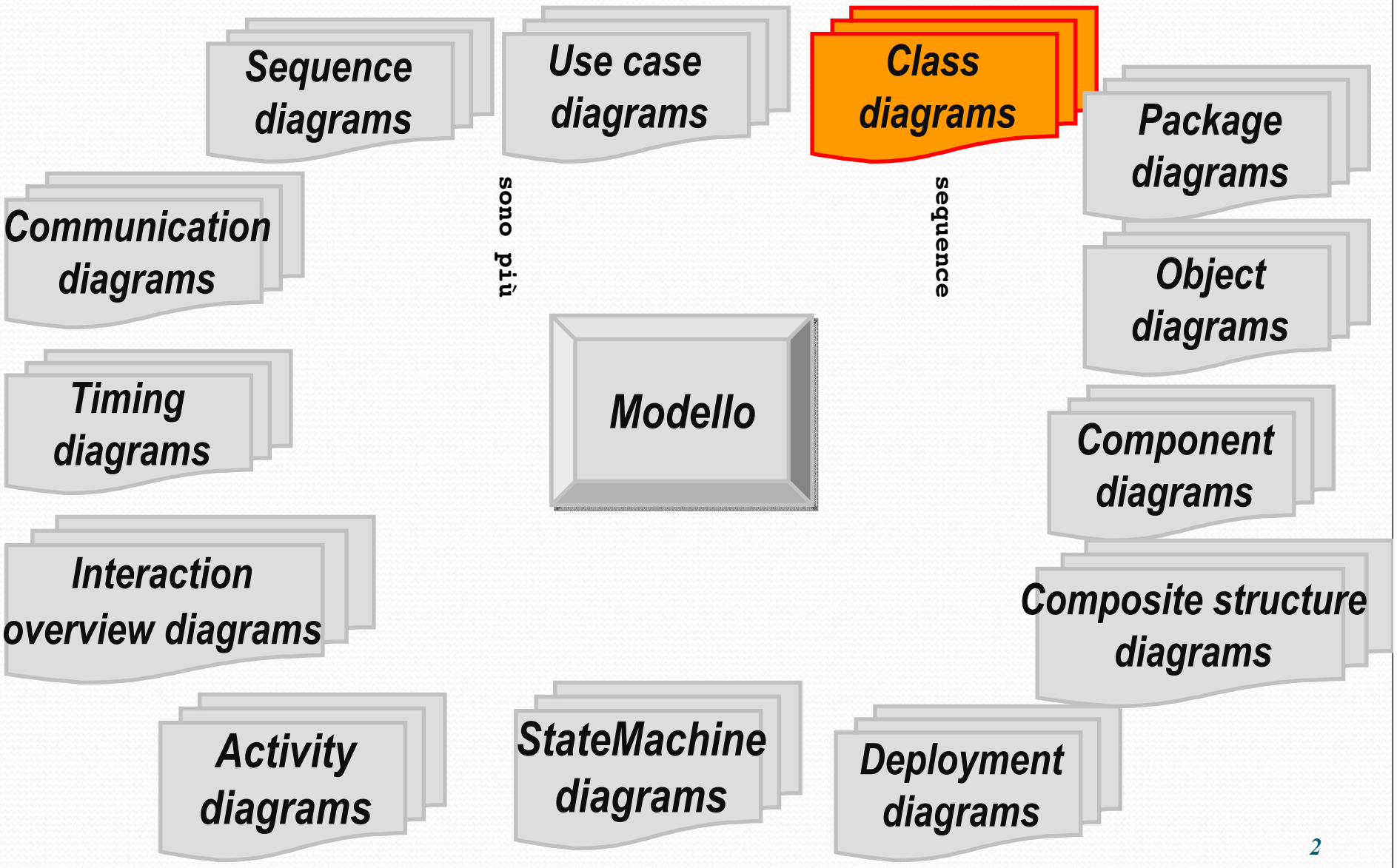
Sviluppo delle applicazioni software

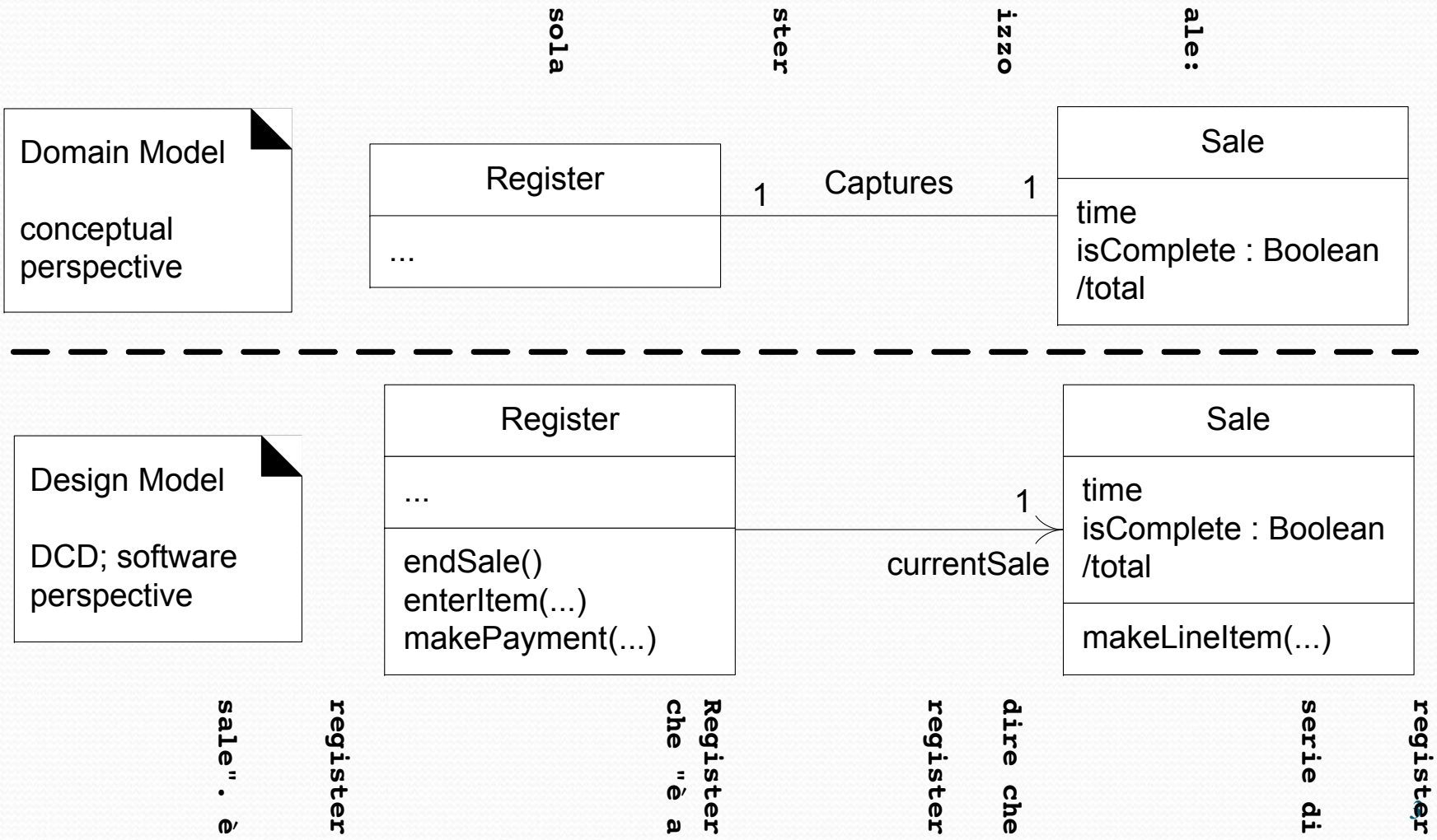
- ***Panoramica su Unified Modeling Language***

Grazie a Simona Bernardi (Universita` di Saragoza, Spagna), che ha messo a disposizione il suo materiale

