

LEZIONE 18

CLASS DIAGRAM 4

Associazioni, Aggregazioni, Composizioni e Dipendenze

Ingegneria del Software e Progettazione Web
Università degli Studi di Tor Vergata - Roma

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

class diagram

- struttura statica del sistema:
 - **nodi** + relazioni

class diagram

- struttura statica del sistema:
 - **nodi** + relazioni
- un nodo modella una **classe** (o una interfaccia) che rappresenta:
 - una entità del dominio
 - elementi che non fanno parte del dominio ma utili nell'ingegnerizzazione del sistema

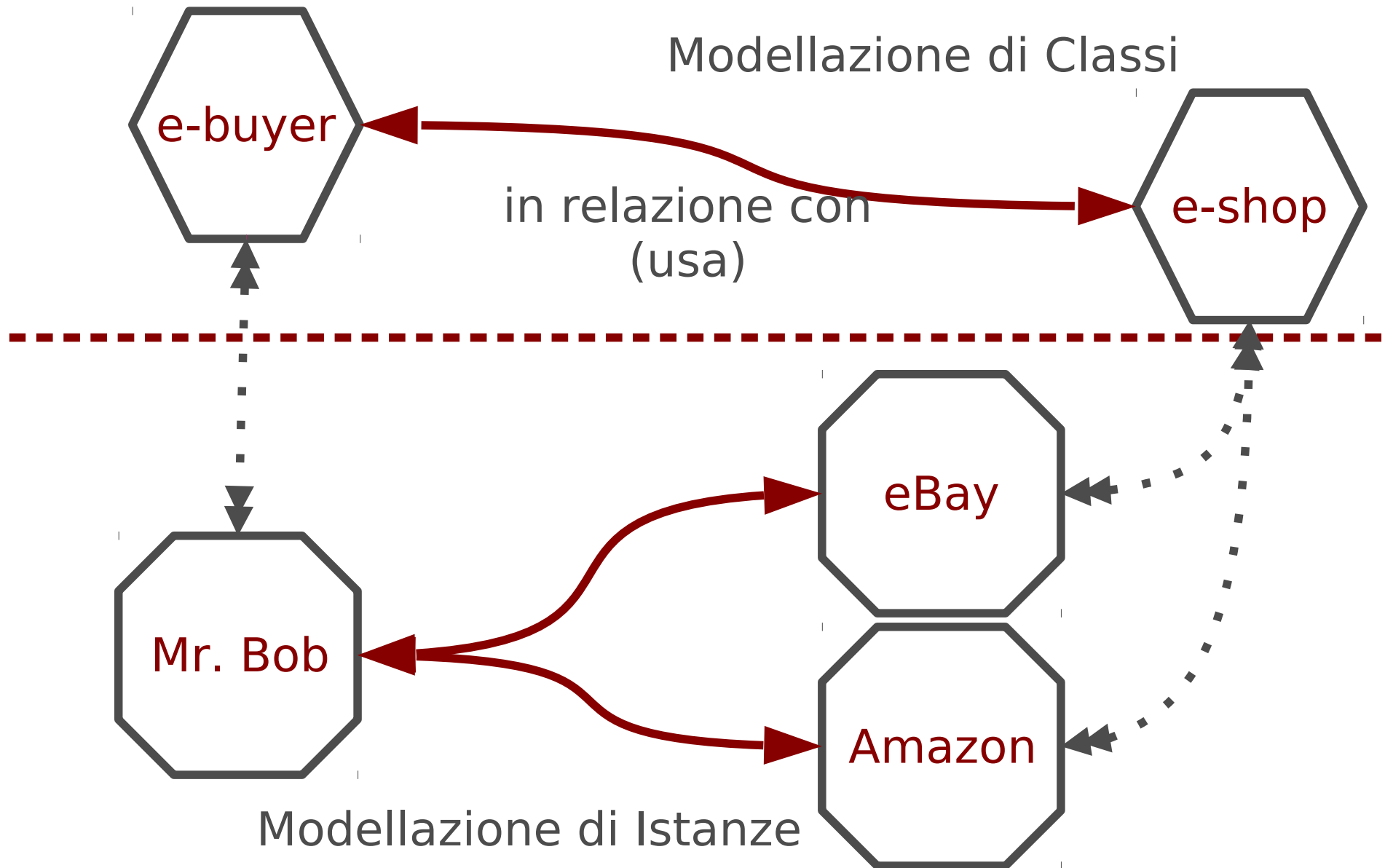
le relazioni in un class diagram

- una relazione rappresenta una “interazione” tra gli elementi modellati
 - una *relazione* tra una classe A ed una classe B significa che A è a conoscenza di B e (in qualche *modo*) può interagirvi
- il tipo di una relazione definisce il “modo di interazione” tra gli elementi che essa coinvolge
- nei class diagram si specificano relazioni tra :
 - classi, oggetti, interfacce, package, etc...

ricordiamo che:


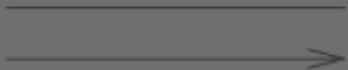

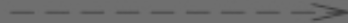

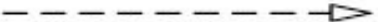
- **operazione** : specifica di un “*segnatura*” (prototipo) che un oggetto mette a disposizione di altri oggetti
- **metodo** : implementazione vera e propria dell'operazione di un oggetto
- **messaggio** : è la richiesta a run-time di invocazione di un metodo fornito da un oggetto \mathcal{B} da parte di un oggetto \mathcal{A}

classi **VS.** oggetti







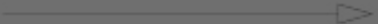

le relazioni in un class diagram

Table 7.2 Graphic paths included in structure diagrams

<i>PATH TYPE</i>	<i>NOTATION</i>	<i>REFERENCE</i>
Aggregation		See “AggregationKind (from Kernel)” on page 38.
Association		See “Association (from Kernel)” on page 39.
Composition		See “AggregationKind (from Kernel)” on page 38.
Dependency		See “Dependency (from Dependencies)” on page 62.
Generalization		See “Generalization (from Kernel, PowerTypes)” on page 71.
InterfaceRealization		See “InterfaceRealization (from Interfaces)” on page 89.

le relazioni in un class diagram

Table 7.3 - Graphic paths included in structure diagrams

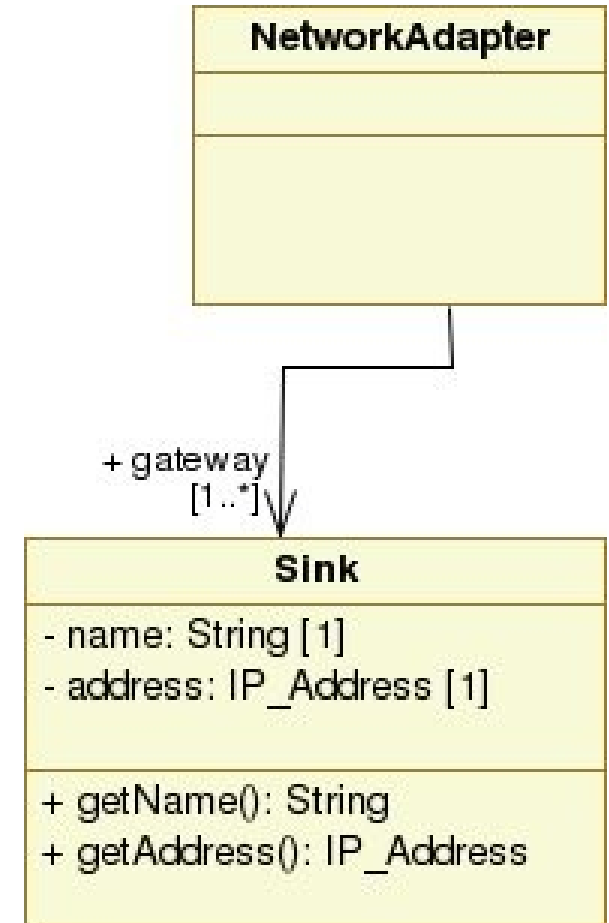
<i>PATH TYPE</i>	<i>NOTATION</i>	<i>REFERENCE</i>
Aggregation		See "AggregationKind (from Kernel)" on page 38.
Association		See "Association (from Kernel)" on page 39.
Composition		See "AggregationKind (from Kernel)" on page 38.
Dependency		See "Dependency (from Dependencies)" on page 62.
Generalization		See "Generalization (from Kernel, PowerTypes)" on page 71.
InterfaceRealization		See "InterfaceRealization (from Interfaces)" on page 89.

associazione

- relazione strutturale tra oggetti di classi differenti
 - può essere simmetrica (navigabile nelle due direzioni)
- una associazione può essere riflessiva, ovvero tra oggetti della stessa classe
- binaria o N-aria (è poco usata)
- rappresentata mediante una linea continua che collega le classi in relazione
- di fatto : indica la possibilità che una classe possa inviare messaggi a quelle associate

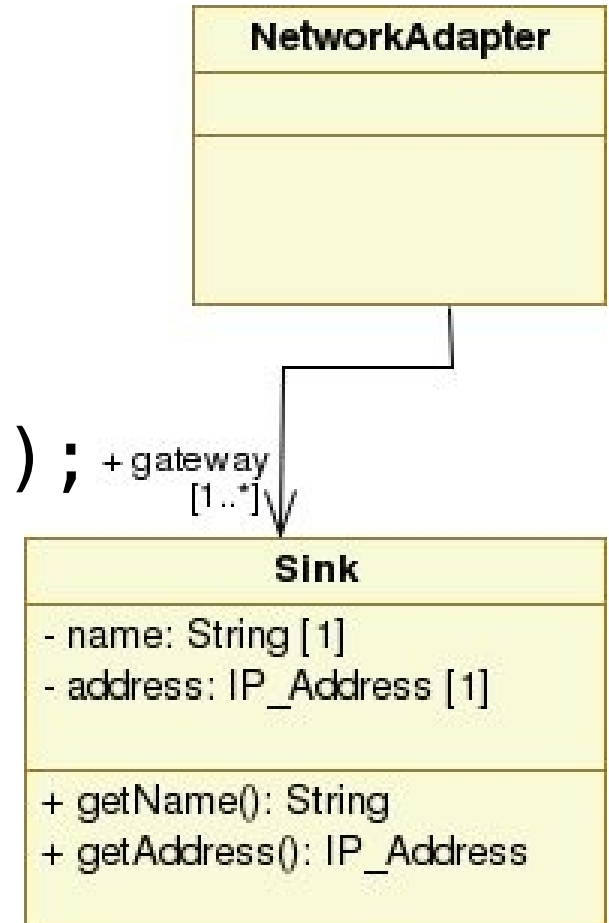
associazione – map in Java

- un associazione può avere:
 - un nome,
 - il ruolo degli operandi,
 - stereotipo
 - cardinalità



associazione – map in Java

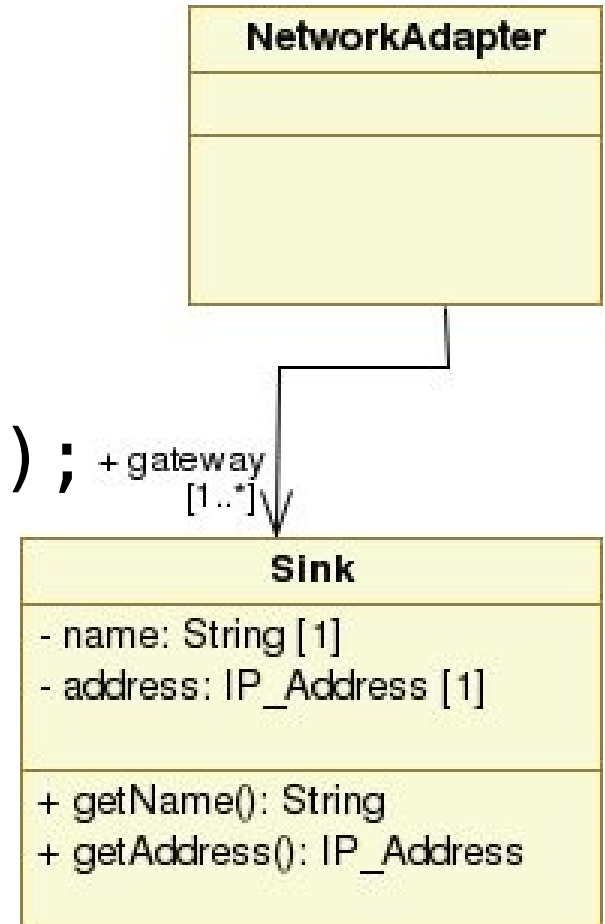
```
public class Sink {  
    private String name;  
    private IP_Address address;  
    public String getName();  
    public IP_Address getAddress();  
}
```



