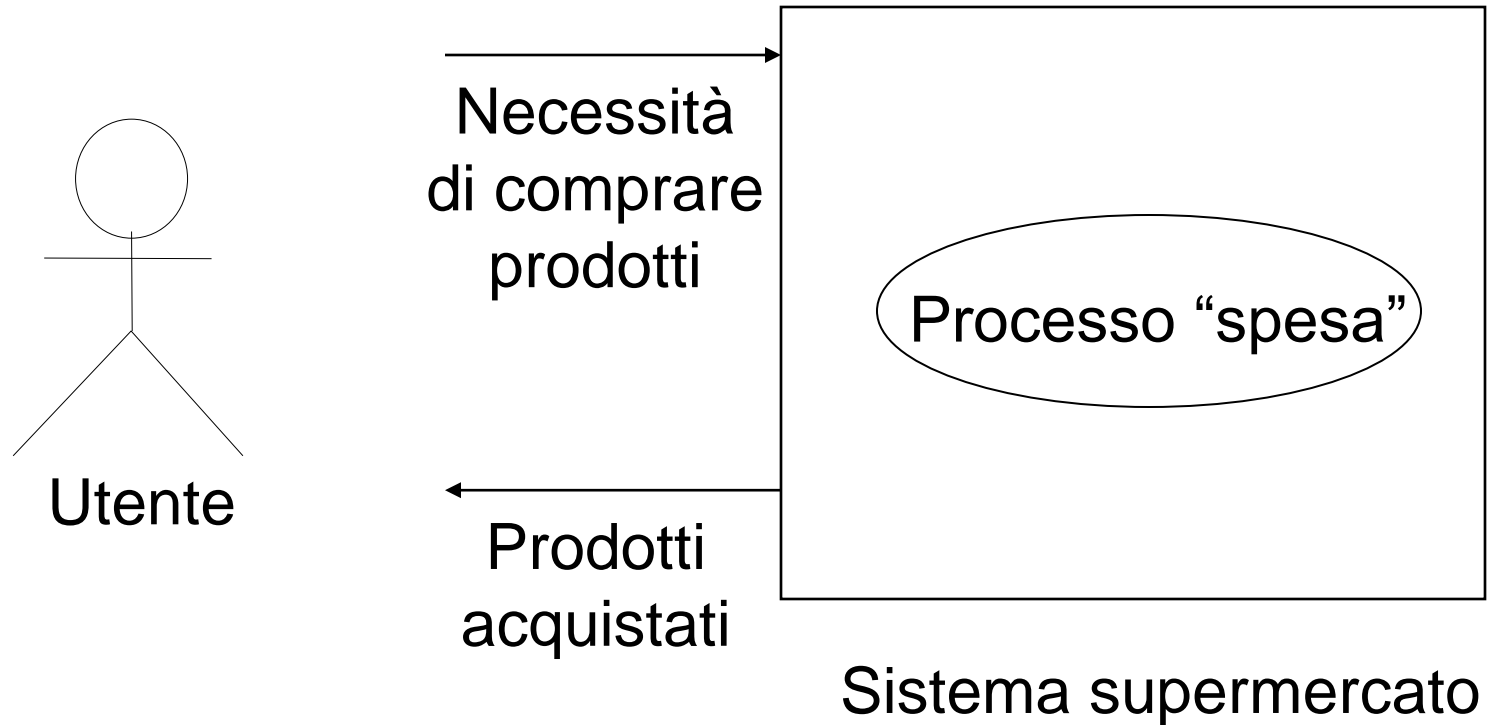
# Argomenti

---

- L'analisi dei processi business
- UML ed il suo ruolo
- Analisi del processo come successione di attività
- Analisi del processo come successione di casi d'uso
- Analisi delle entità che prendono parte ai processi
- Analisi delle interazioni tra gli elementi di un processo
- Un esempio completo di analisi
- Una visione d'insieme: il legame fra le viste e i pattern

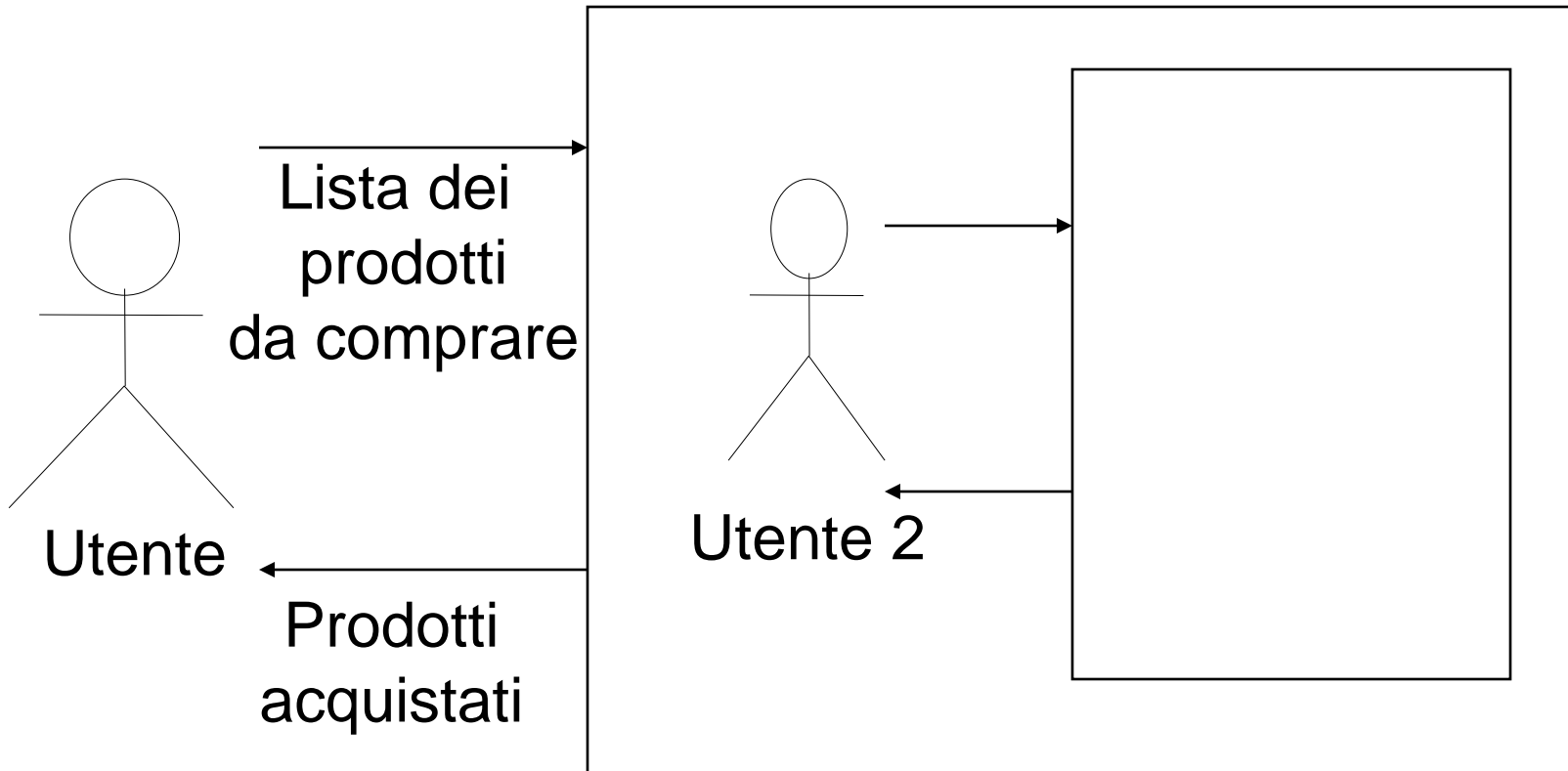
# Un processo del mondo reale: la spesa

---



# Quali sono i limiti di un sistema?

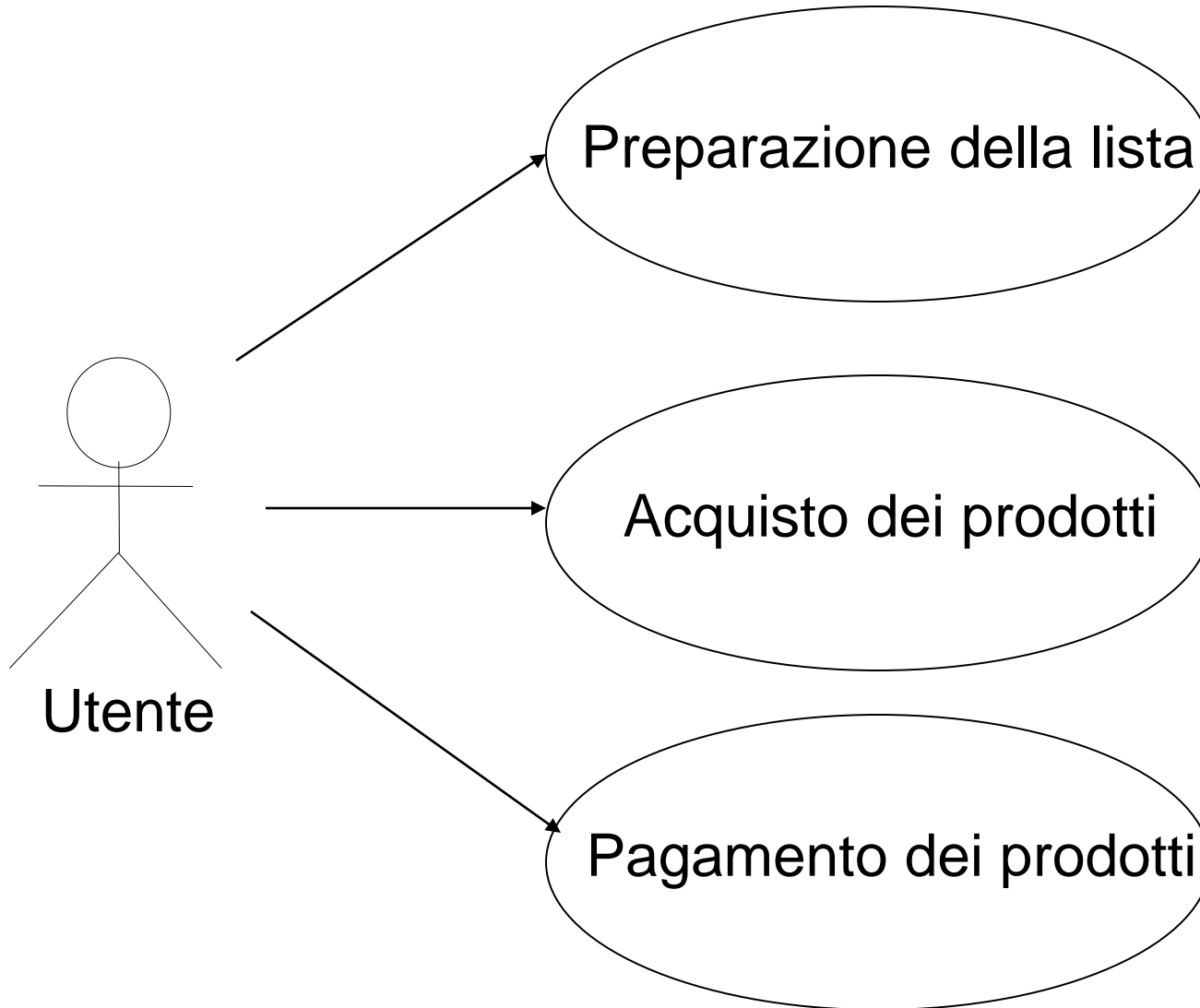
---



Sistema: utente 2 + negozio  
Processo: “fai la spesa”