Modellazione con UML



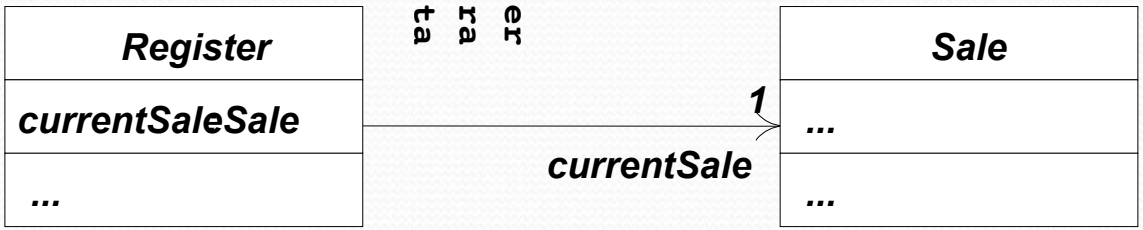
Notazione Classi



Attributi come associazioni

Possibili alternative

NO: Introduce ridondanza

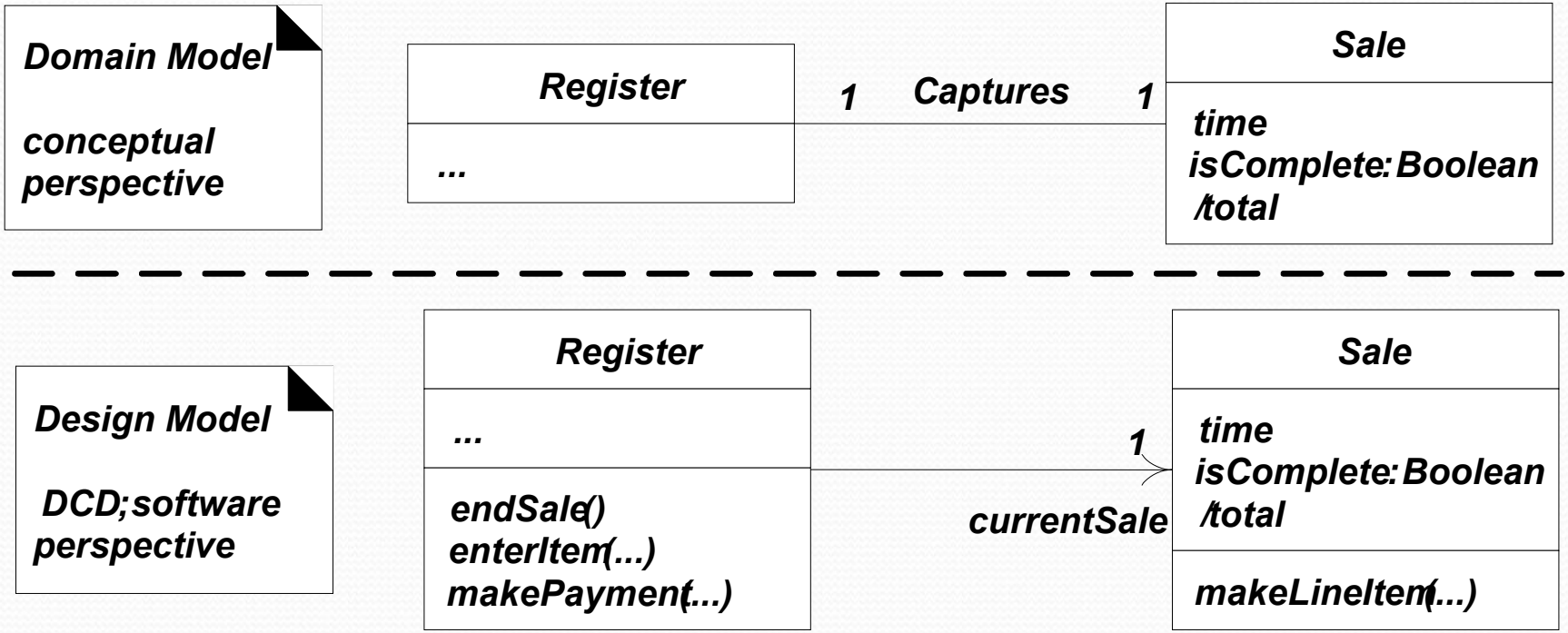


Sintassi per gli attributi

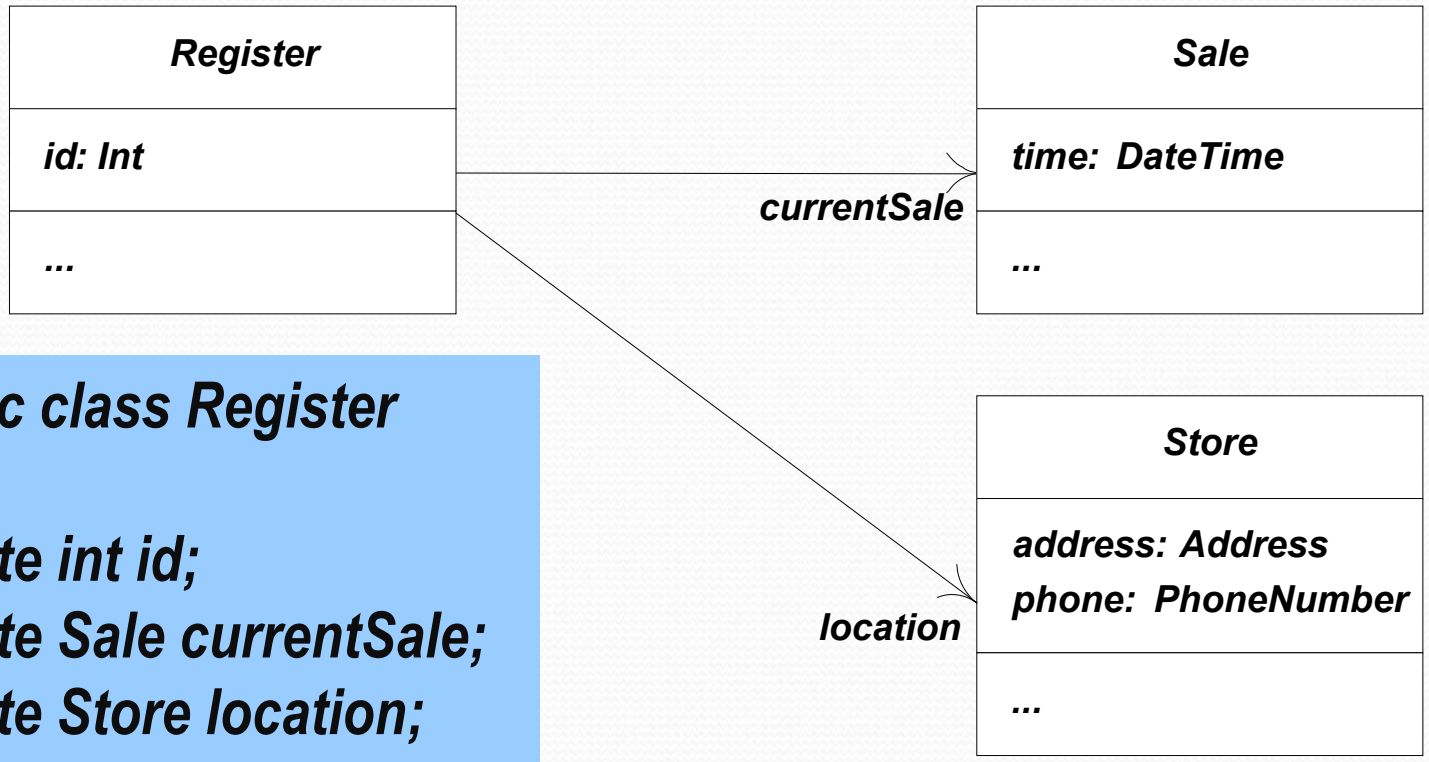
visibility name: type multiplicity = default {property}

- Per default gli attributi sono definiti come privati (visibility="-")
- Quando si usano "attributi come associazioni" nel DCD
  - Indicare la freccia di navigabilità
  - La molteplicità (se <>1) della destinazione ma non della sorgente (normalmente)
  - Nome del ruolo (= nome dell'attributo) della destinazione
  - Nessun nome all'associazione

Associazioni nei diversi modelli



Attributo o associazione ?