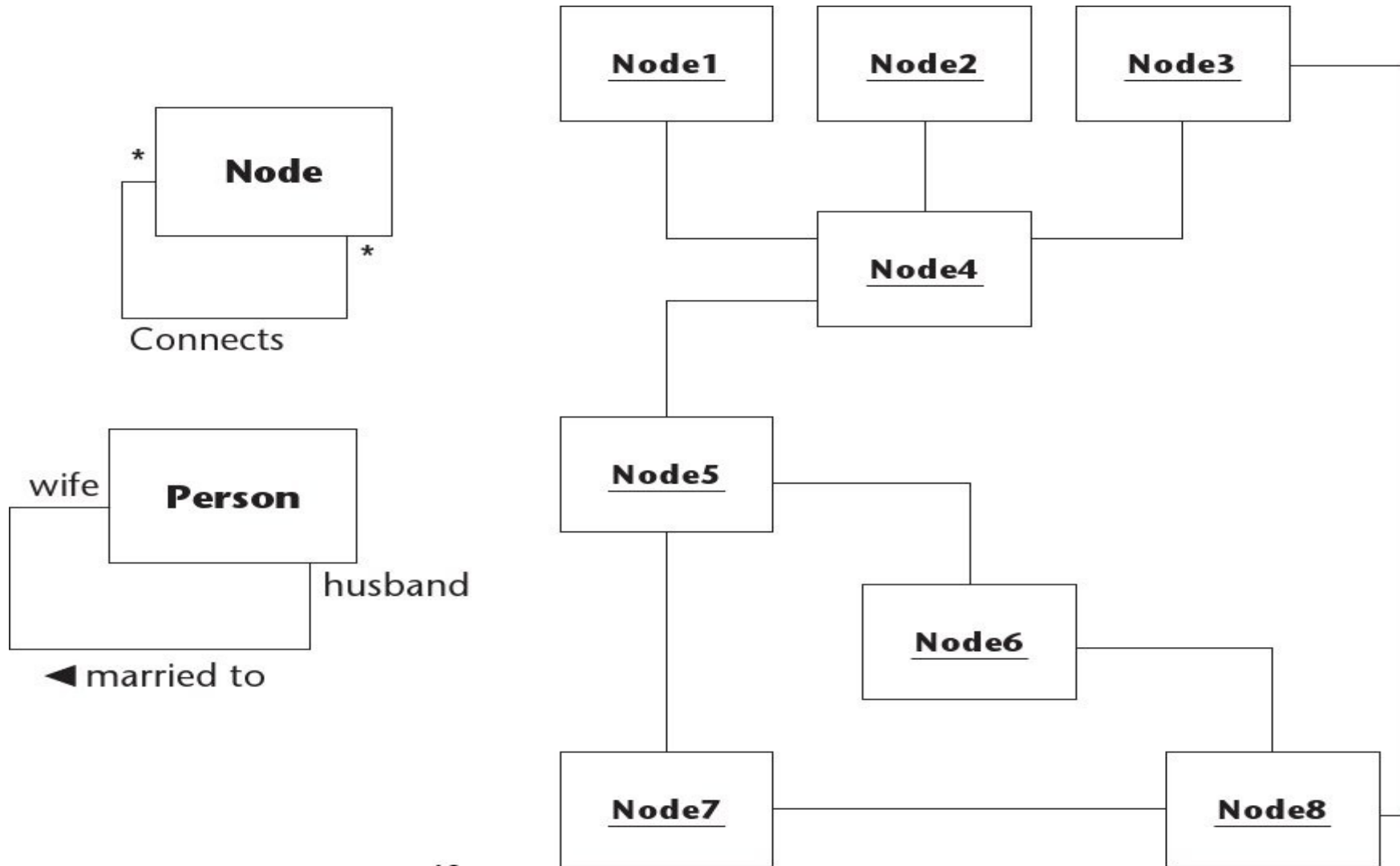
associazione – map in Java

```
public class Sink {  
    private String name;  
    private IP_Address address;  
    public String getName();  
    public IP_Address getAddress();  
}
```

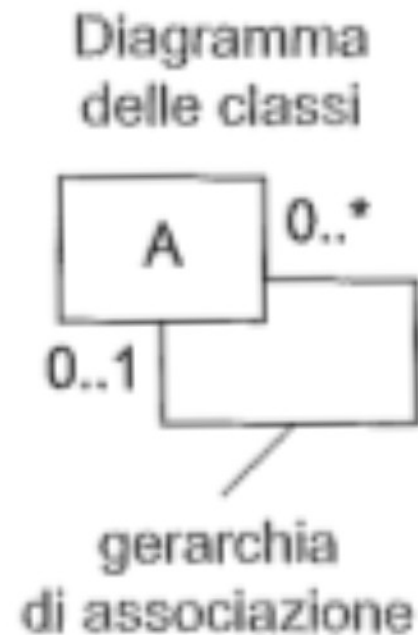
```
public class NetworkAdapter {  
    private Vector<Sink> gateway;  
}
```