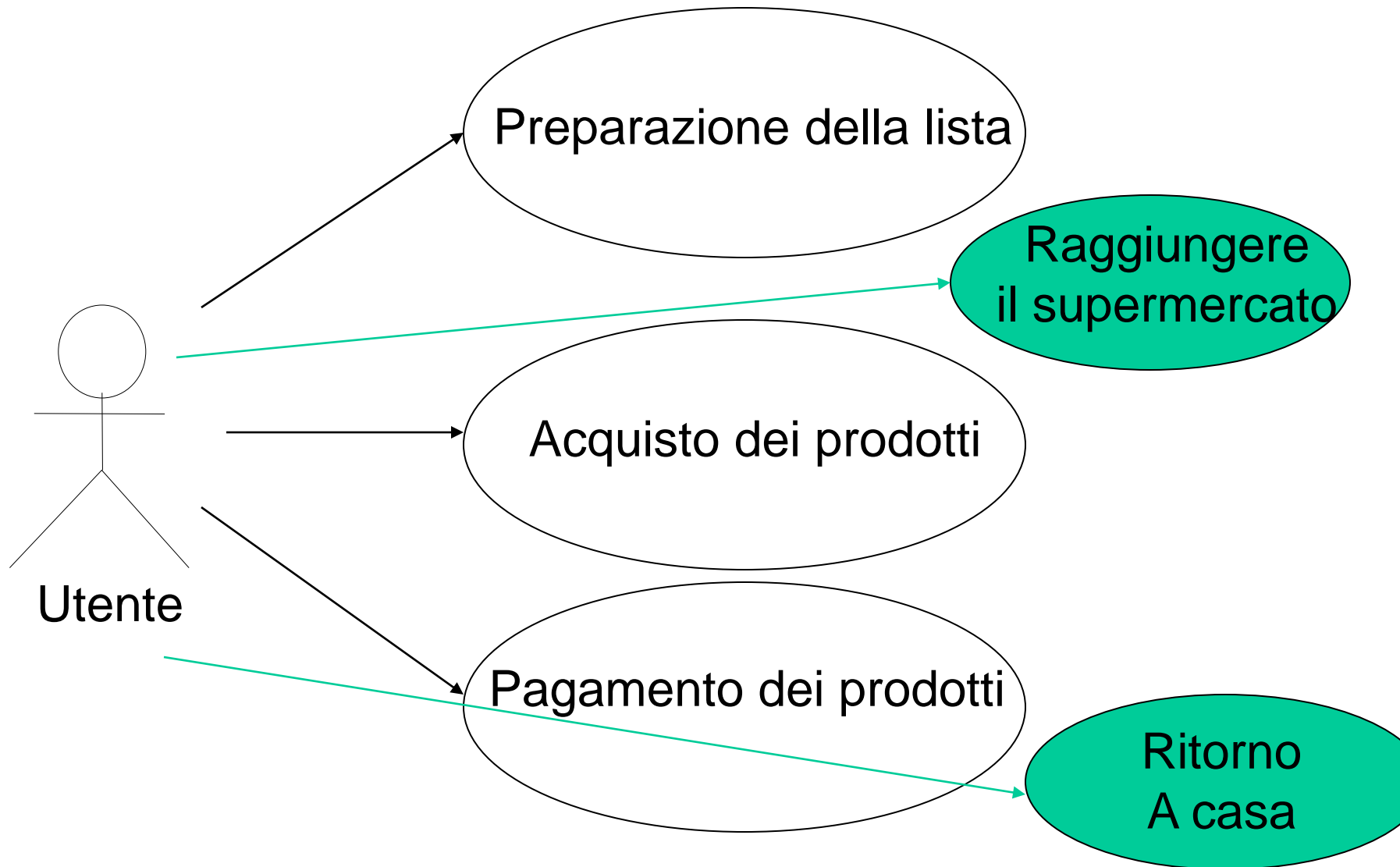
# Le funzionalità della spesa: use case

---



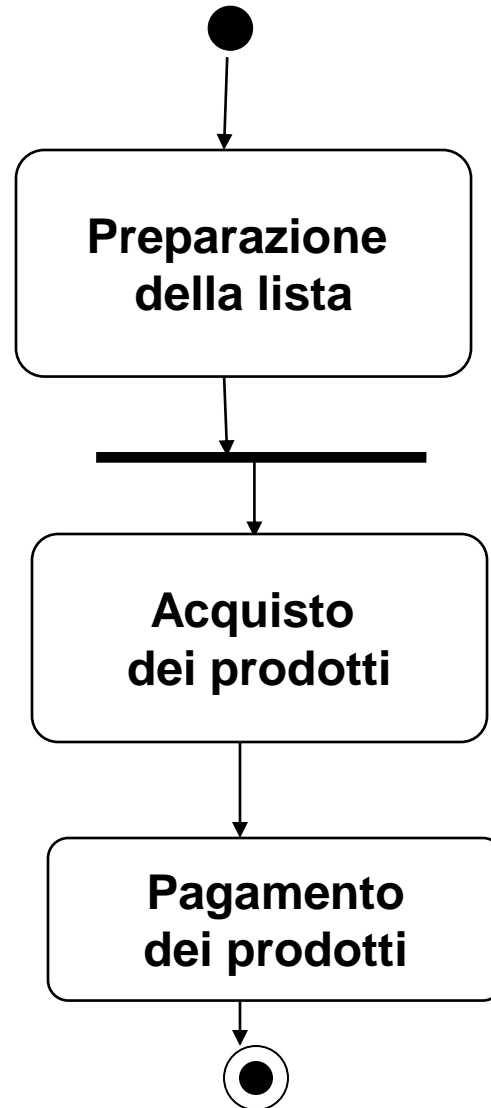
# Le funzionalità della spesa: use case

---



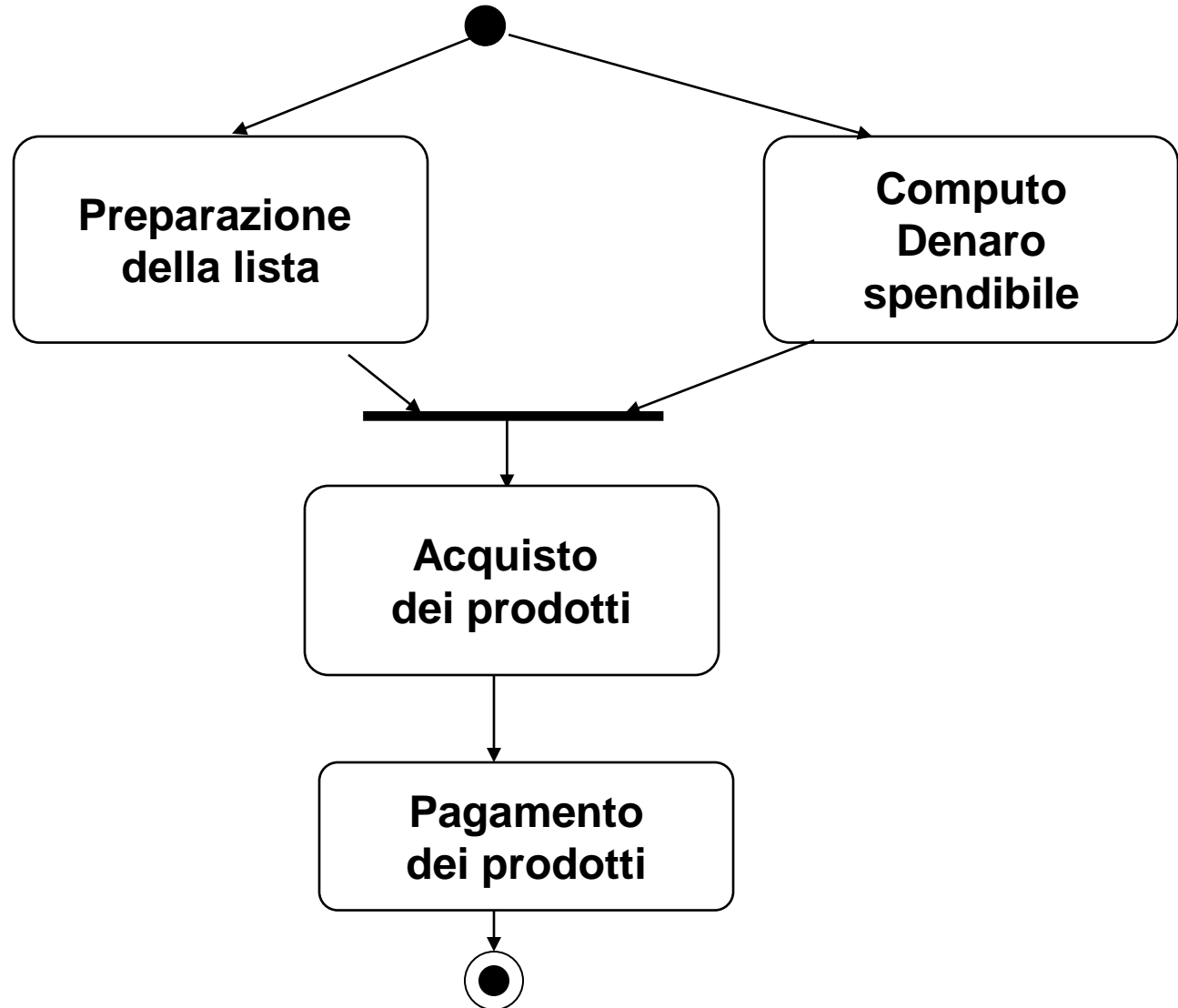
# Spesa: Diagramma di attività 1

---



# Spesa: Diagramma di attività 1

---

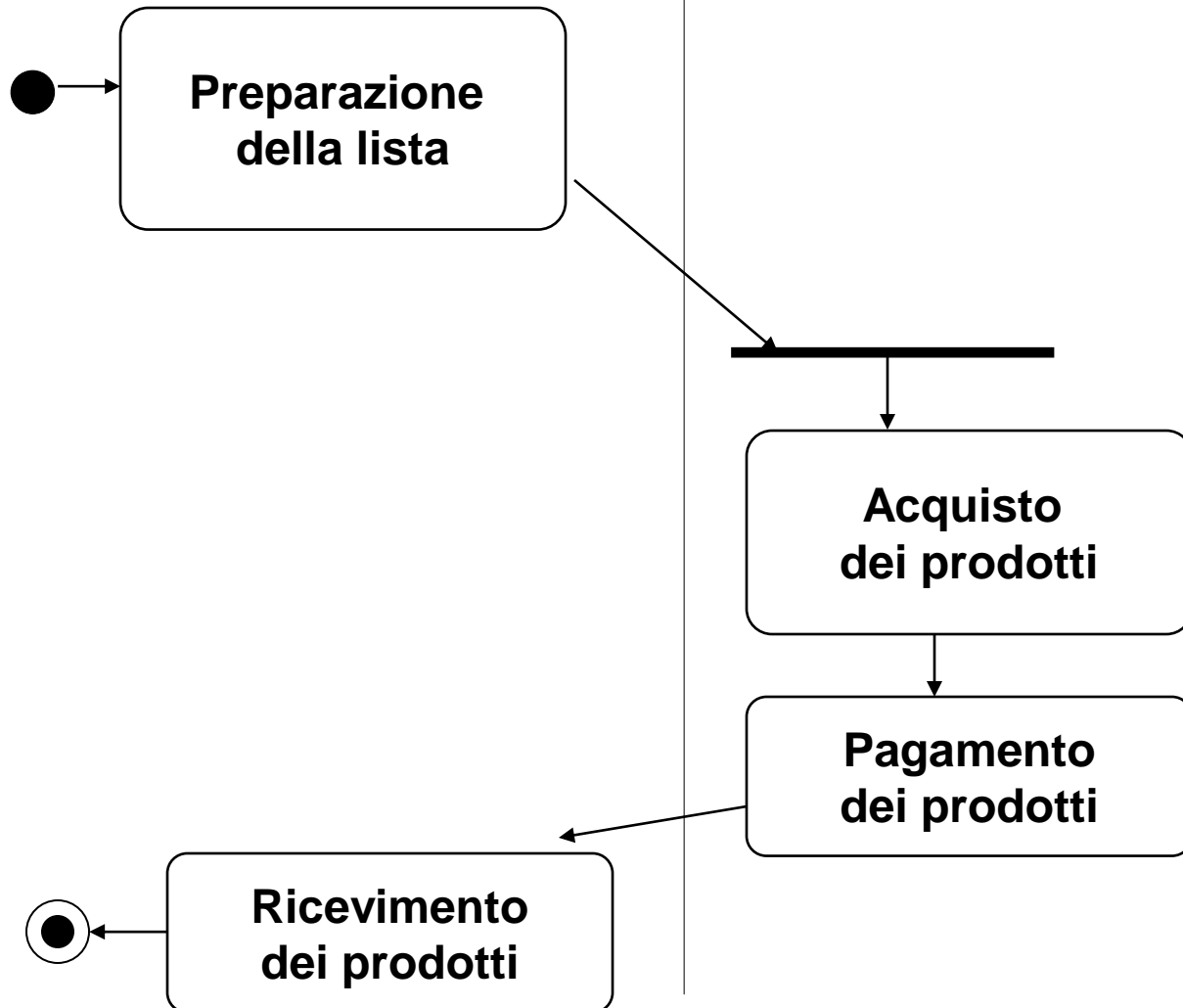