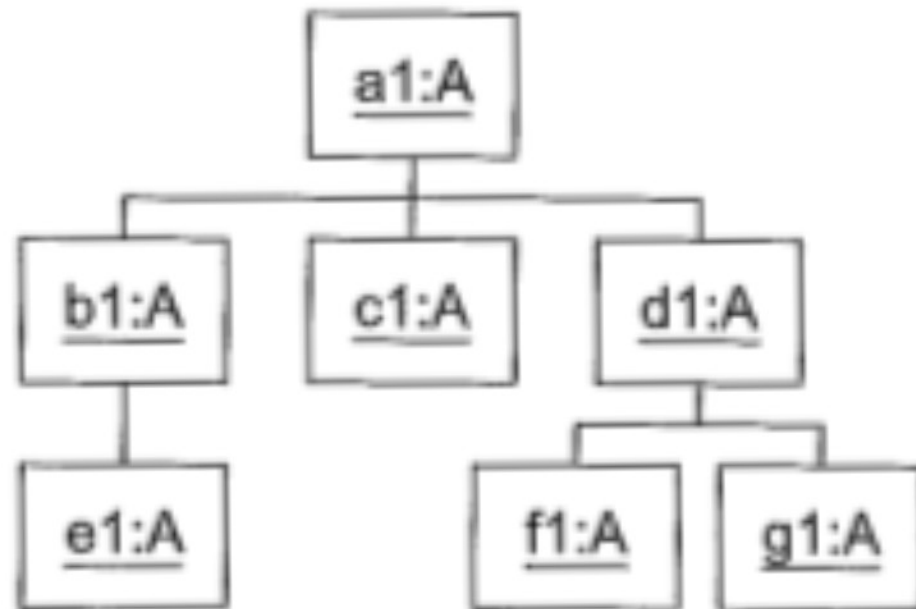
associazione - esempi vari



associazione : classi && oggetti – 1



Esempio di diagramma degli oggetti

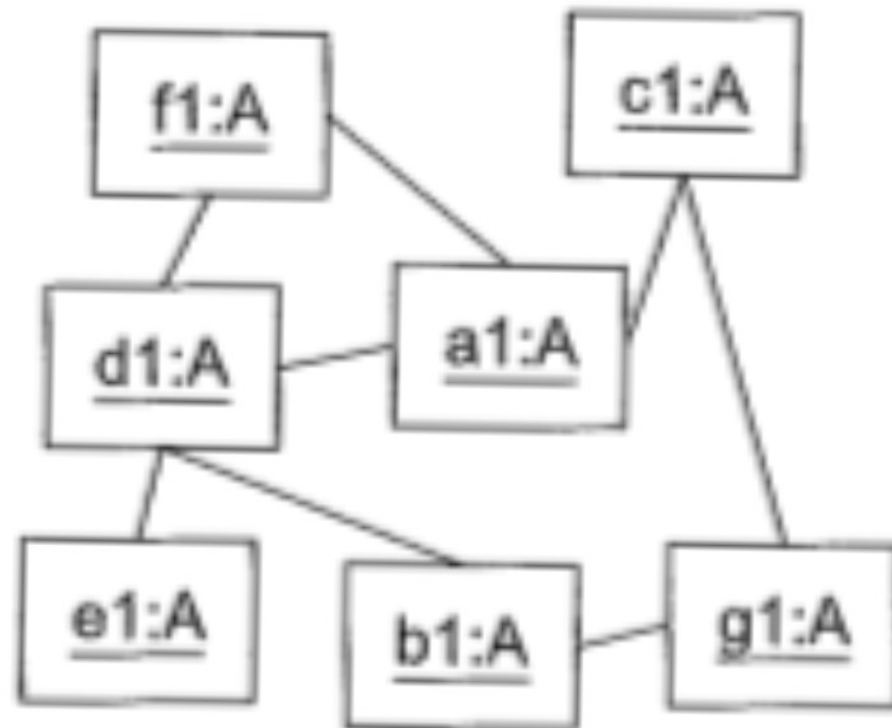


associazione : classi && oggetti – 2

Diagramma
delle classi



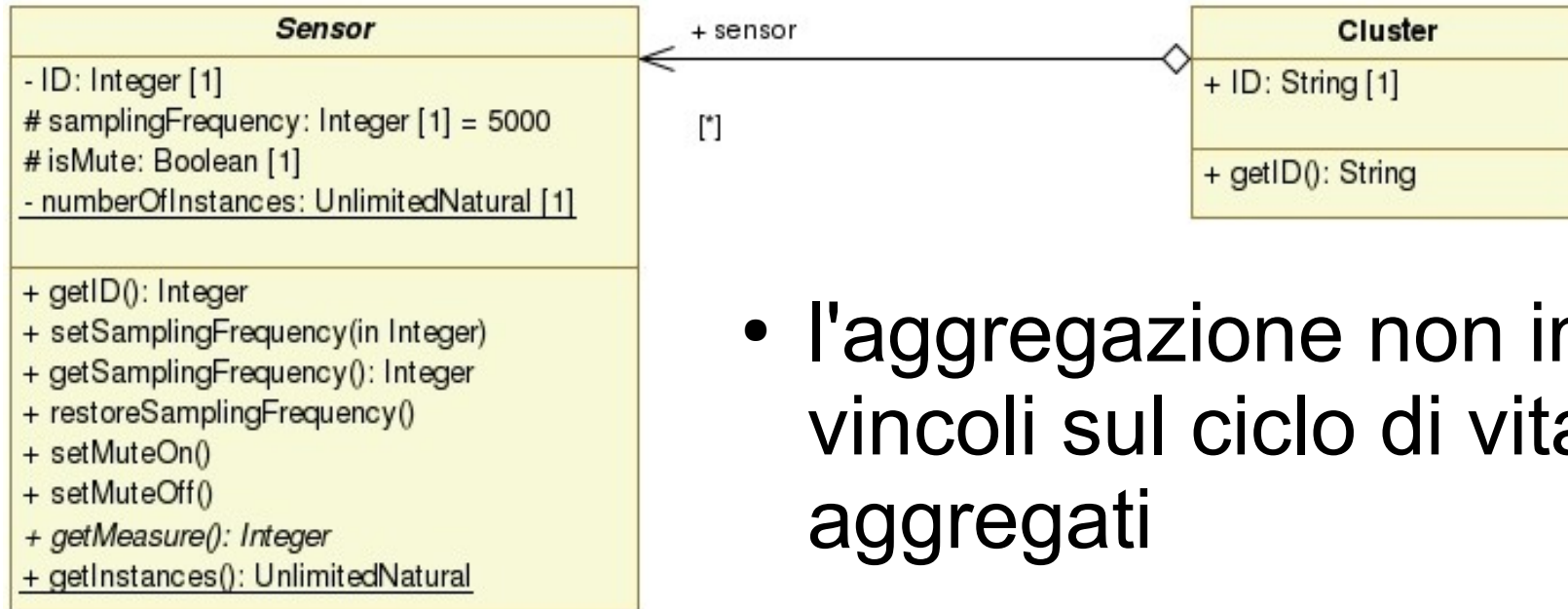
Esempio di diagramma degli oggetti



aggregazione

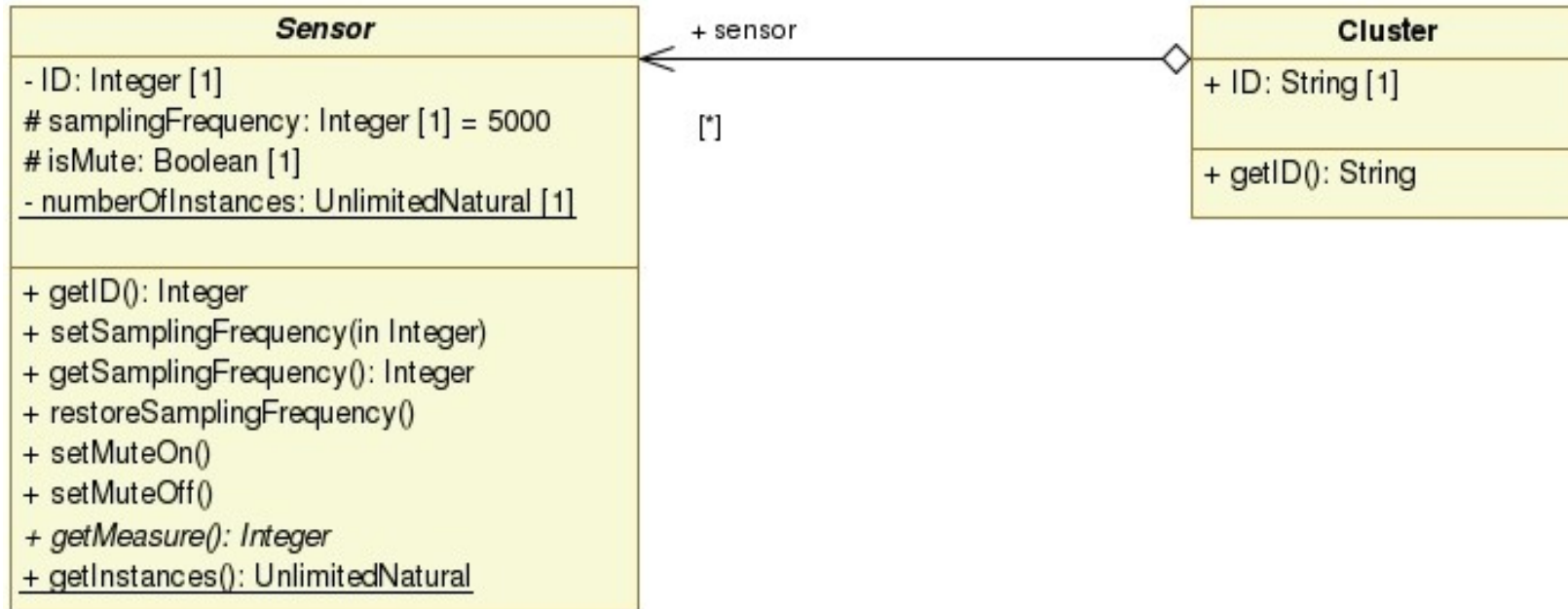
- è una relazione di tipo gerarchico :
 - una classe che rappresenta “entità autonoma”
 - una classe che rappresenta che aggrega l' “entità autonoma”
- esempio:
 - auto = motore + 4 ruote + scocca + ... *altre cose*
 - Paniere Titoli = \sum Titolo_Azionario_i (class view)
 - FTSE MIB (Borsa Milano) = Enel + ENI + Telecom Italia + ... *altri titoli (instance view)*
- denominata anche : relazione “*whole-part*”
- rappresentata mediante una linea tra la classe aggregante e quella aggregata. all'estremità della classe aggregante, la linea ha un rombo bianco

aggregazione



- l'aggregazione non impone vincoli sul ciclo di vita degli aggregati
- le aggregazioni circolari sono **semanticamente errate**
 - A aggrega B
 - B aggrega C
 - C aggrega A

aggregazione – map in Java



```
public class Cluster {  
    private Vector<Sensor> vSensor;  
    public Cluster (Vector<Sensor> s) {  
        this.vSensor= s;  
    }  
}
```

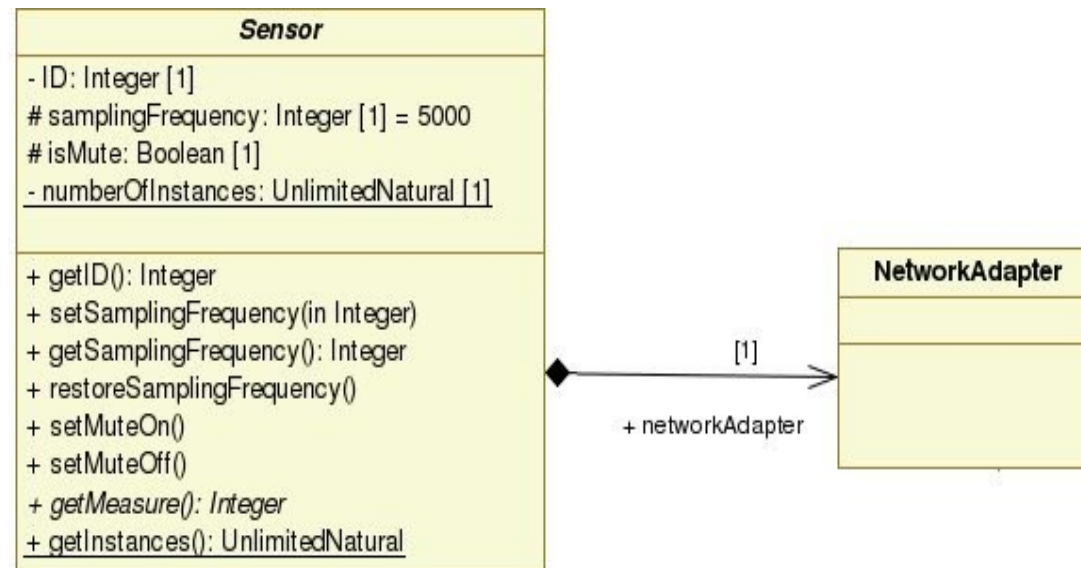
composizione

- è un'aggregazione forte
 - le istanze composte non devono necessariamente essere create con l'istanza della classe "componente"
 - ... ma una volta composte, le istanze delle componenti seguono il ciclo di vita dell'istanza che le compone
- esempio:
 - carbonara = spaghetti + uova + pancetta + ... altre cose
 - forum = \sum thread_di_discussione_i
 - thread_di_discussione = \sum commenti_i
- dipendentemente dall'ambiente di sviluppo:
 - la classe composita elimina le parti in un momento antecedente alla propria distruzione (dispose in C++)
 - l'implementazione è strutturata in modo che le istanze create con la composizione siano distrutte (con il garbage collector in Java)

composizione

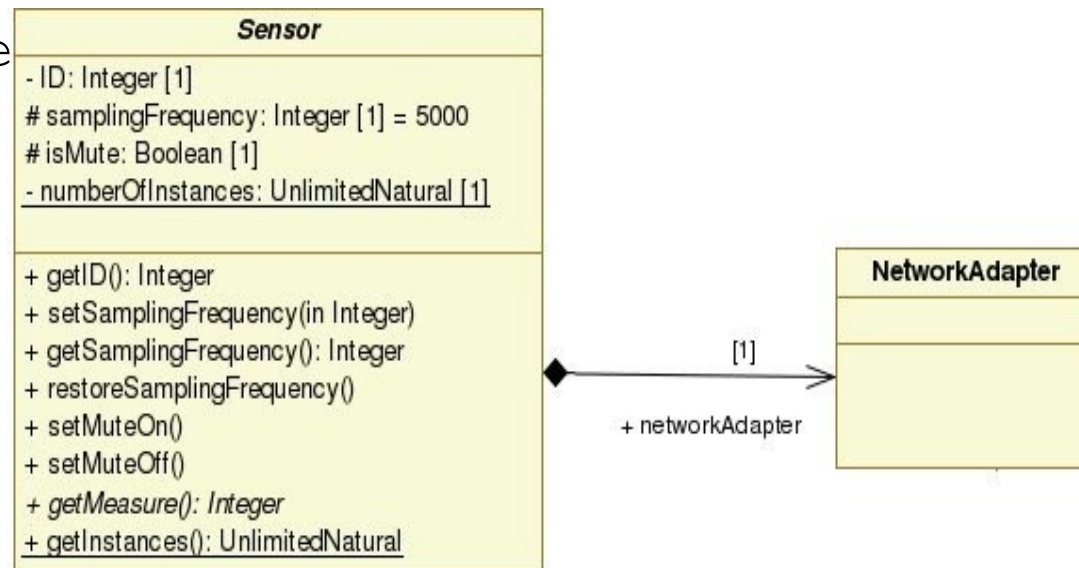
- conseguenze :
 - un oggetto può essere parte di un solo oggetto composto alla volta
 - invece nell'aggregazione una parte può essere condivisa tra più oggetti composti
- in UML è rappresentata mediante una linea tra la classe che compone (whole) e quella componente (part). all'estremità della classe che compone, la linea ha un rombo nero

composizione – map in Java – 1



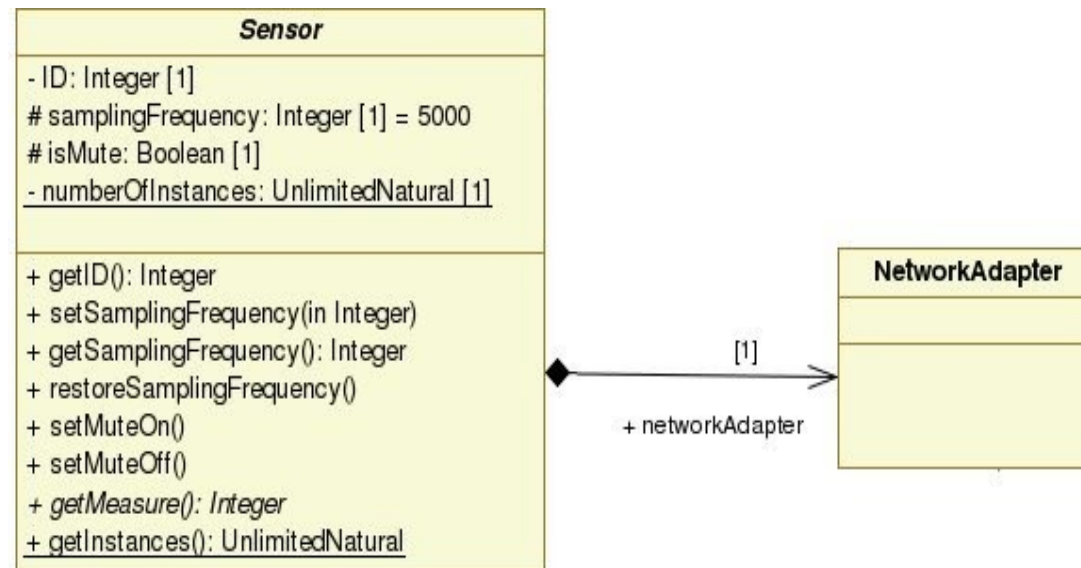
composizione – map in Java – 2

```
public abstract class Sensor {  
    private NetworkAdapter na;  
    public Sensor () {  
        this.na = new NetworkAdapter();  
        ...  
    }  
    ...  
    /* Never pass  
    * the reference  
    * to this.na  
    * out of this  
    * class  
    */  
}
```



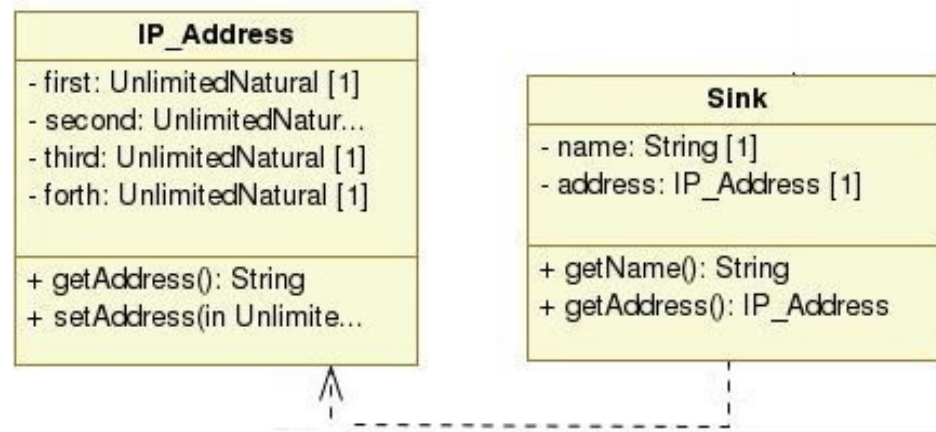
composizione – map in Java – 3

```
public abstract class Sensor {  
    private NetworkAdapter na;  
  
    public Sensor (NetworkAdapter net) {  
        this.na = new NetworkAdapter();  
        /* Clone the state of net  
         * into this.na  
         */  
    }  
}
```



dipendenza

- è una relazione tra due elementi dove il cambiamento di un elemento può causare il cambiamento nell'altro
 - non necessariamente è implicato il viceversa
- rappresentata mediante una linea tratteggiata che collega le classi in relazione



esempio – una semplice sensor network

una sensor network è una rete costituita da un insieme di piccoli apparecchi elettronici in grado di ricevere dati dall'ambiente circostante e comunicare tra loro mediante degli elementi per l'interfacciamento alla rete. ogni sensore invia i dati collezionati ad un gateway. I sensori possono essere organizzati in aree di sensing o cluster.

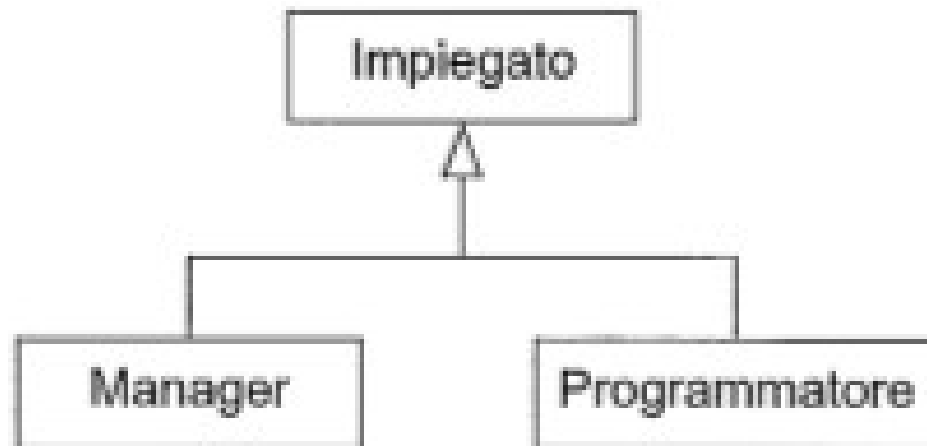
relazioni : quali? come?

relazioni : quali? come?

- problema : modellare i ruoli organizzativi in una azienda

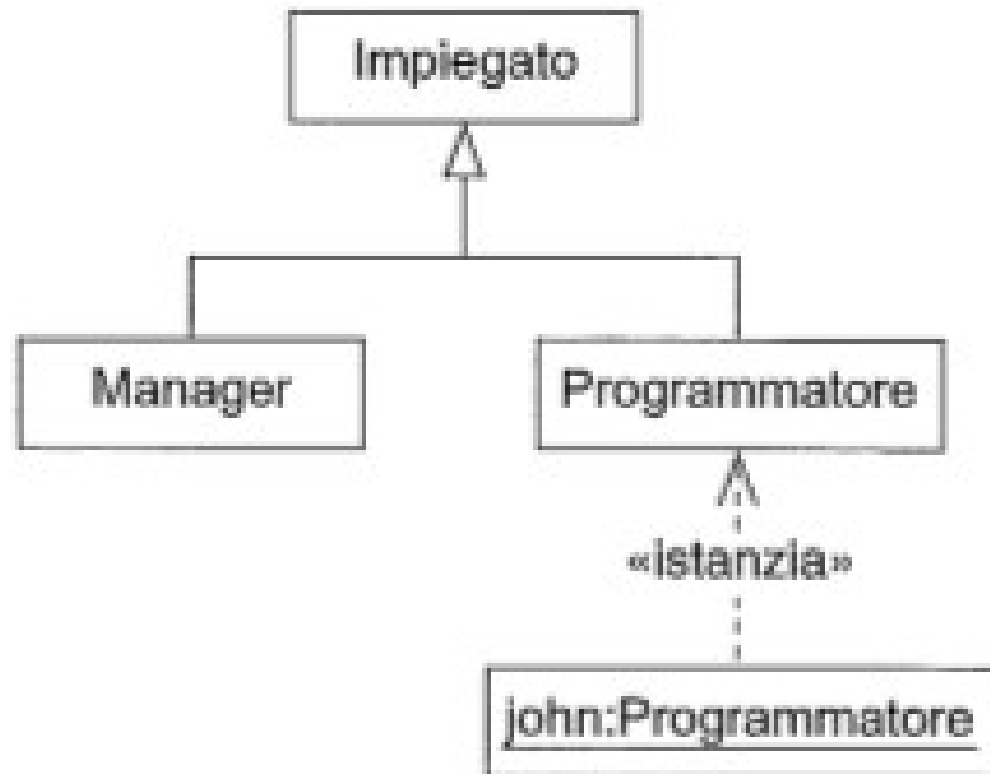
relazioni : quali? come?

- problema : modellare i ruoli organizzativi in una azienda
- (possibile) soluzione :



relazioni : quali? come?

- problema : modellare i ruoli organizzativi in una azienda
- (possibile) soluzione :



relazioni : quali? come?

- problema : modellare i ruoli organizzativi in una azienda

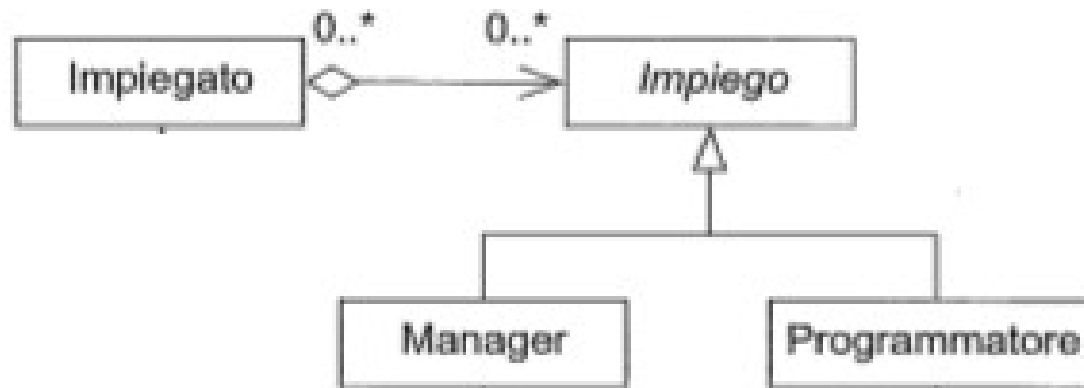
- (possibilità)

**modello rappresentativo, ma
... come gestiamo a runtime
il fatto che John possa
essere "promosso" a
Manager
(Pattern della Metamorfosi)**



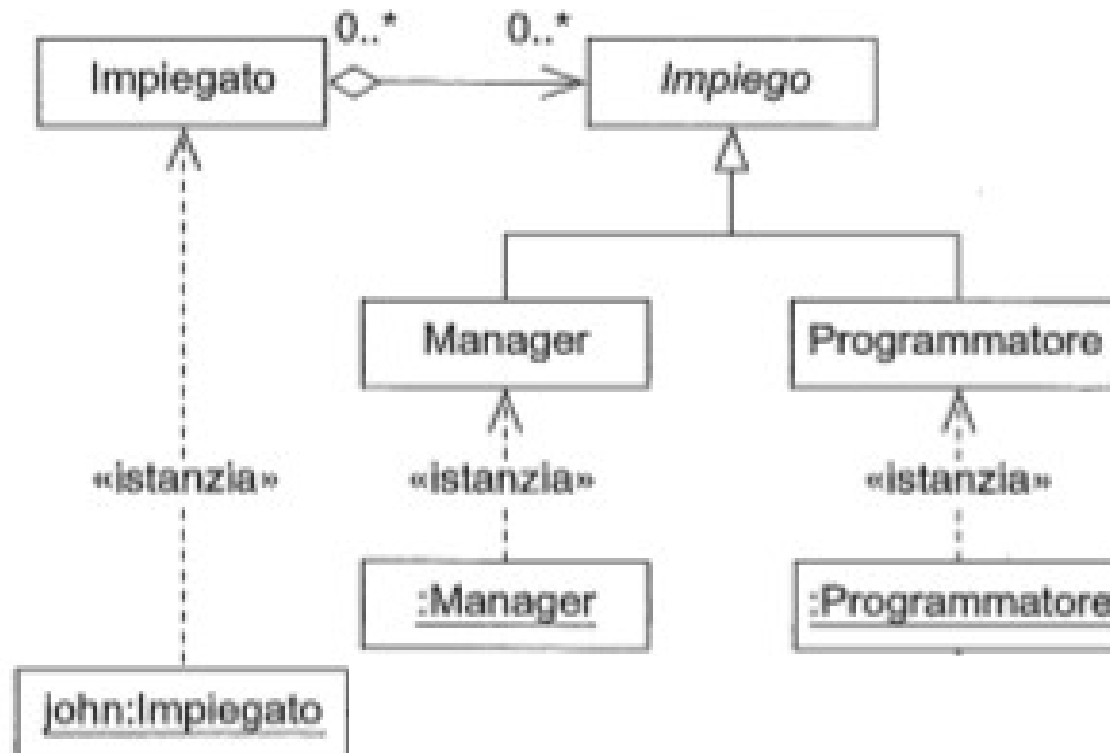
relazioni : quali? come?