# Spesa: Diagramma di attività 2

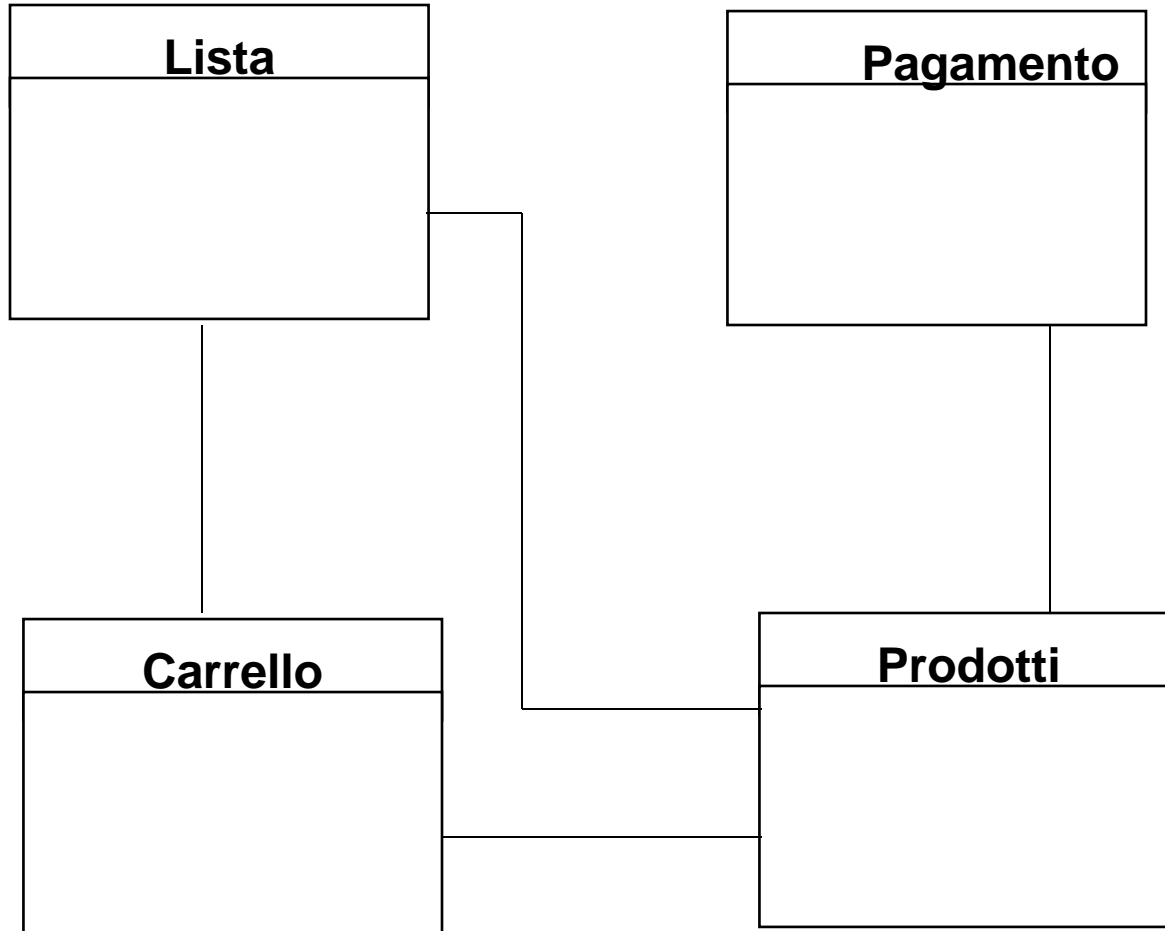
Utente 1: capo

Utente 2: esecutore

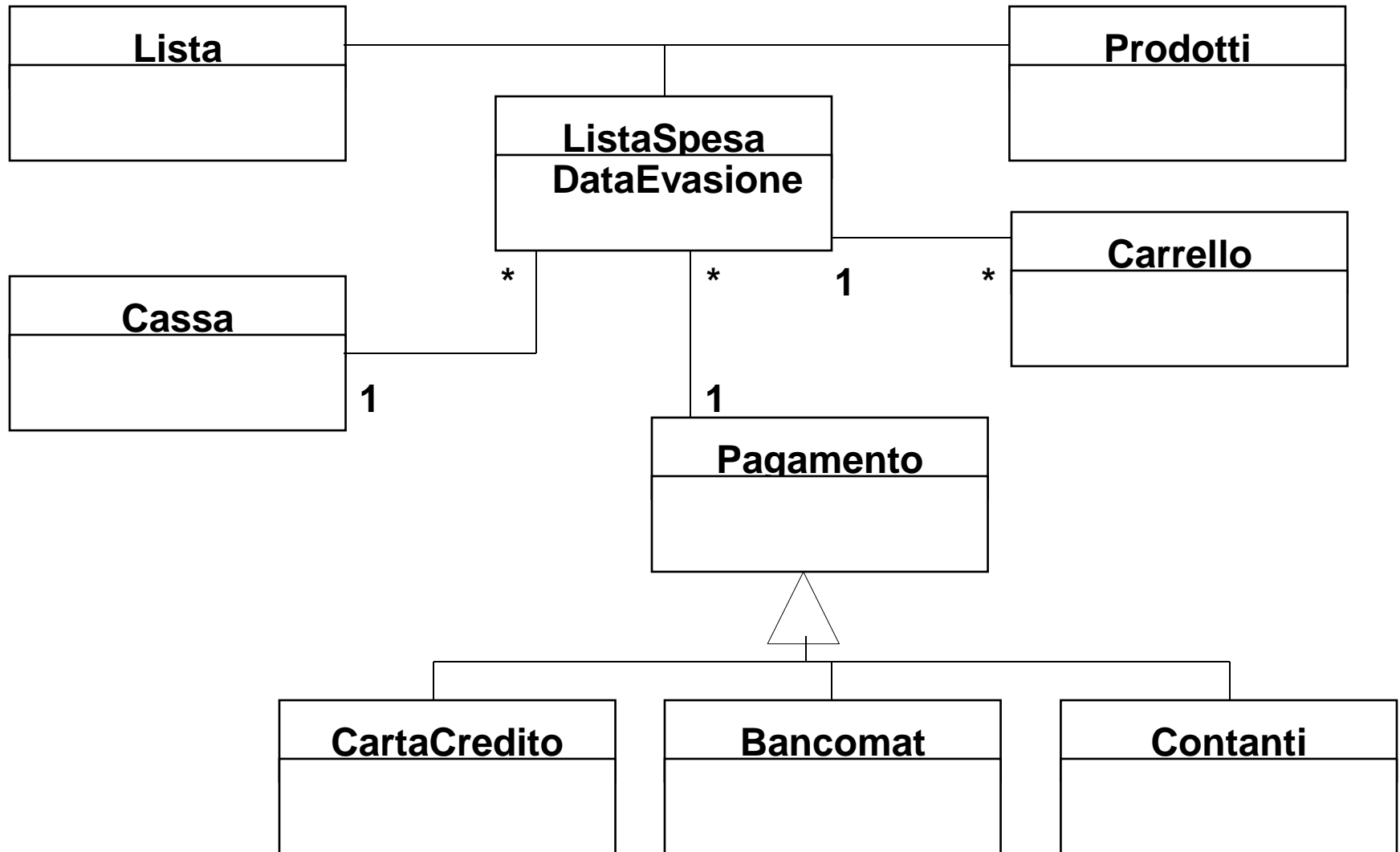


# Spesa: cercare le classi

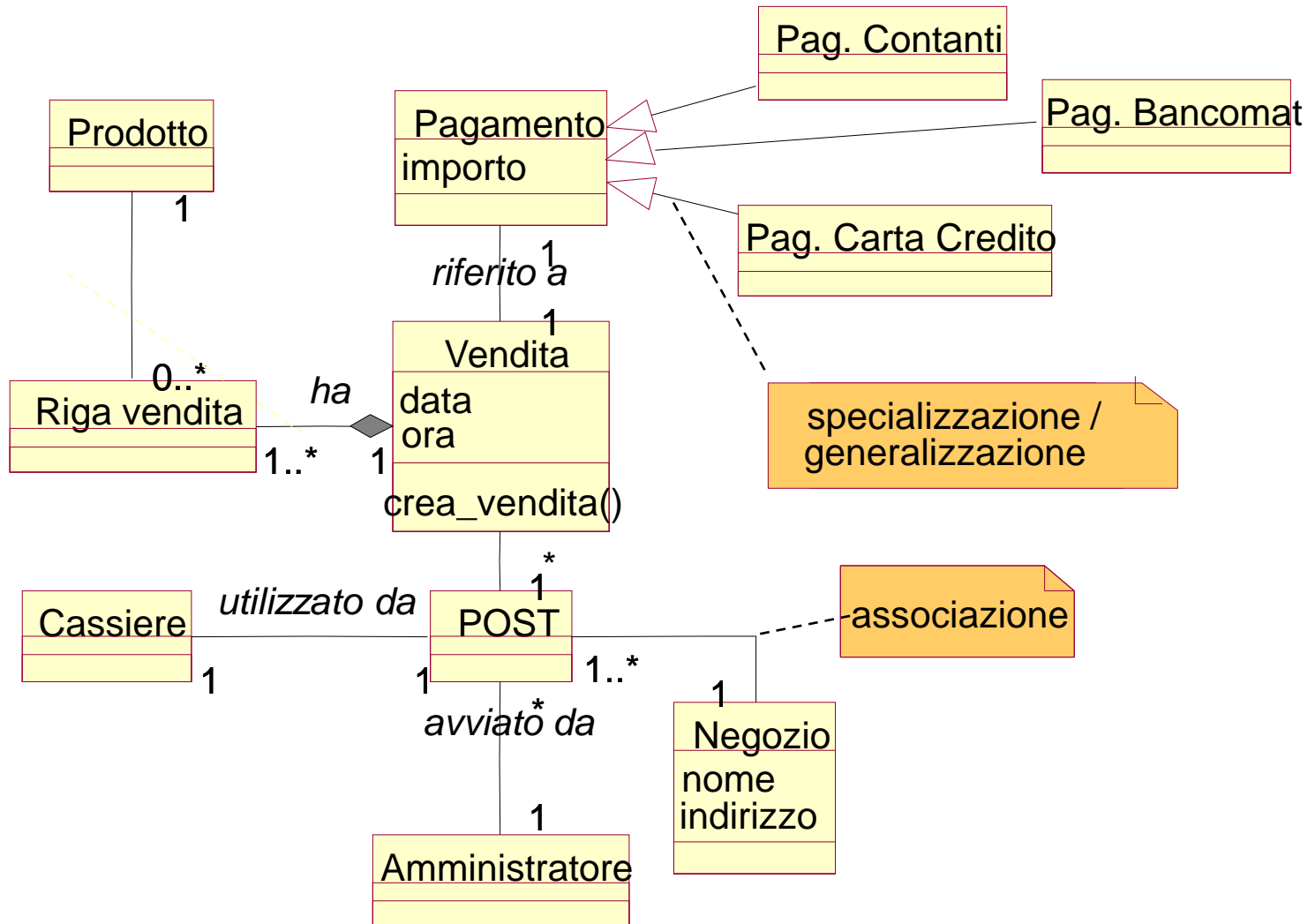
---



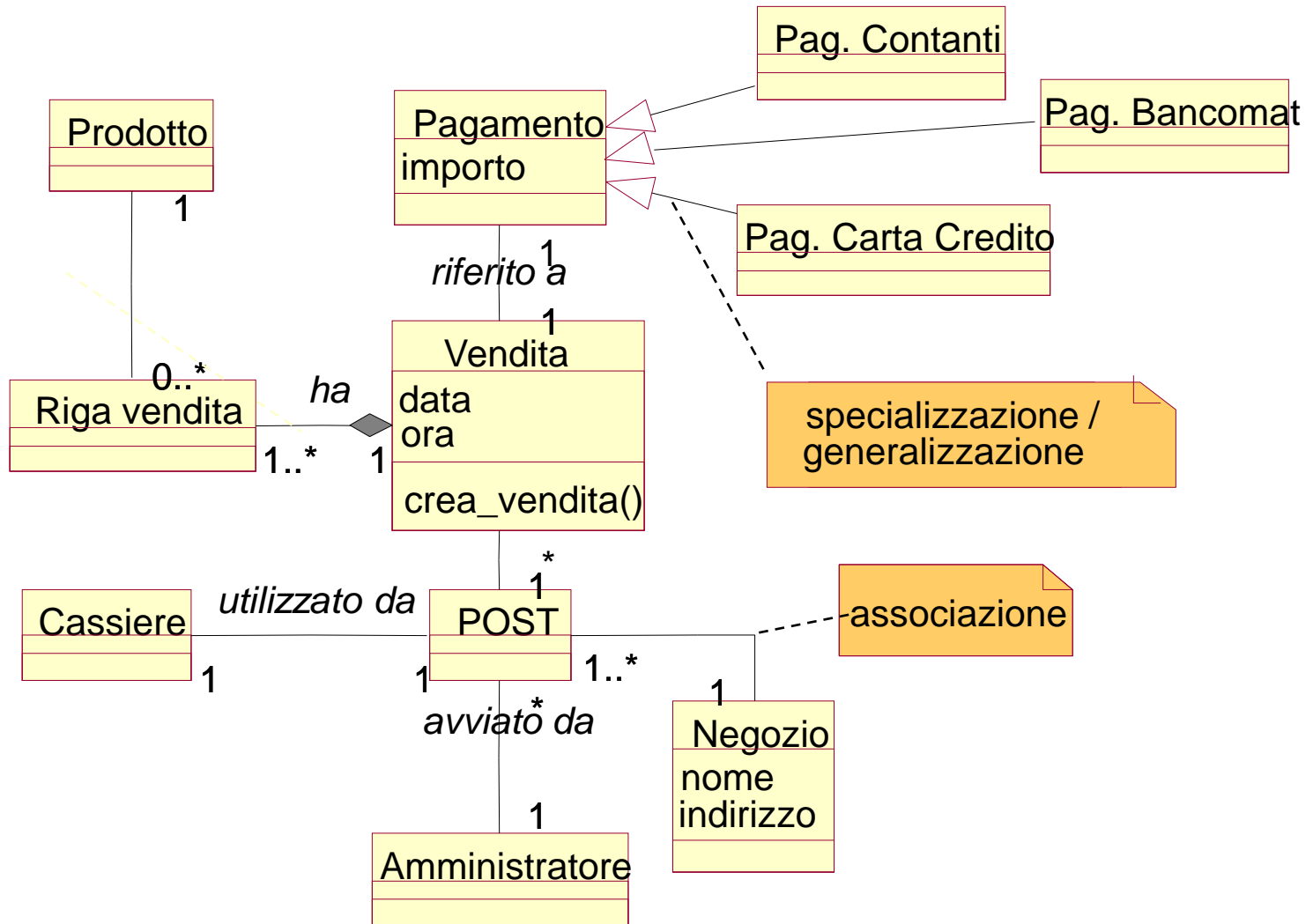
# Spesa: Diagramma delle classi



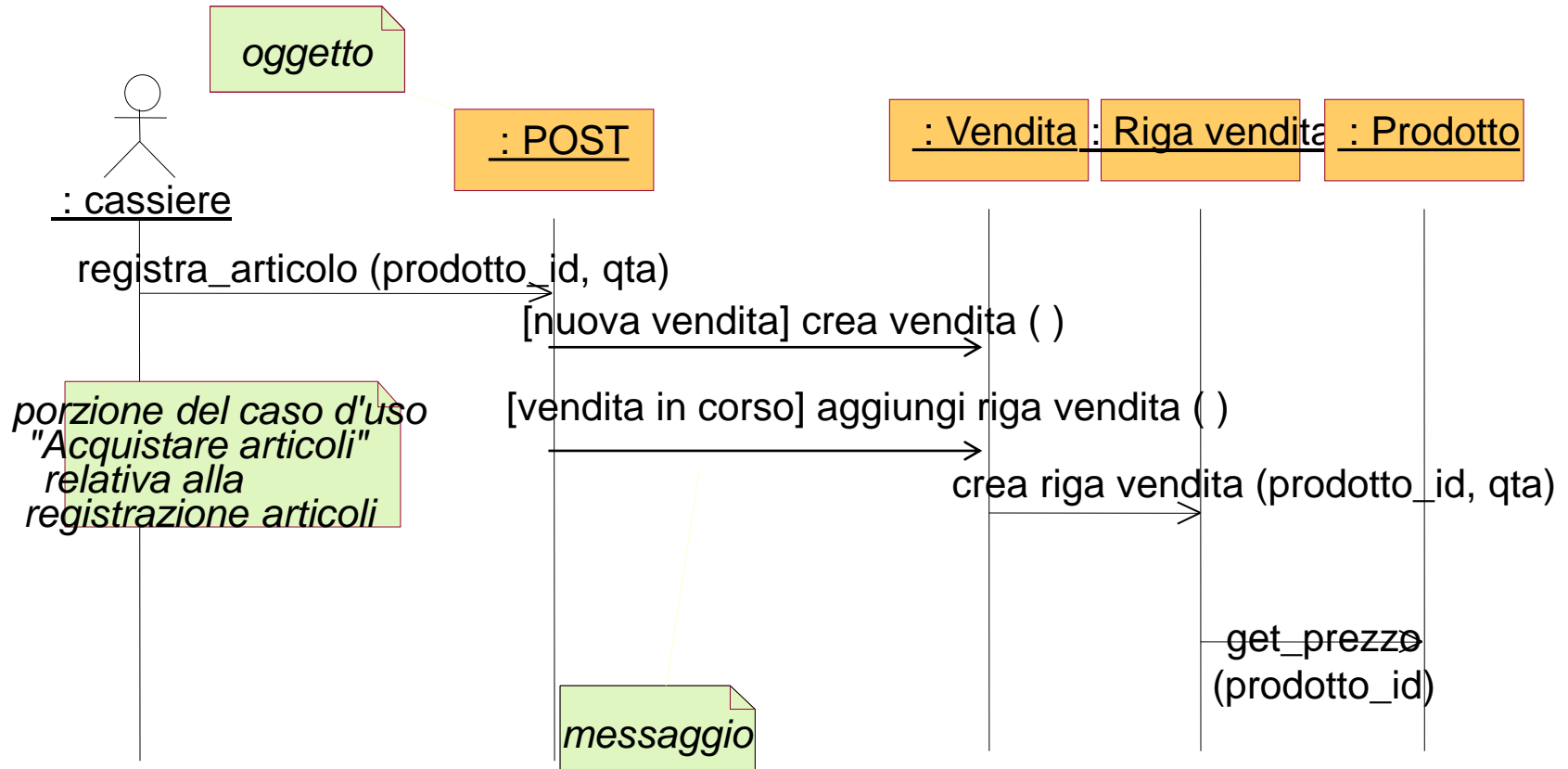
# Spesa: Class Diagram completo



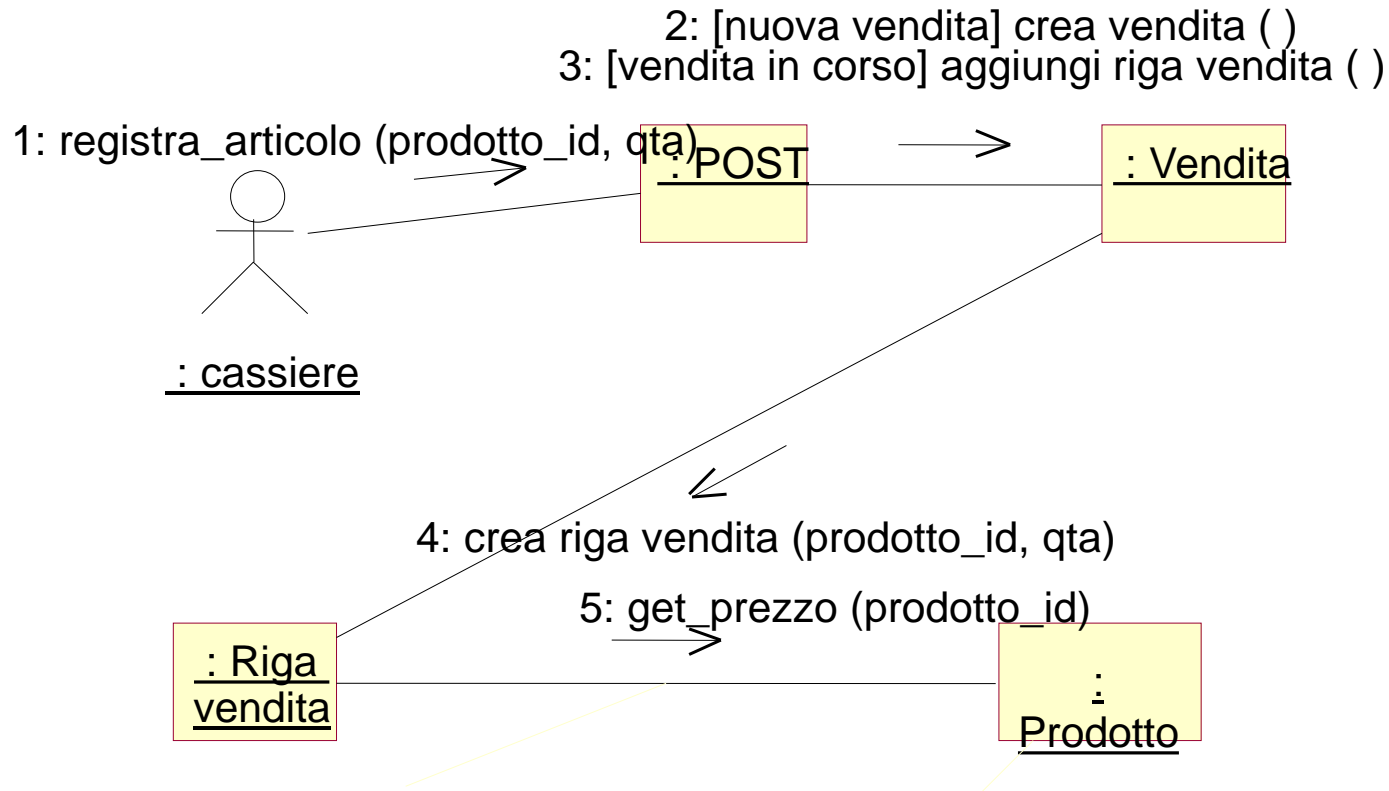
# Spesa: Class Diagram completo



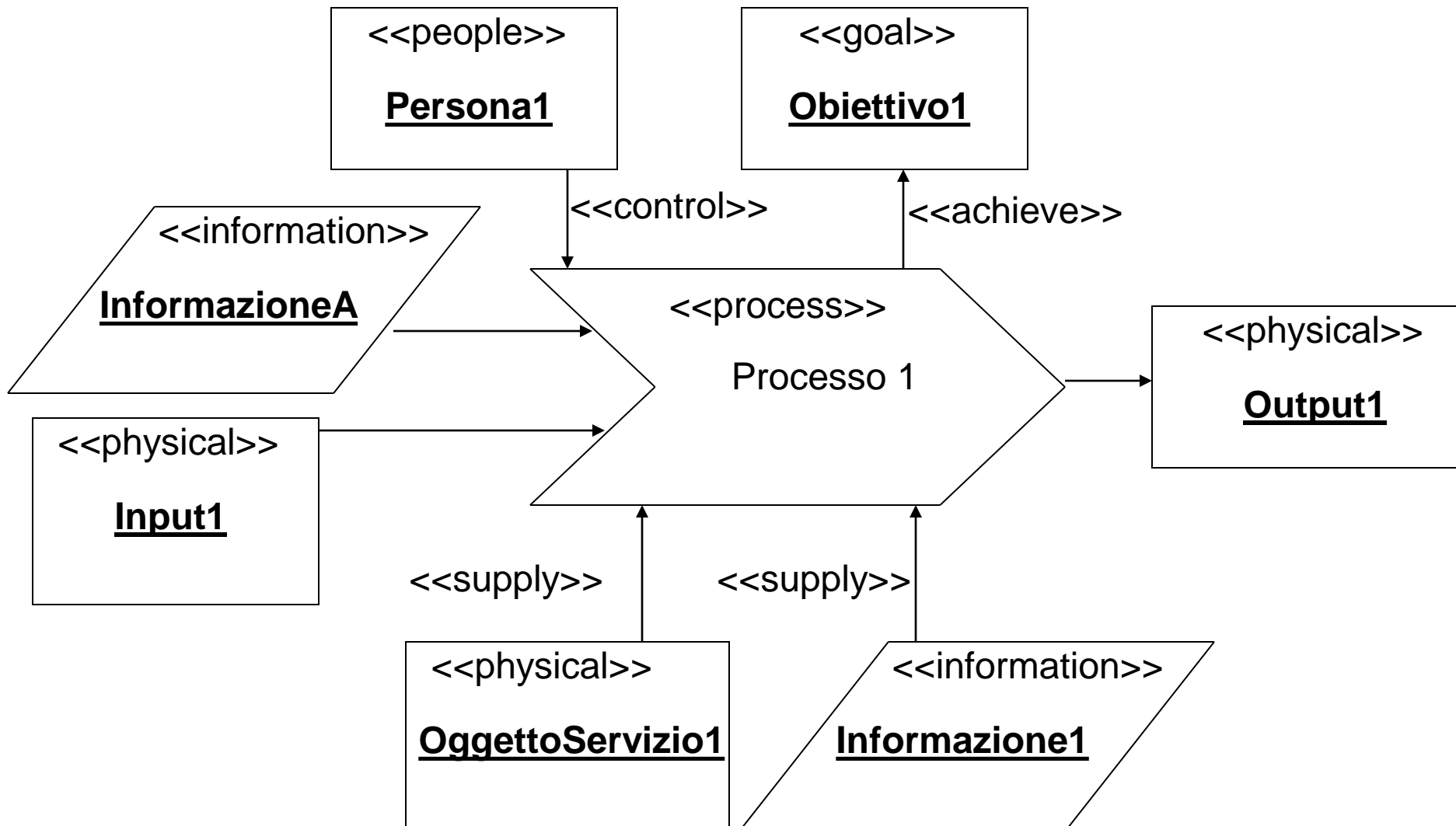
# Spesa: diagramma di sequenza



# Spesa: collaborazione

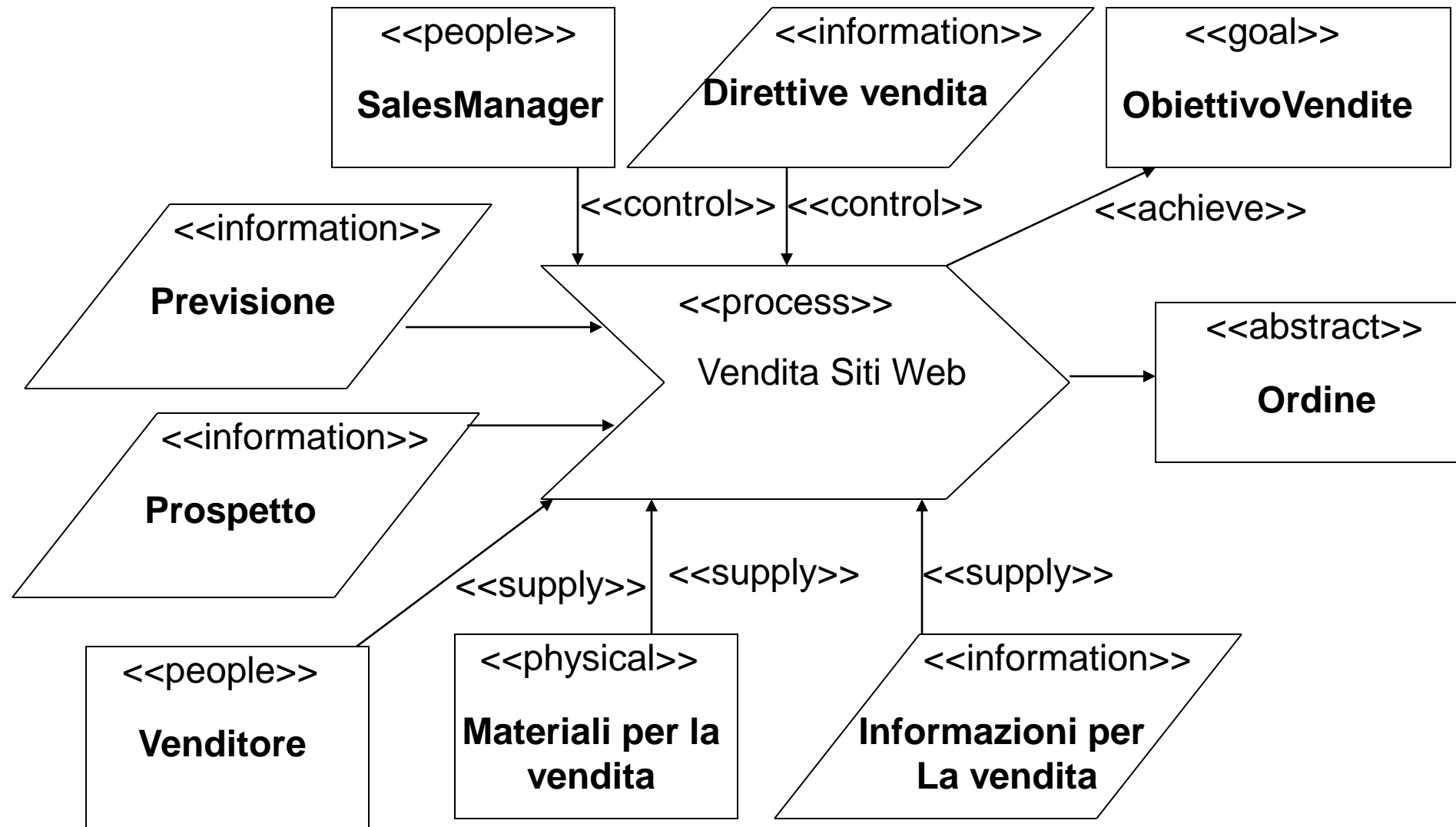


# Activity Diagram: il processo generico





# Esempio di AD: vendita di