- problema : modellare i ruoli organizzativi in una azienda
- (possibile) **raffinamento** :



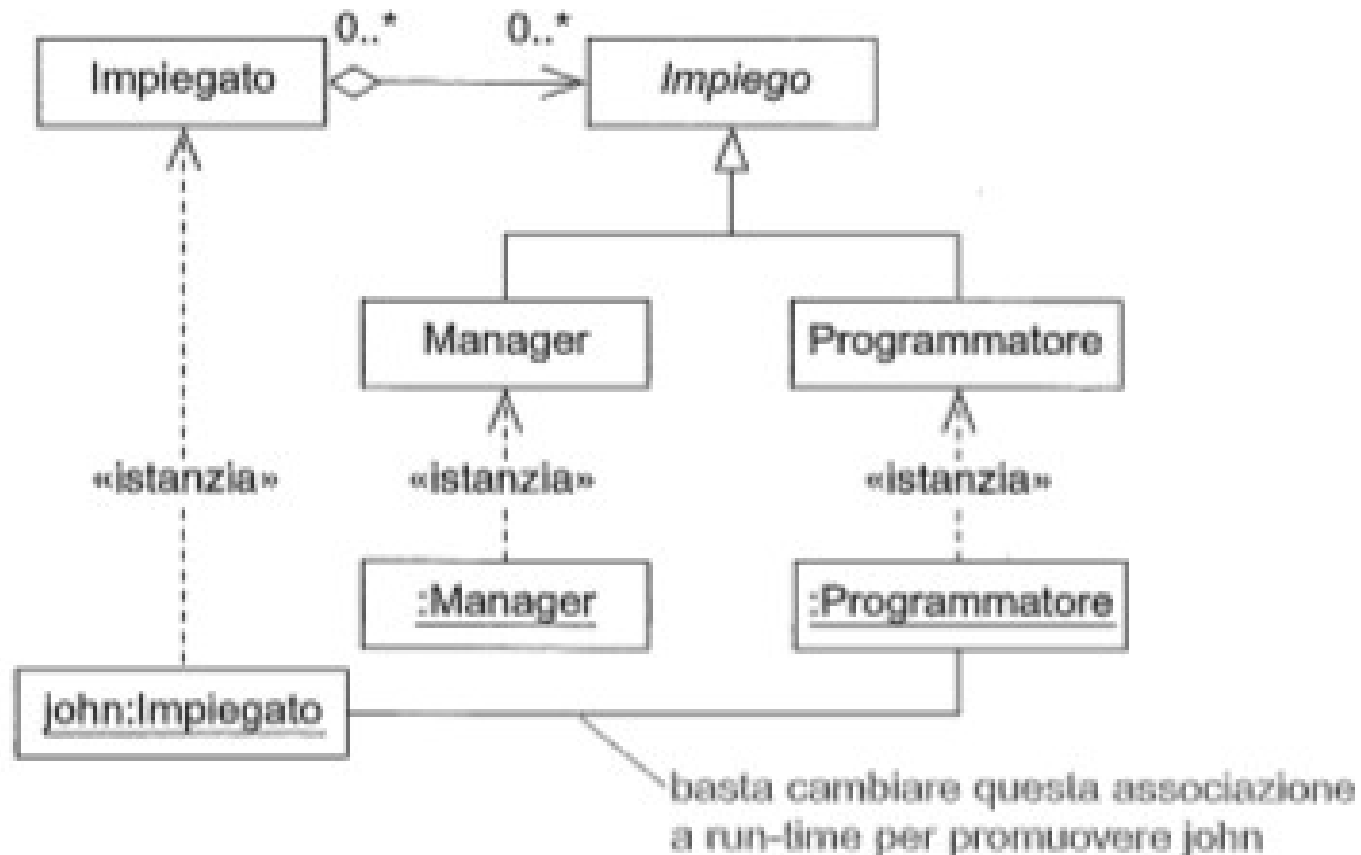
relazioni : quali? come?

- problema : modellare i ruoli organizzativi in una azienda
- (possibile) raffinamento :

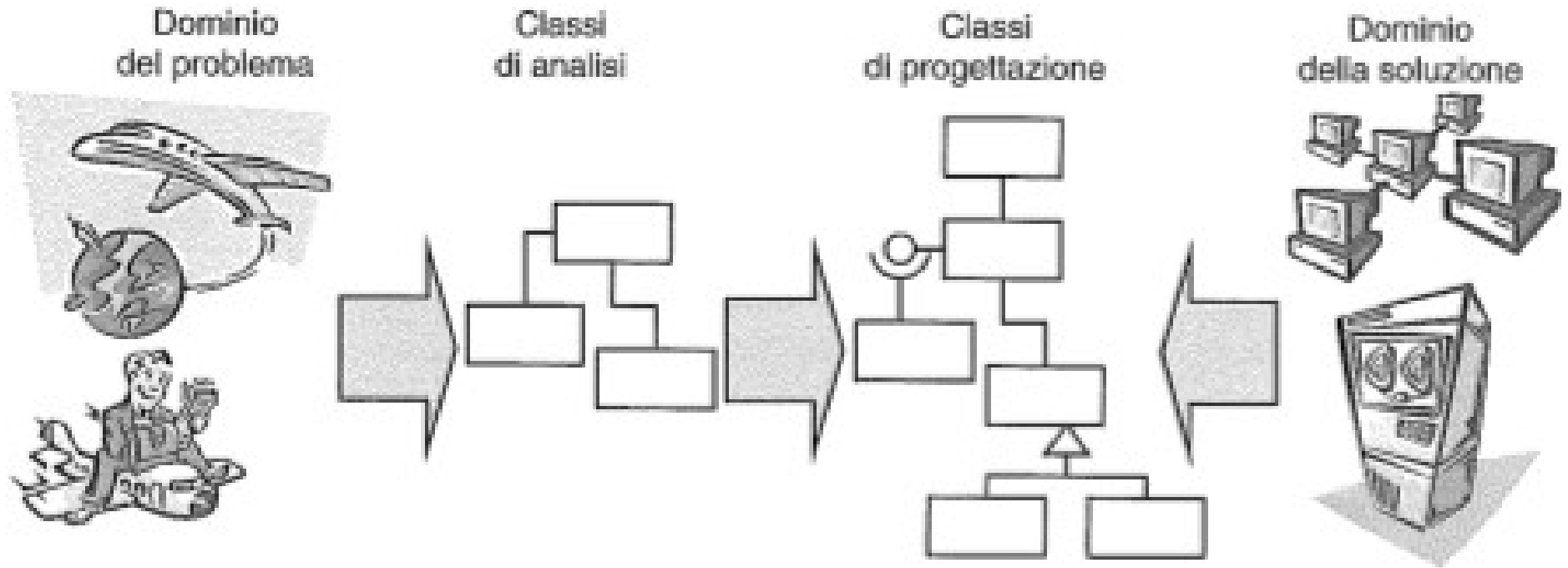


relazioni : quali? come?

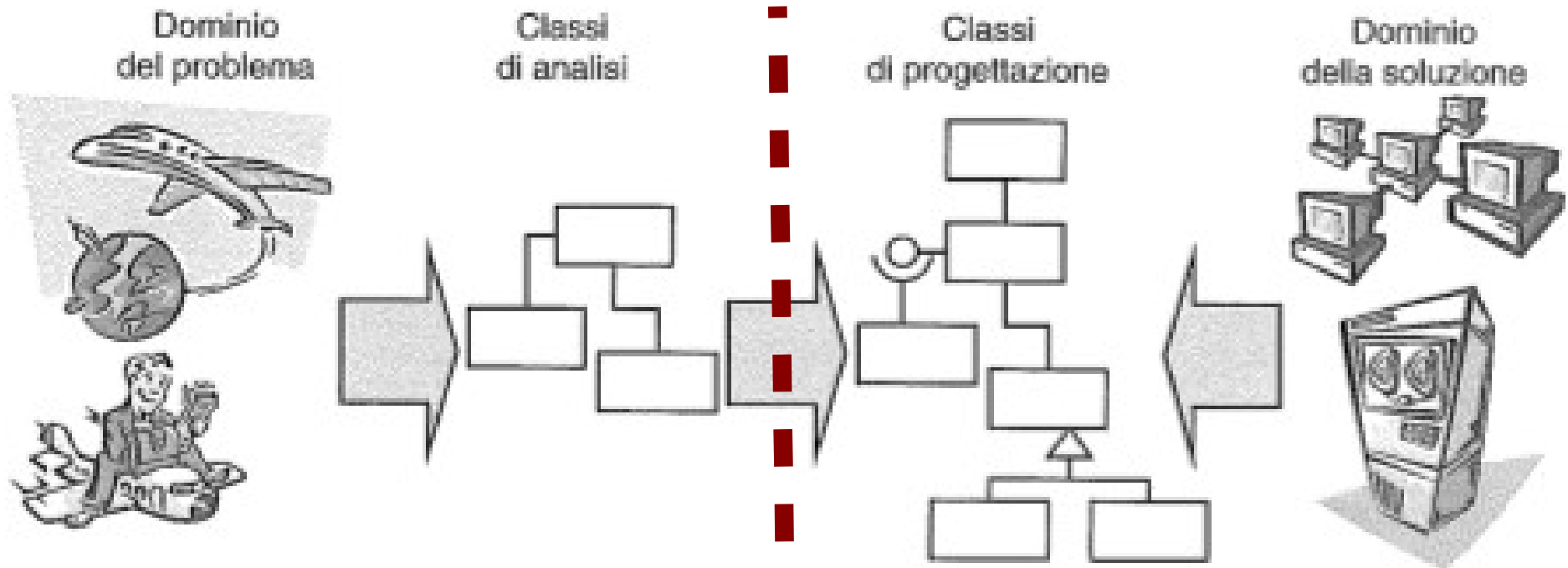
- problema : modellare i ruoli organizzativi in una azienda
- (possibile) raffinamento :