siti Web

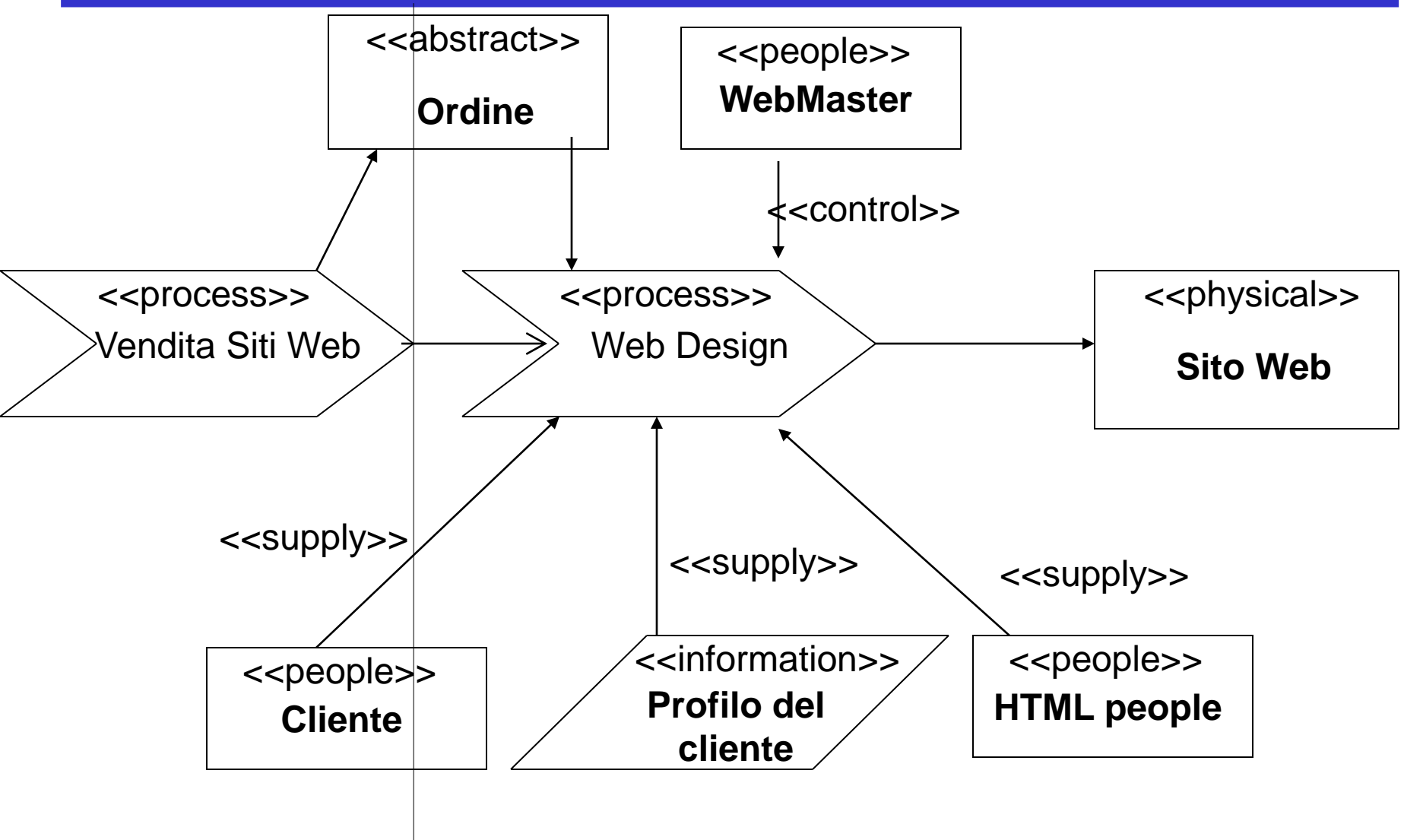


# Un esempio di goal: l'obiettivo vendite espanso

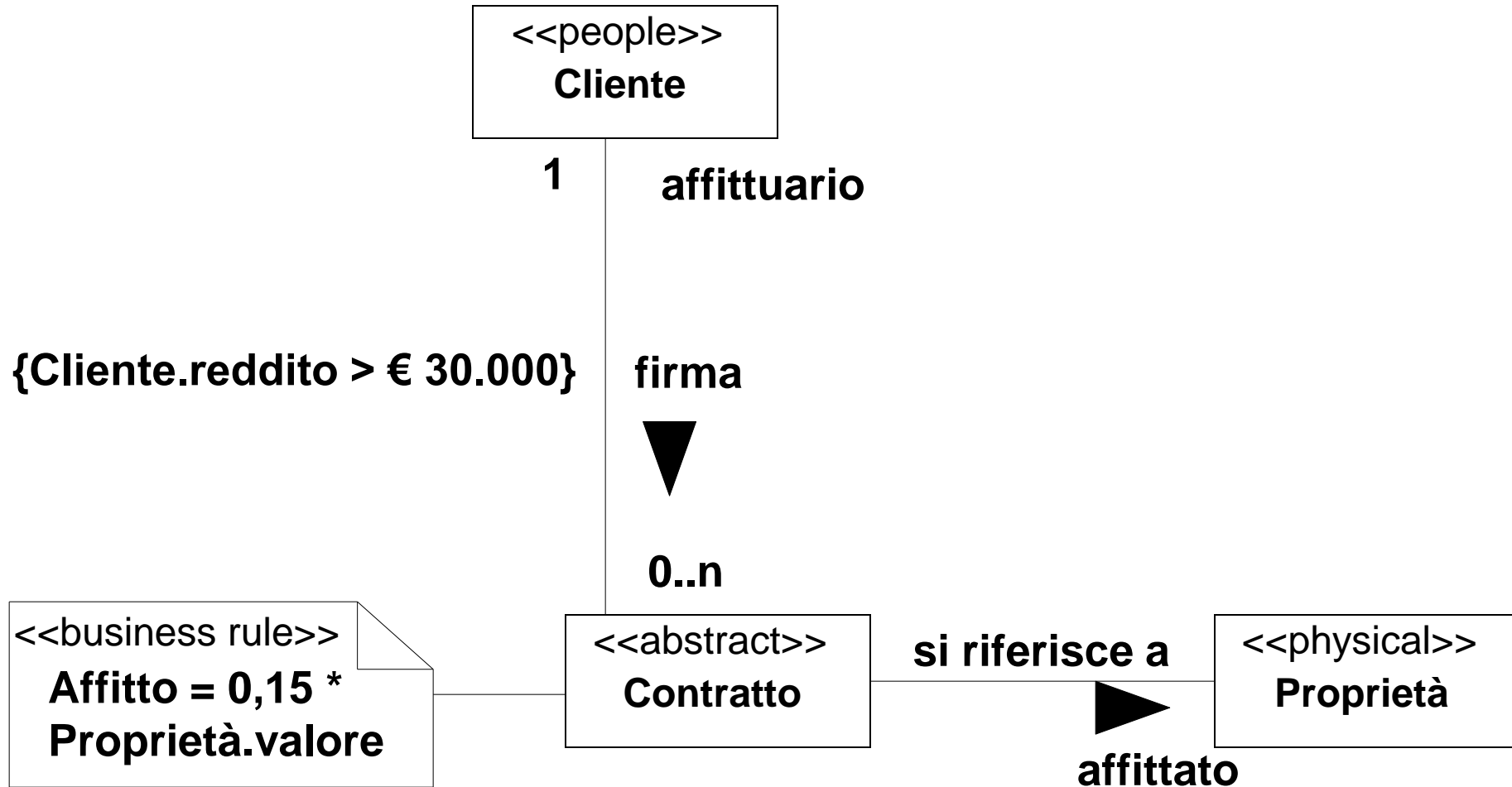
---

<p>&lt;&lt;goal&gt;&gt;</p> <p><b>ObiettivoVendite:</b> <b>quantitativo</b></p>
<p>fatturato: valuta = € 125.000,00</p> <p>costi: valuta = € 75.000,00</p> <p>scadenza: data = 31/12/2003</p>