contesto generale



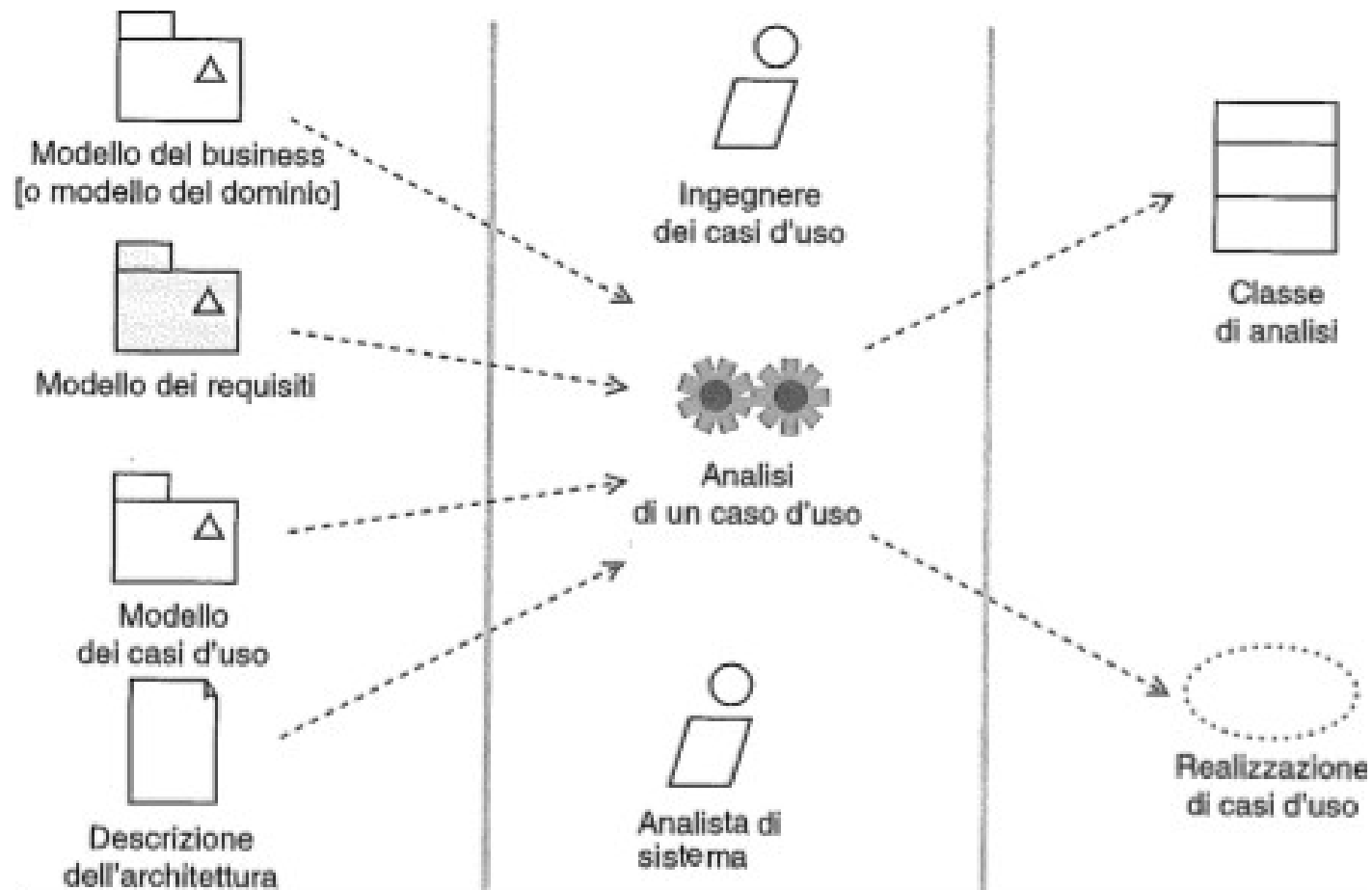
contesto generale



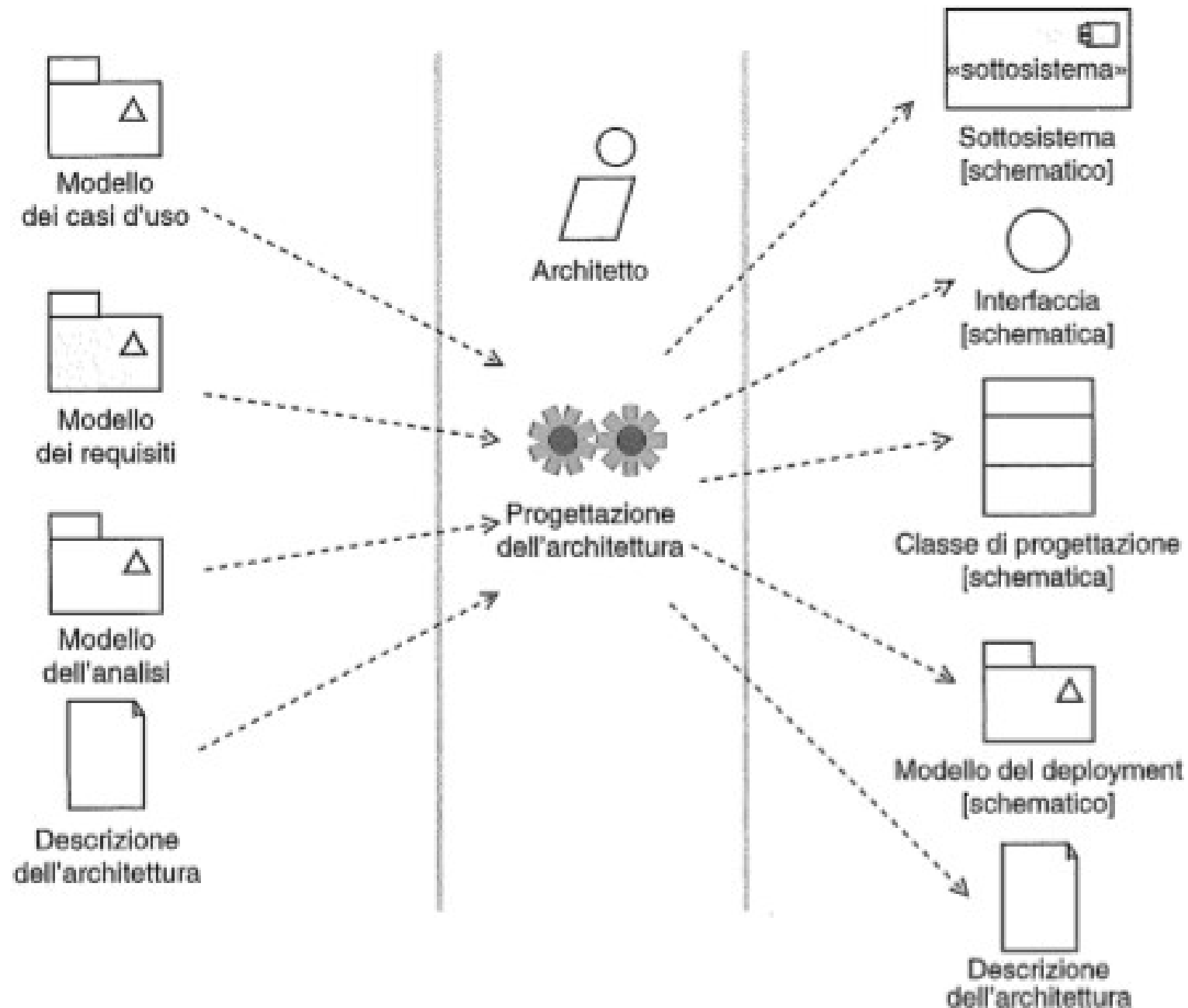
- obiettivo dell'analisi è capire il problema e sviluppare modelli (visuali) che descrivono ciò che si vuole ottenere indipendentemente dalle tecnologie e da aspetti implementativi
- l'analisi si focalizza sul raffinamento dei requisiti (funzionali) in concetti rappresentabili come elementi software

- obiettivo delle fasi di progettazione è raffinare i modelli di analisi con l'intento di includere dettagli e scelte che più si prestino alle tecnologie di sviluppo ed ai contesti di deployment scelti
- I modelli di progettazione raffinano aspetti tecnologici rispetto ai modelli di analisi, ma non si focalizzano su dettagli implementativi, che sono invece delegati alle fasi più orientate alla scrittura del codice

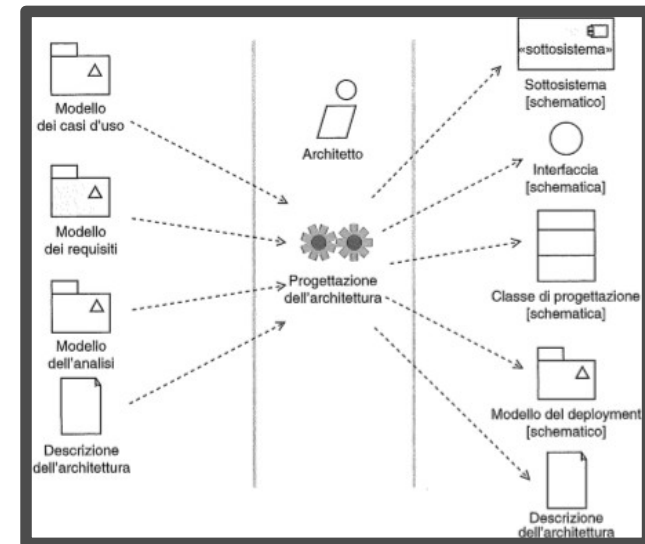
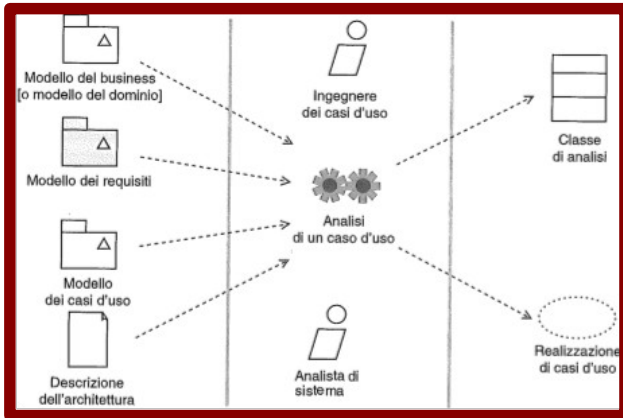
... nello specifico – 1



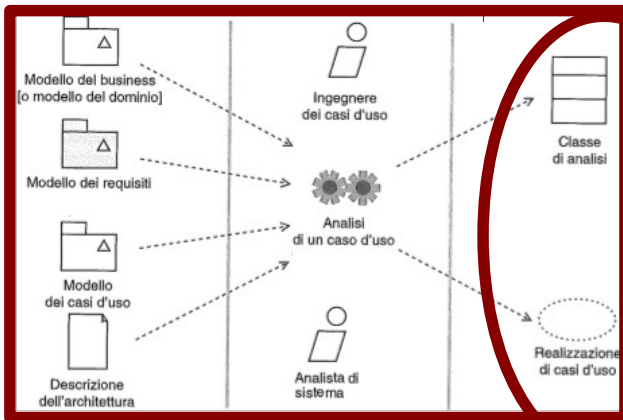
... nello specifico – 2



... nello specifico – 3



... nello specifico – 3



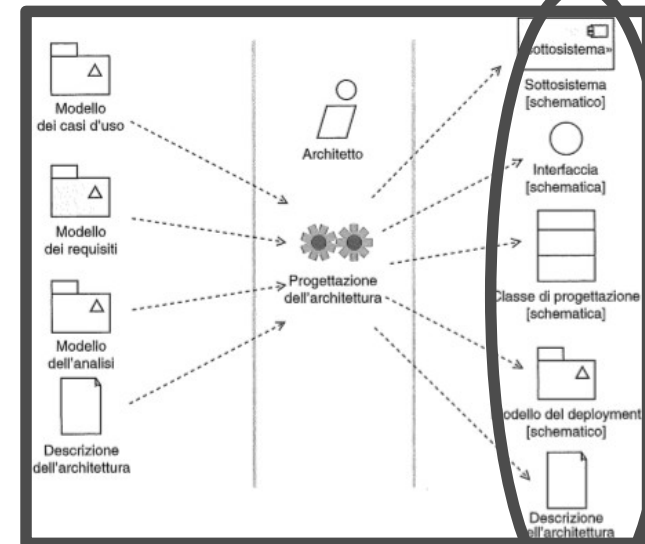
modello
concettuale



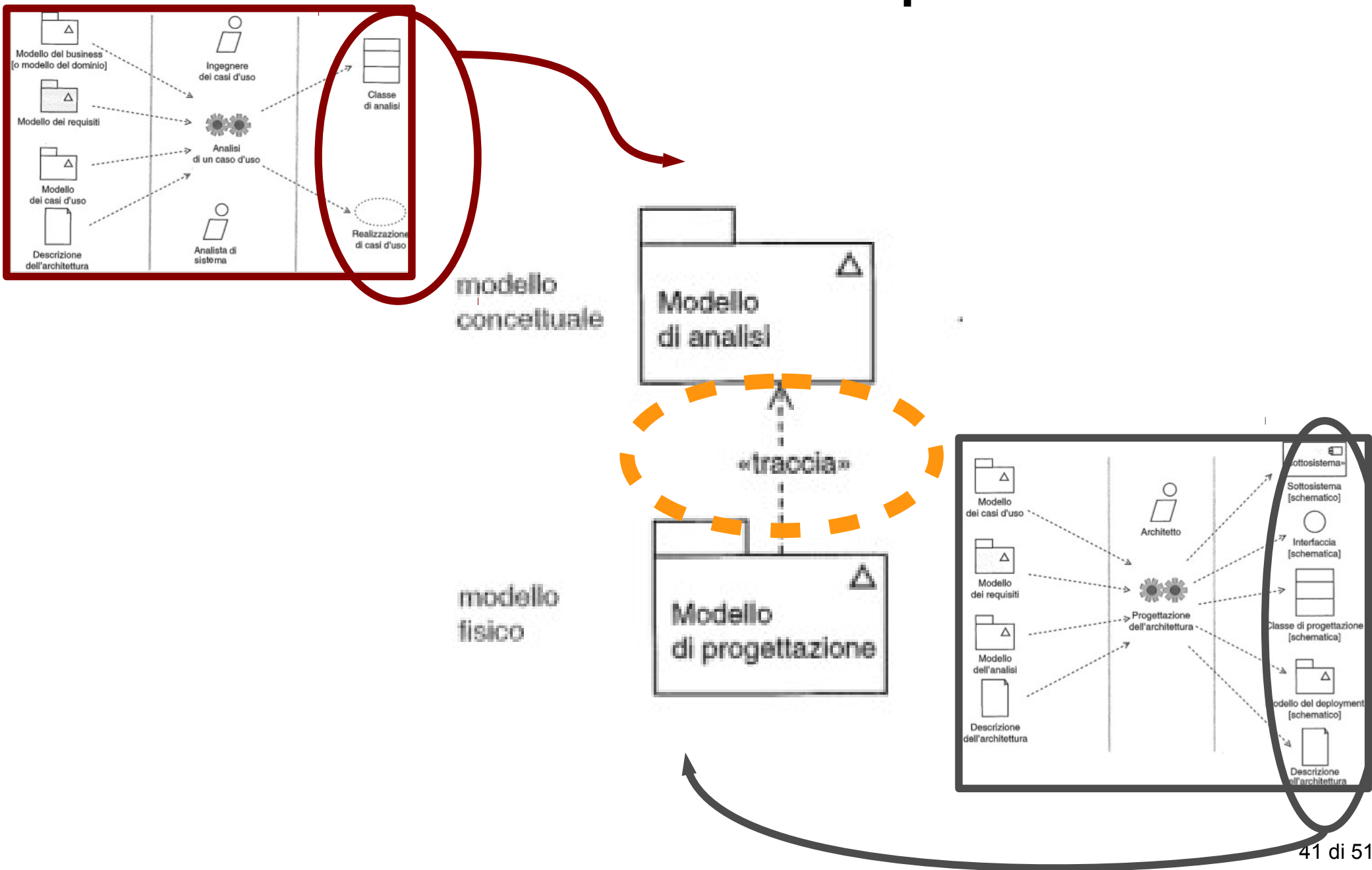
«traccia»



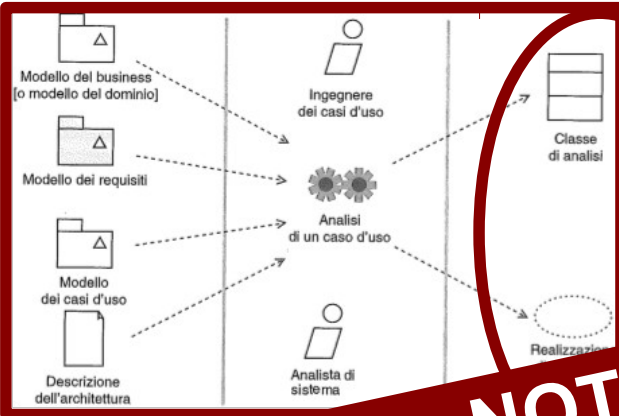
modello
fisico



... nello specifico – 3



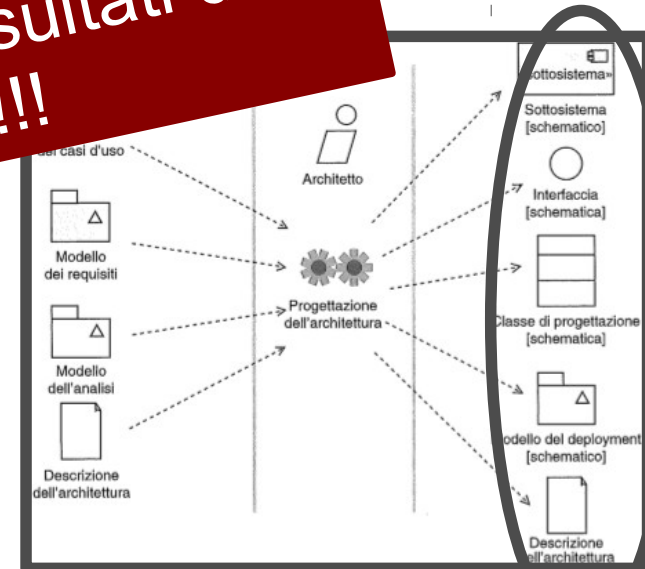
... nello specifico – 3



NOTA : nei processi di sviluppo di ITERATIVI, le fasi di analisi e di progettazione si avvicendano in modo ciclico affrontando gli stessi problemi a diversi livelli di dettaglio, e raffinando le soluzioni proposte in base ai risultati delle iterazioni precedenti!!!

Il ciclo

Modello di progettazione



ancora su ... relazioni : come, quando?

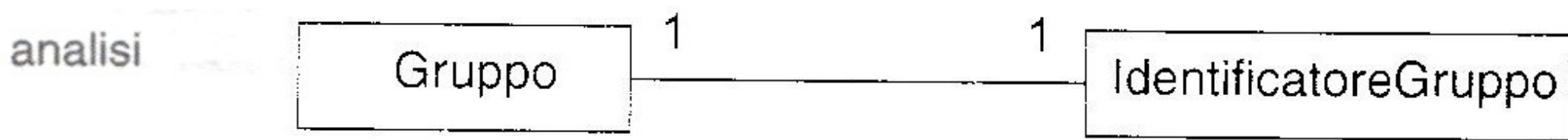
- (normalmente) la fase di analisi enfatizza l'uso di
 - generalizzazioni
 - associazioni
- (normalmente) la fase di progettazione enfatizza il raffinamento di
 - generalizzazioni → realizzazioni
 - classi → interfacce
 - associazioni → aggregazioni
 - associazioni → composizioni
 - associazioni → dipendenze

raffinare associazioni di analisi – 1

- se non presenti, specificare molteplicità e ruoli
- verificare se esistono i presupposti per l'applicazione del “pattern” *whole-part*
 - identificare l'estremo *whole*
 - identificare l'estremo *part*
- controllare la molteplicità per *whole* :
 - **se 0..1 o 1** → è possibile (non obbligatorio) raffinare come composizione
 - **altrimenti** → raffinabile solo come aggregazione
- verificare la navigabilità dell'associazione
 - (in generale) nella progettazione si fa riferimento a relazioni unidirezionali

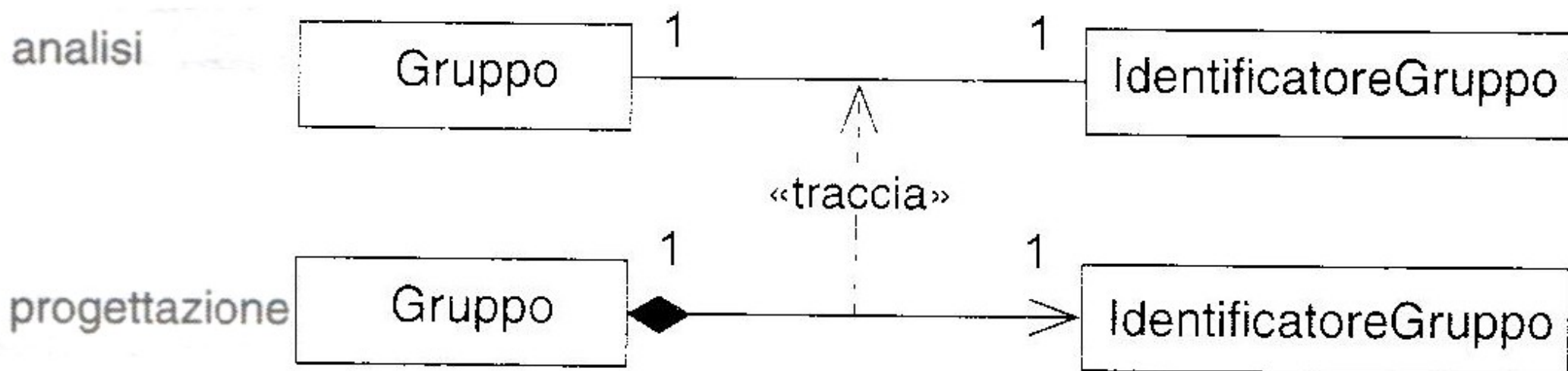
raffinare associazioni di analisi – 2

- associazioni uno-a-uno
 - queste associazioni possono essere raffinate in relazioni di composizione