# Combinazione AD/swimlanes

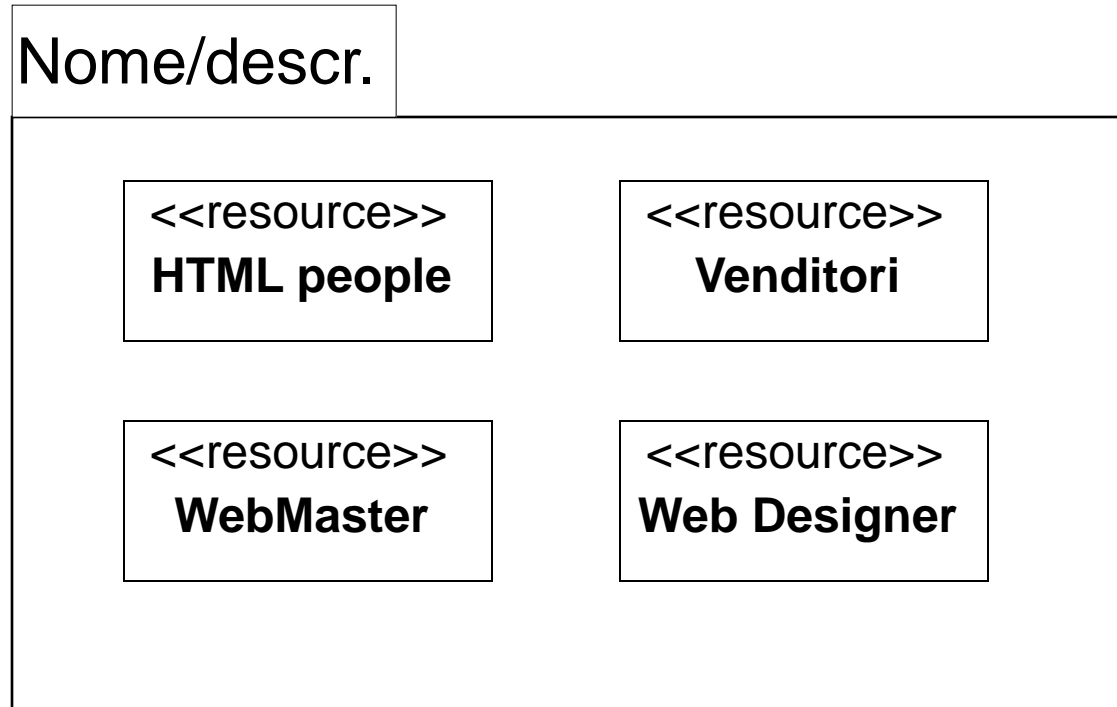


# Class Diagram per regole business



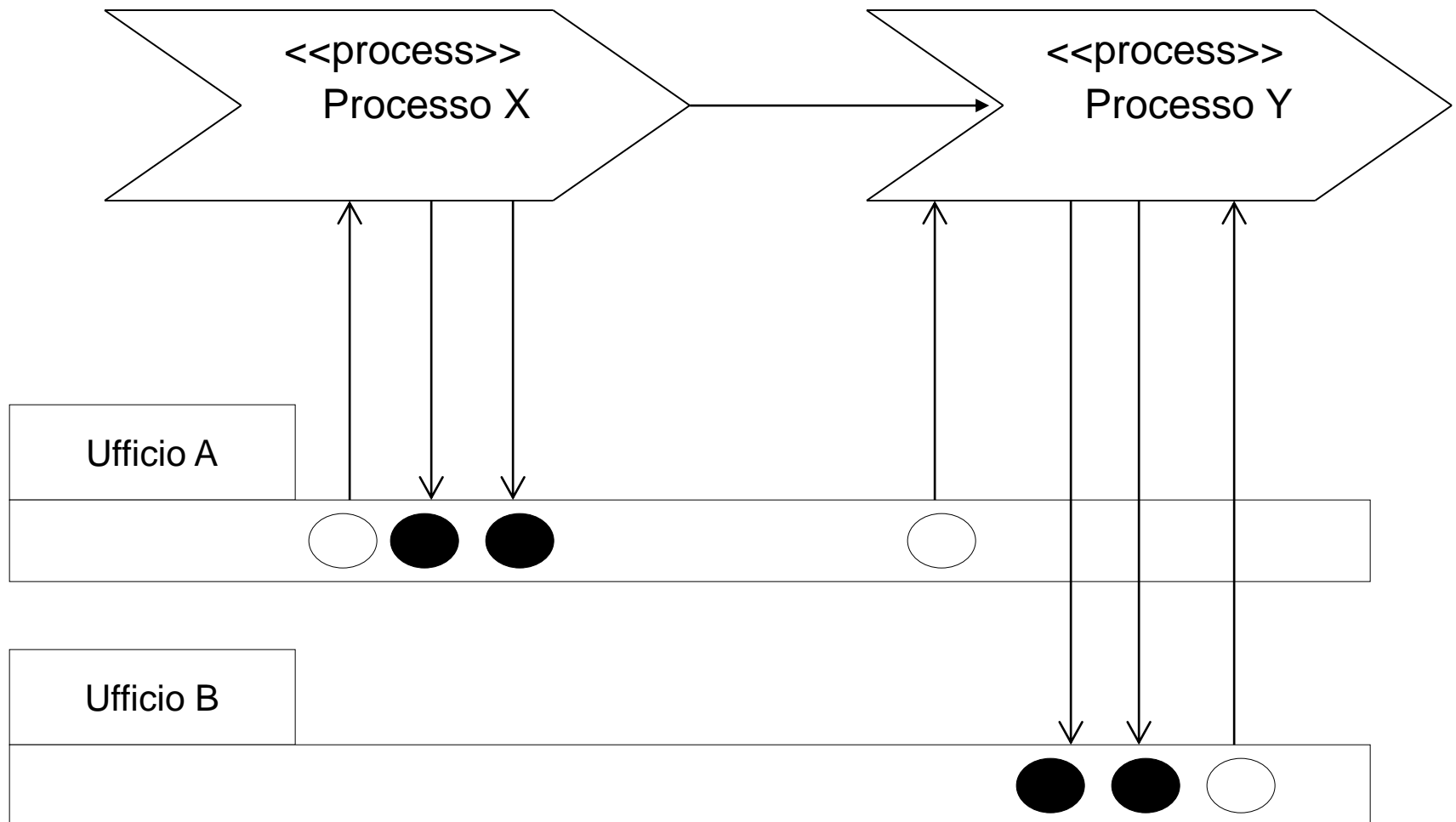
# I Package di UML

---

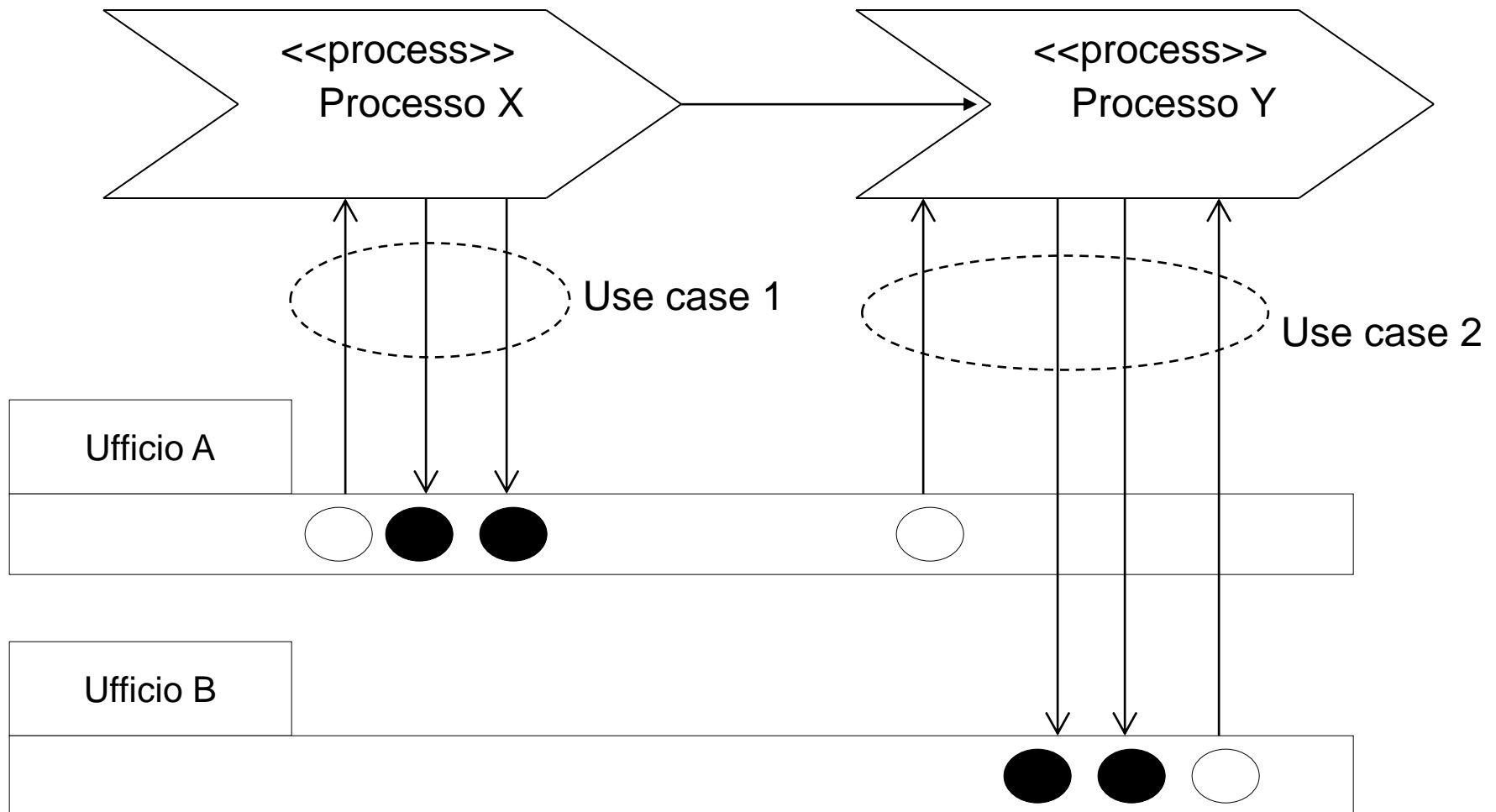


**Un package è un raggruppamento generale di elementi correlati fra loro da un legame logico che il modellatore ritiene importante**

# Assembly line



# Il legame tra processi e use case



# Argomenti

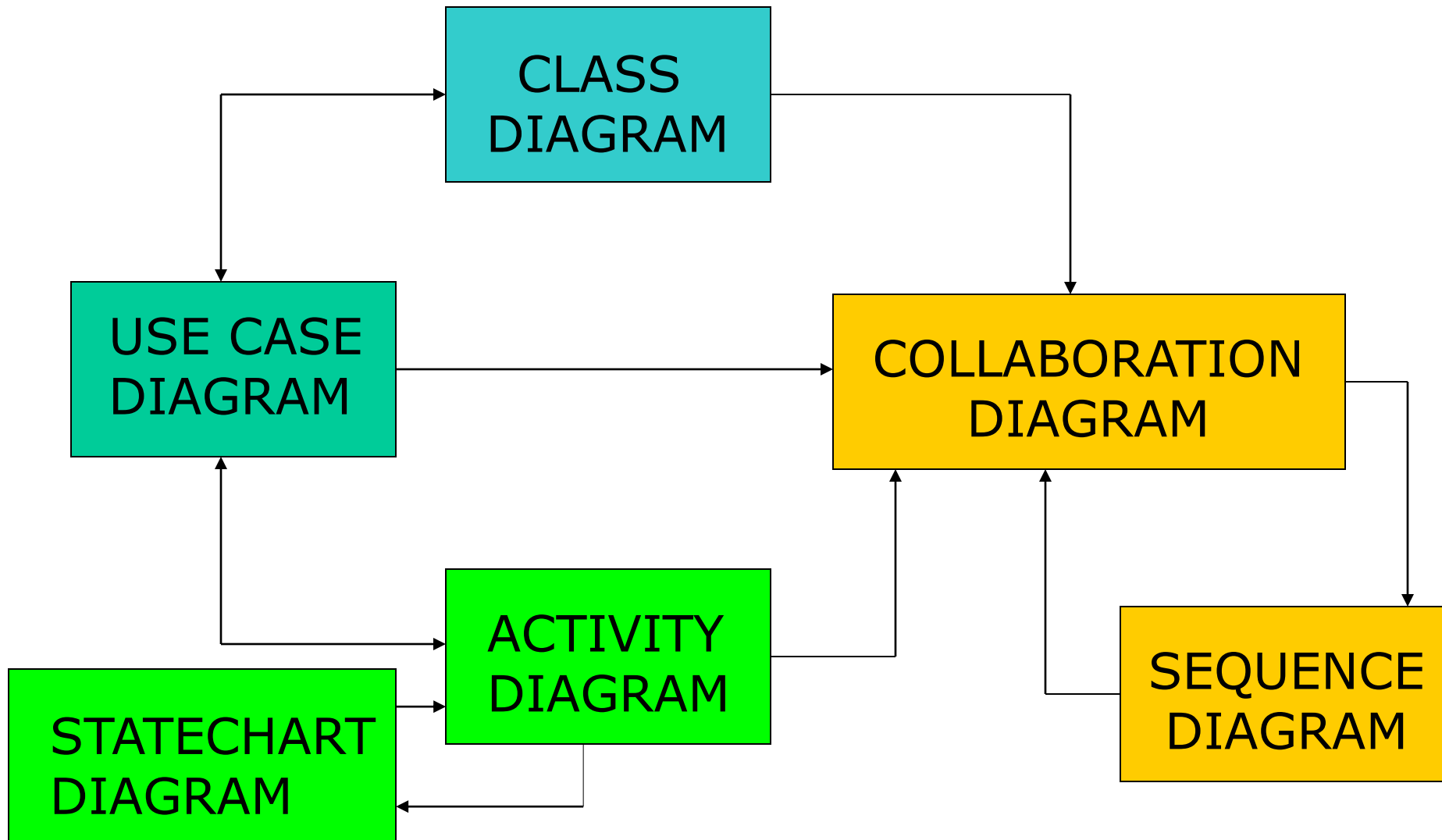
---

- L'analisi dei processi business
- UML ed il suo ruolo
- Analisi del processo come successione di attività
- Analisi del processo come successione di casi d'uso
- Analisi delle entità che prendono parte ai processi
- Analisi delle interazioni tra gli elementi di un processo
- Un esempio completo di analisi
- Una visione d'insieme: il legame fra le viste e i pattern



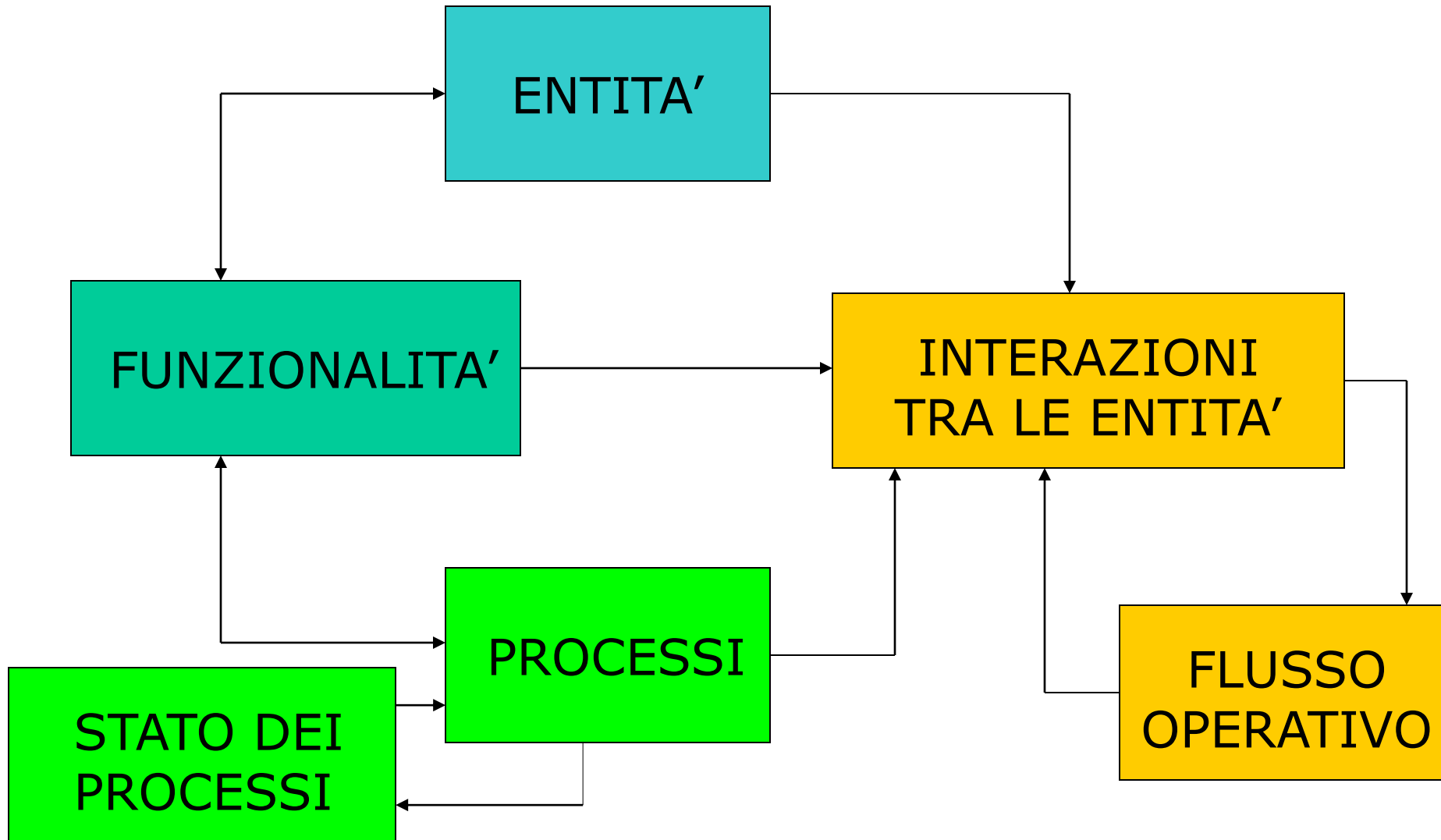
# Il legame fra i diagrammi UML

---



# Il legame fra i diagrammi UML

---



# Un Pattern...

---

- descrive un problema
- che ricorre in specifici contesti
- propone un generico, ma ben dimostrato, schema per la sua soluzione.

# Quindi un Pattern è

---

- una soluzione
  - ad un determinato problema
  - in un definito contesto
- 
- che, talvolta, può essere generalizzata ed adattata a molti contesti diversi

# E un Business Pattern

---

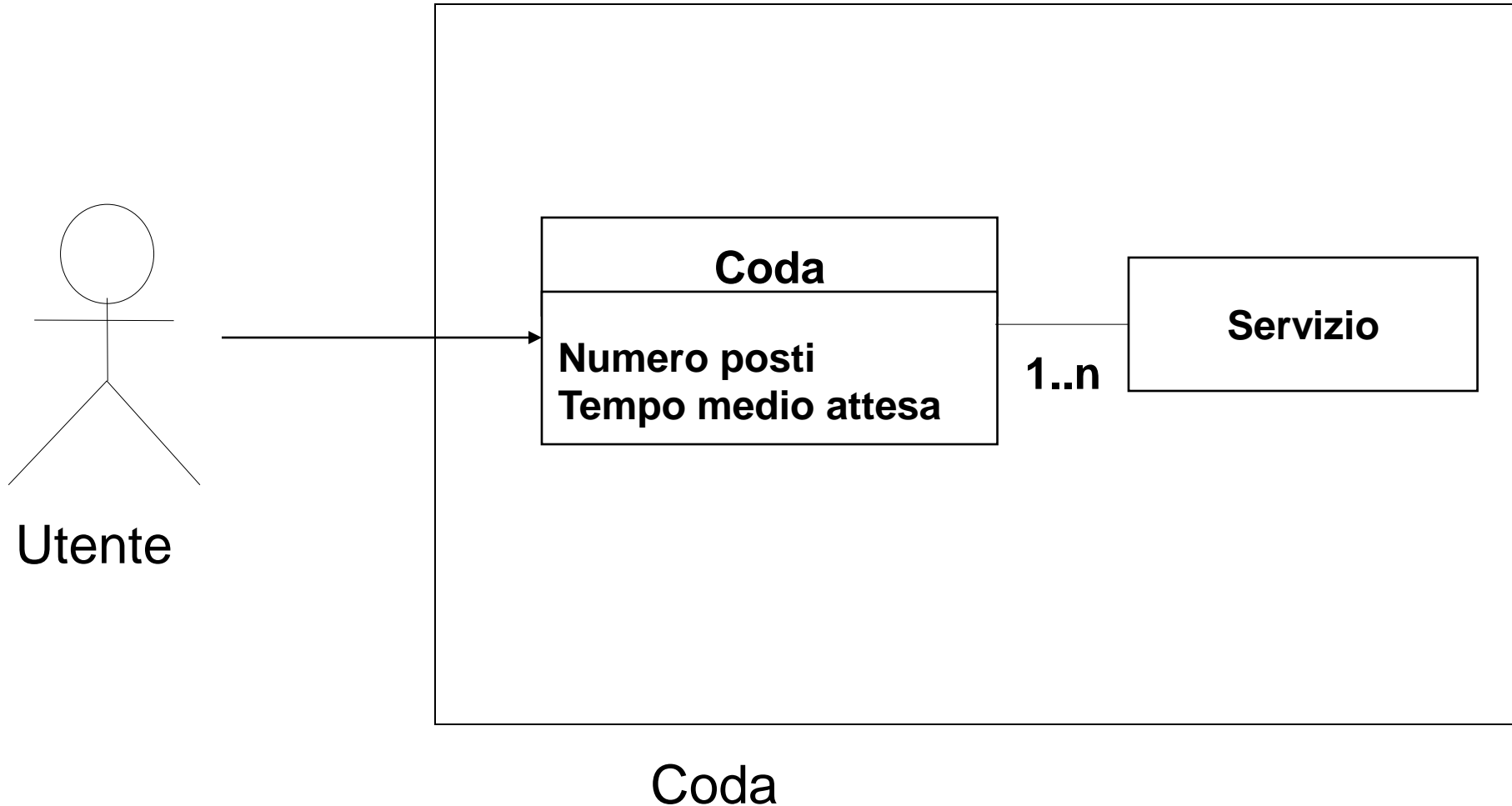
- Si riferisce a problemi di Business
- Tipicamente analizza situazioni di modellizzare e/o strutturare risorse business che comprendono documenti, organizzazione, informazioni

# Suddivisione dei Business Pattern

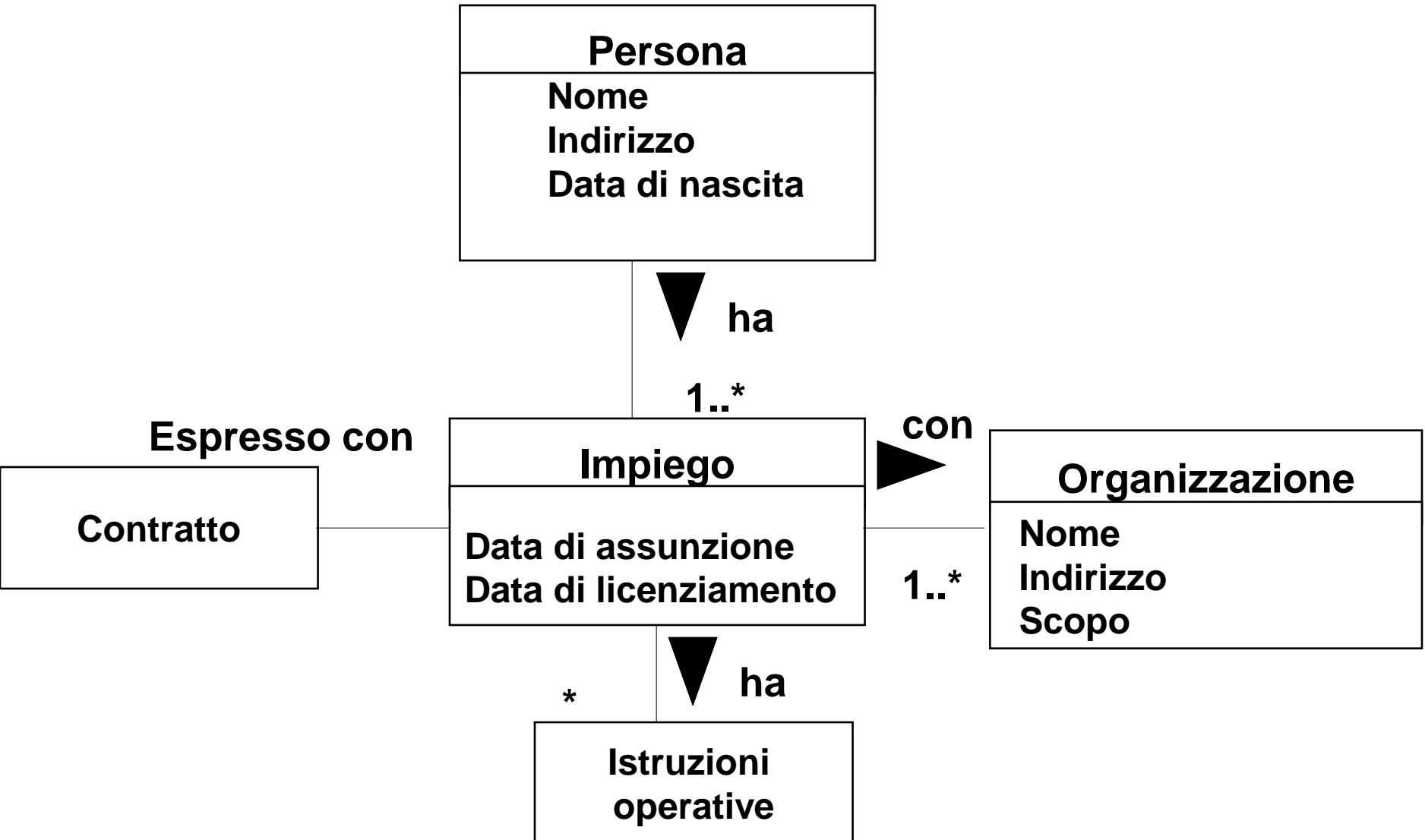
---

- Risorse e ruoli (Resource and Role patterns)
- Obiettivo (Goal patterns)
- Processo (Process patterns)