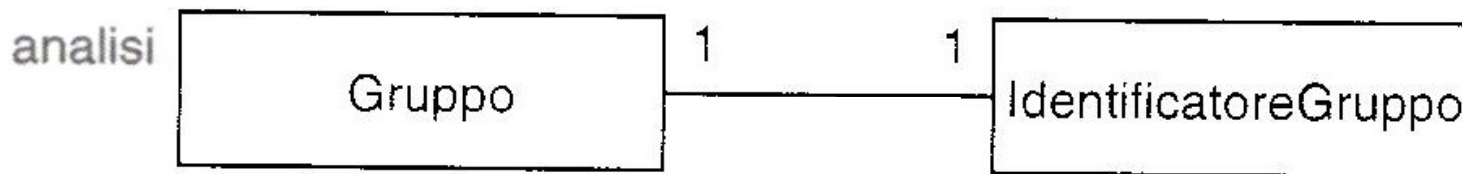
raffinare associazioni di analisi – 2

- associazioni uno-a-uno
 - queste associazioni possono essere raffinate in relazioni di composizione



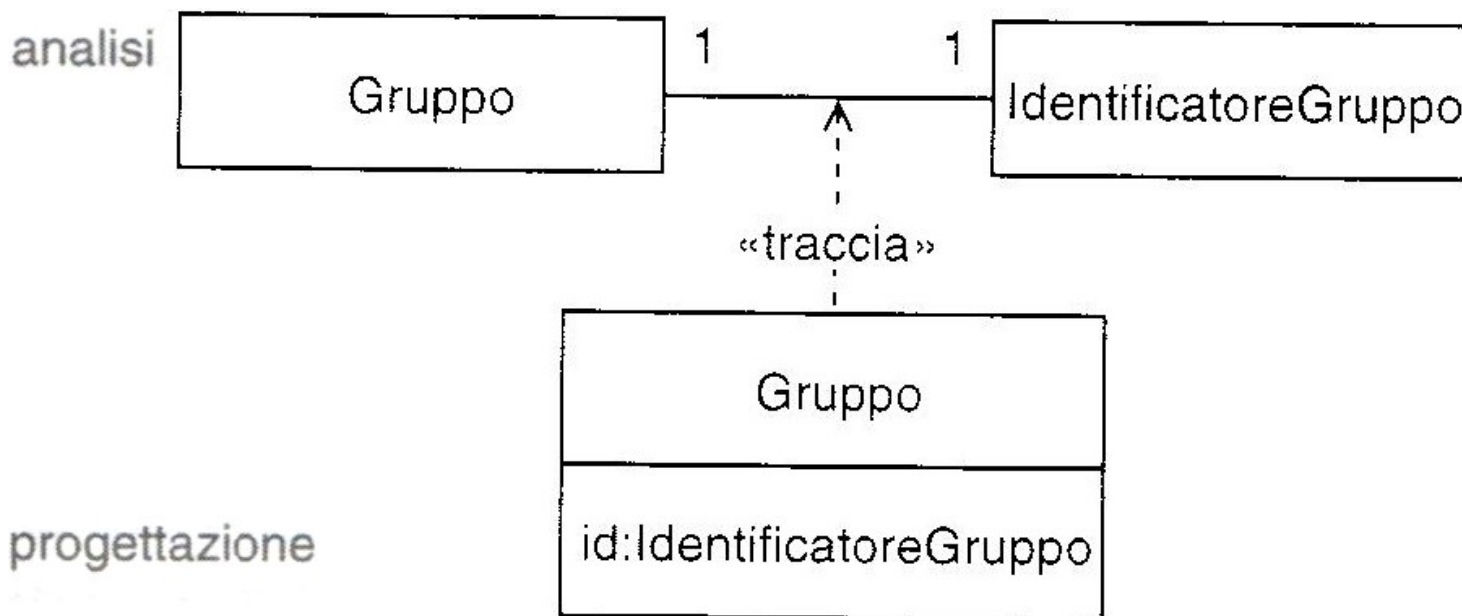
raffinare associazioni di analisi – 2

- associazioni uno-a-uno
 - queste associazioni possono essere raffinate in relazioni di composizione
 - è possibile raffinare l'associazione in una dipendenza (implicita) tra le classi coinvolte
 - definizione di un attributo nella classe che giocherebbe il ruolo *whole* se si considerasse una composizione



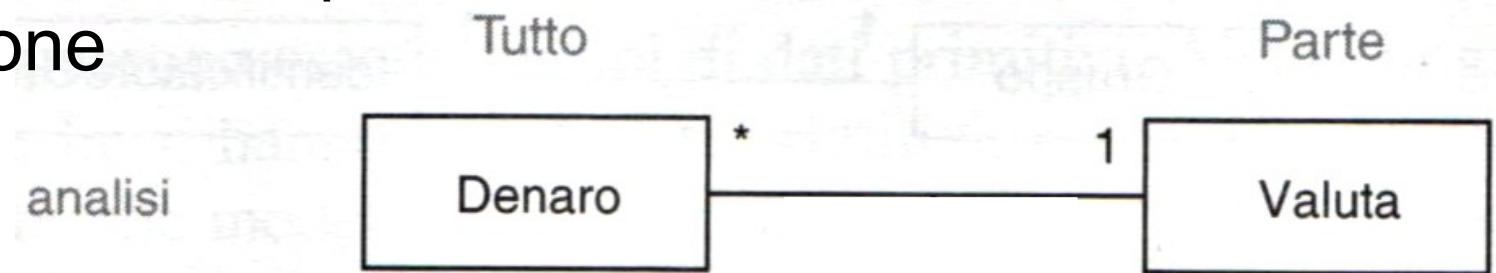
raffinare associazioni di analisi – 2

- associazioni uno-a-uno
 - queste associazioni possono essere raffinate in relazioni di composizione
 - è possibile raffinare l'associazione in una dipendenza (implicita) tra le classi coinvolte
 - definizione di un attributo nella classe che giocherebbe il ruolo *whole* se si considerasse una composizione



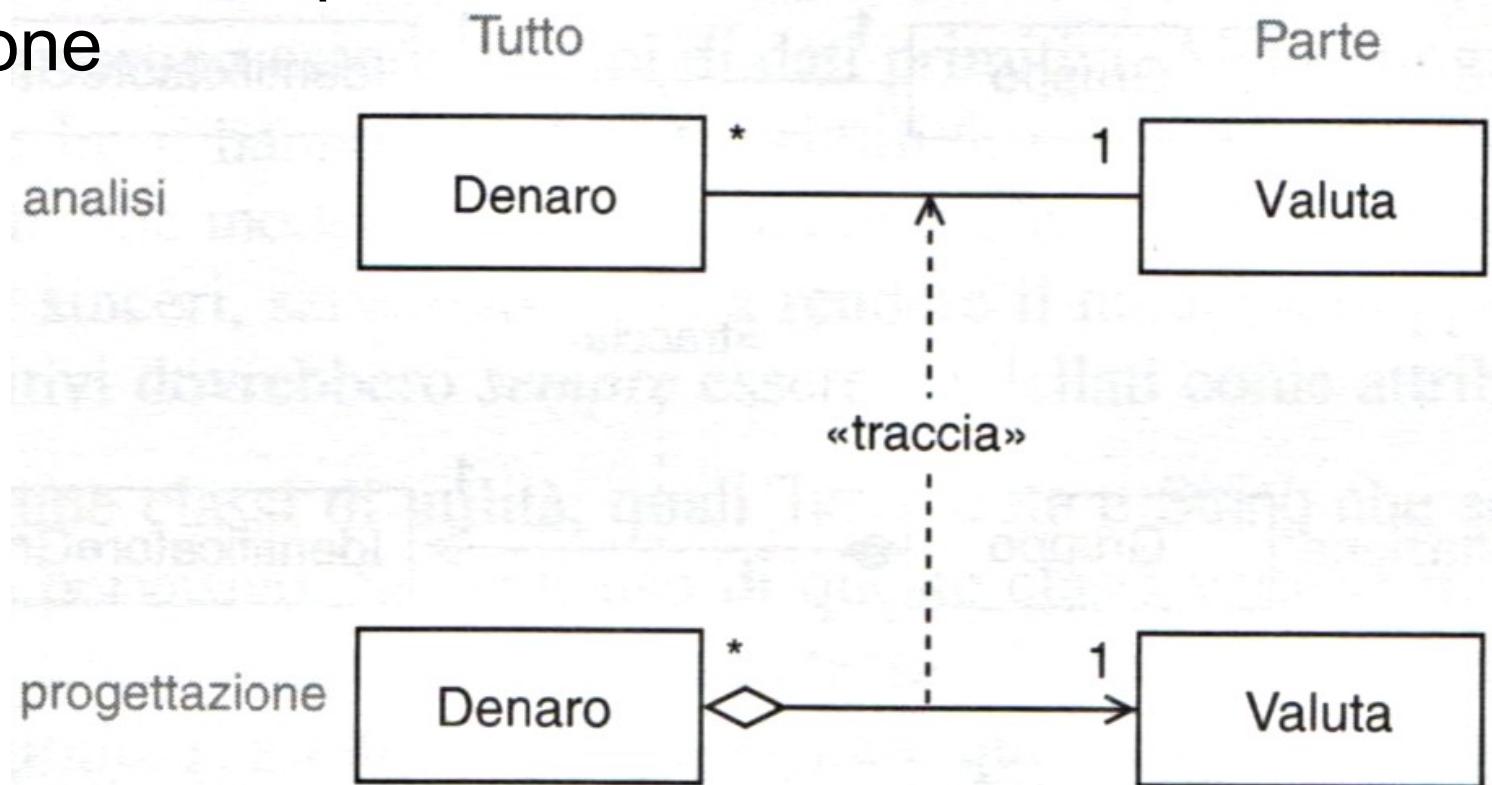
raffinare associazioni di analisi – 3

- associazioni multi-a-uno
 - queste associazioni non sono raffinabili come composizioni
 - il ruolo *part* potrebbe essere condiviso tra più *whole*
 - l'unico raffinamento possibile è in relazione di aggregazione



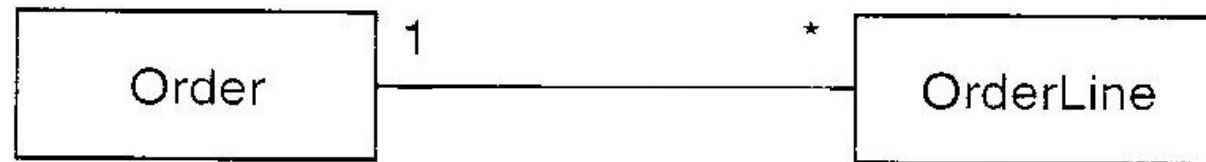
raffinare associazioni di analisi – 3

- associazioni multi-a-uno
 - queste associazioni non sono raffinabili come composizioni
 - il ruolo *part* potrebbe essere condiviso tra più *whole*
 - l'unico raffinamento possibile è in relazione di aggregazione



raffinare associazioni di analisi – 3

- associazioni multi-a-uno
 - queste associazioni non sono raffinabili come composizioni
 - il ruolo *part* potrebbe essere condiviso tra più *whole*
 - l'unico raffinamento possibile è in relazione di aggregazione
 - in alcuni casi potrebbe essere utile introdurre classi intermedie (e.g. classi collezione)



raffinare associazioni di analisi – 3

- associazioni multi-a-uno
 - queste associazioni non sono raffinabili come composizioni
 - il ruolo *part* potrebbe essere condiviso tra più *whole*
 - l'unico raffinamento possibile è in relazione di aggregazione
 - in alcuni casi potrebbe essere utile introdurre classi intermedie (e.g. classi collezione)