# La coda: pattern di processo

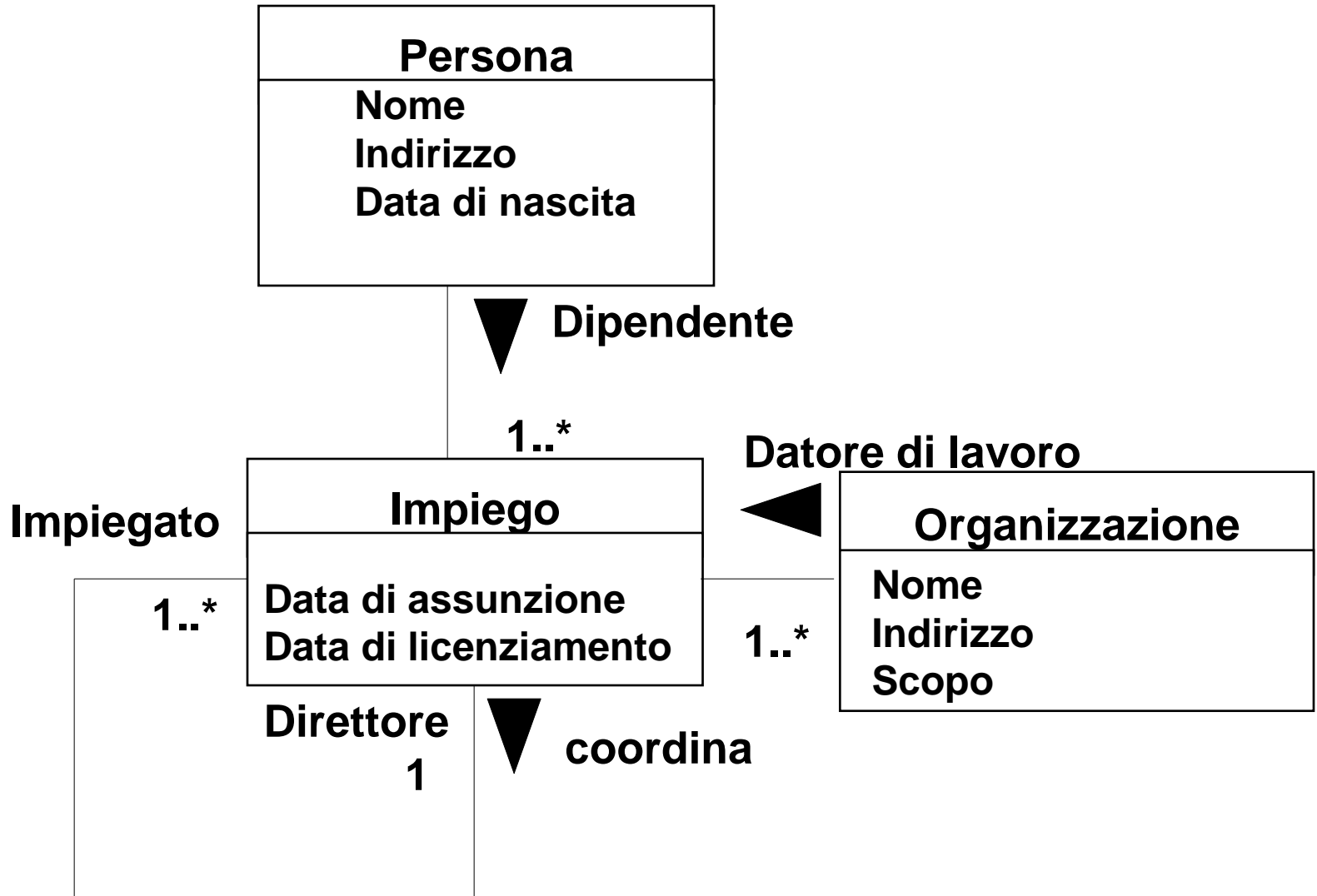


# L'impiego: pattern di ruolo

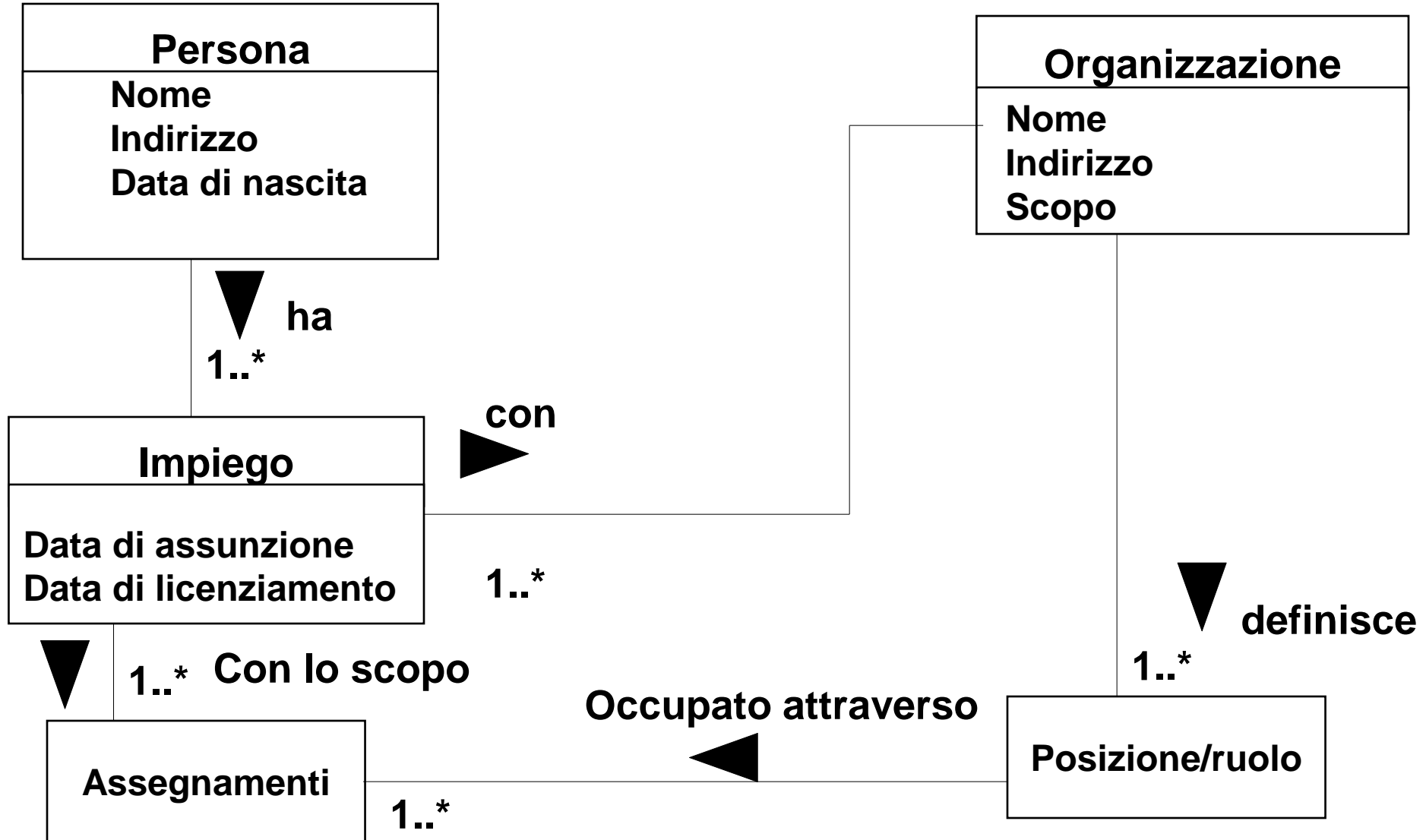




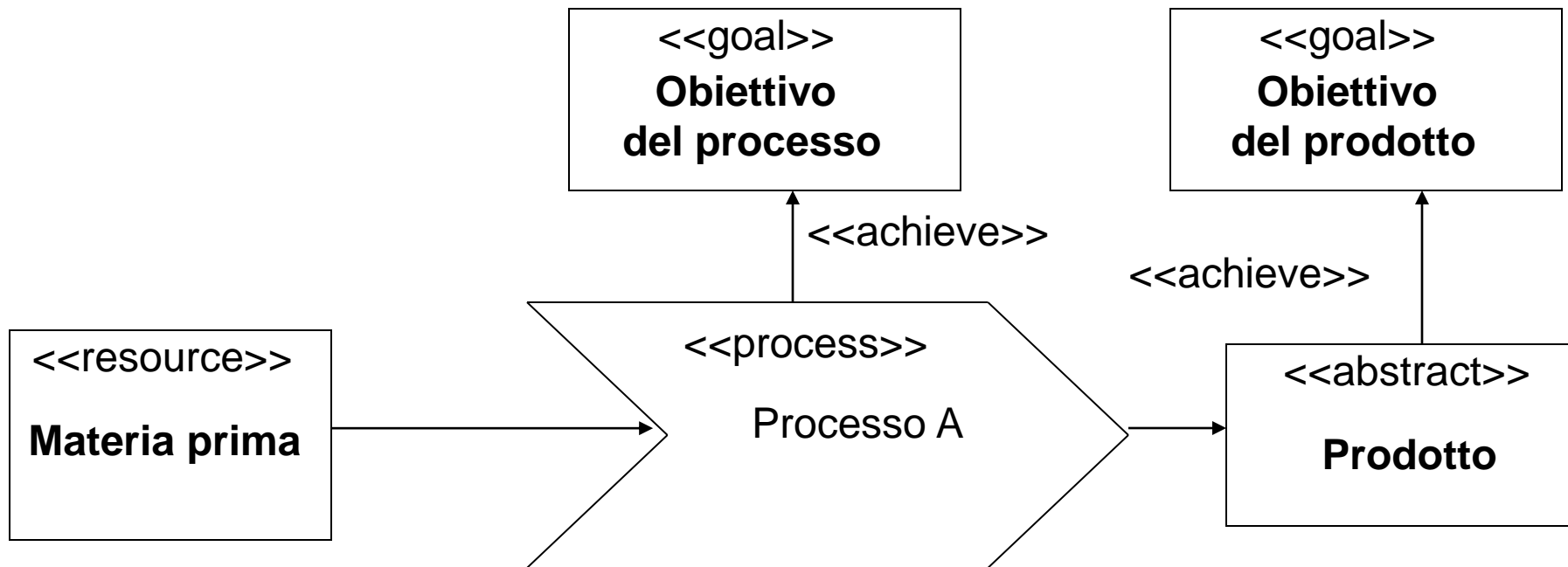
# L'impiego: pattern di ruolo



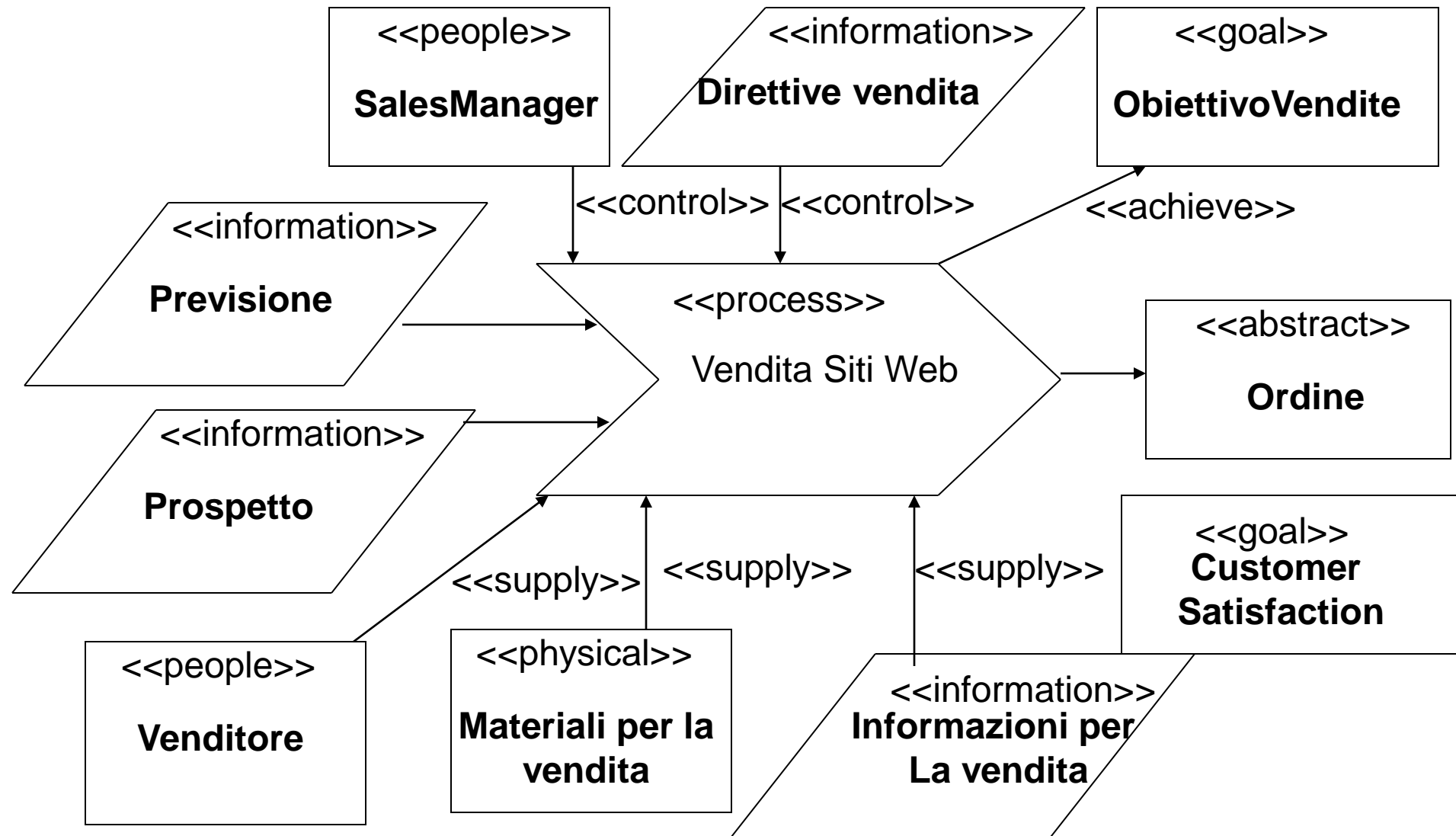
# L'impiego: pattern di ruolo



# Business goal allocation: goal pattern



# I goal nella vendita di siti Web



# Sommario

---

- L'analisi dei processi business
- UML ed il suo ruolo
- Analisi del processo come successione di attività
- Analisi del processo come successione di casi d'uso
- Analisi delle entità che prendono parte ai processi
- Analisi delle interazioni tra gli elementi di un processo
- Un esempio completo di analisi
- Una visione d'insieme: il legame fra le viste e i pattern

# Bibliografia

---

H. Eriksson et M. Penker  
Business Modeling with UML  
Ed. John Wiley & Sons., 2000

A.G. Nillson et al.  
Perspective on Business Modeling  
Ed. Springer-Verlag, 1999

# Bibliografia Web

---

- OMG
- <http://www.omg.org>
- Astrakan
- <http://www.astrakan.com>
- Open Training
- <http://www.opentraining.com>

# Bibliografia Web

---

Documenti reperibili in rete presso

- <http://www.uml.org>
- <http://www.mokabyte.it/umlbook/index.htm>
- <http://www.ebxml.com>
- <http://www.oasis-open.org>
- <http://www.ibm.com>
- <http://www.microsoft.com>
- <http://www.iona.com>