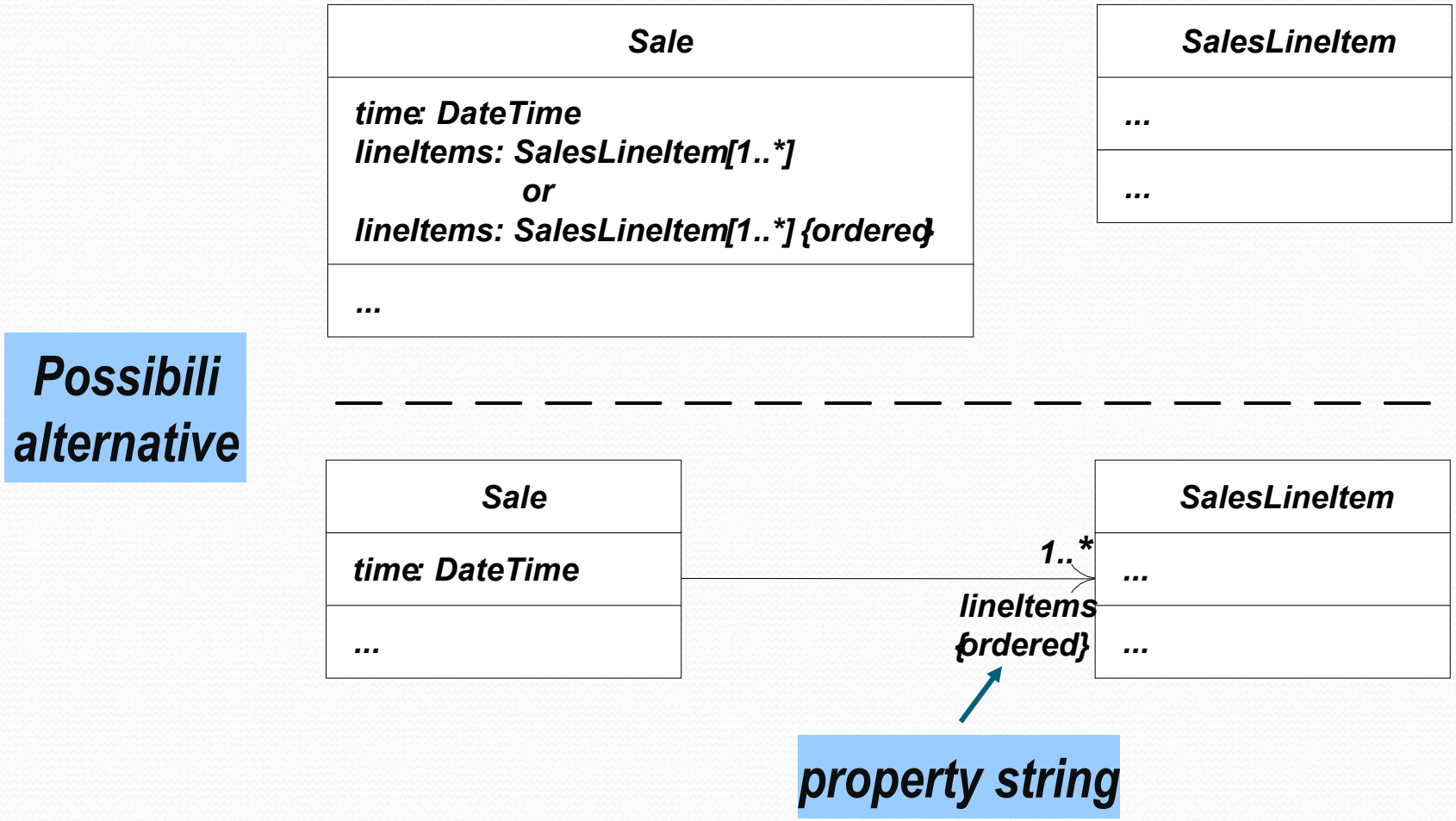



```
public class Register
{
private int id;
private Sale currentSale;
private Store location;
.....
}
```

o, basta

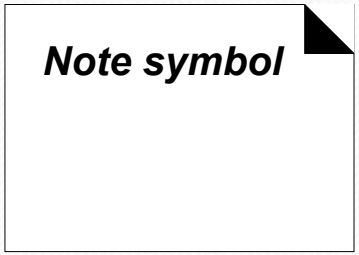
location

Attributi collezione



Simbolo note

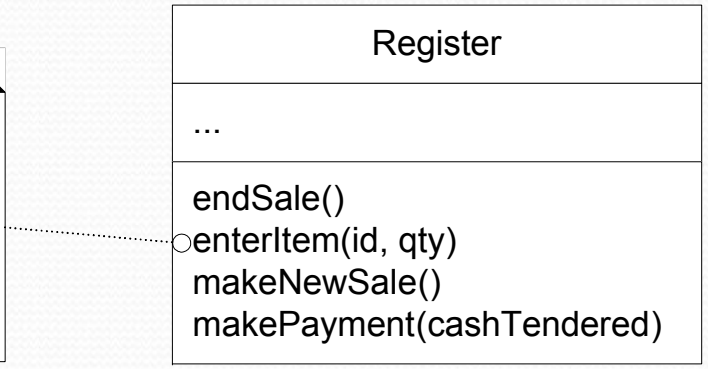
- Note o commenti – in linguaggio naturale
- Vincoli, in tal caso sono posti tra parentesi graffe (in linguaggio naturale, OCL, etc.)
- Metodi – implementazione operazioni



Operazioni e metodi

- Un'operazione è una dichiarazione di un metodo
- Sintassi:
  - **visibility name (parameter-list): return-type {property}**
- Per default, le operazioni hanno visibilità pubblica
- Nei diagrammi di classe vengono solitamente indicate le operazioni (signature – nomi e parametri)
- Nei diagrammi di interazione vengono modellati i metodi, come sequenze di messaggi

```
«method»
// pseudo-code or a specific language is OK
public void enterItem( id, qty )
{
   ProductDescription desc = catalog.getProductDescription(id);
   sale.makeLineItem(desc, qty);
}
```

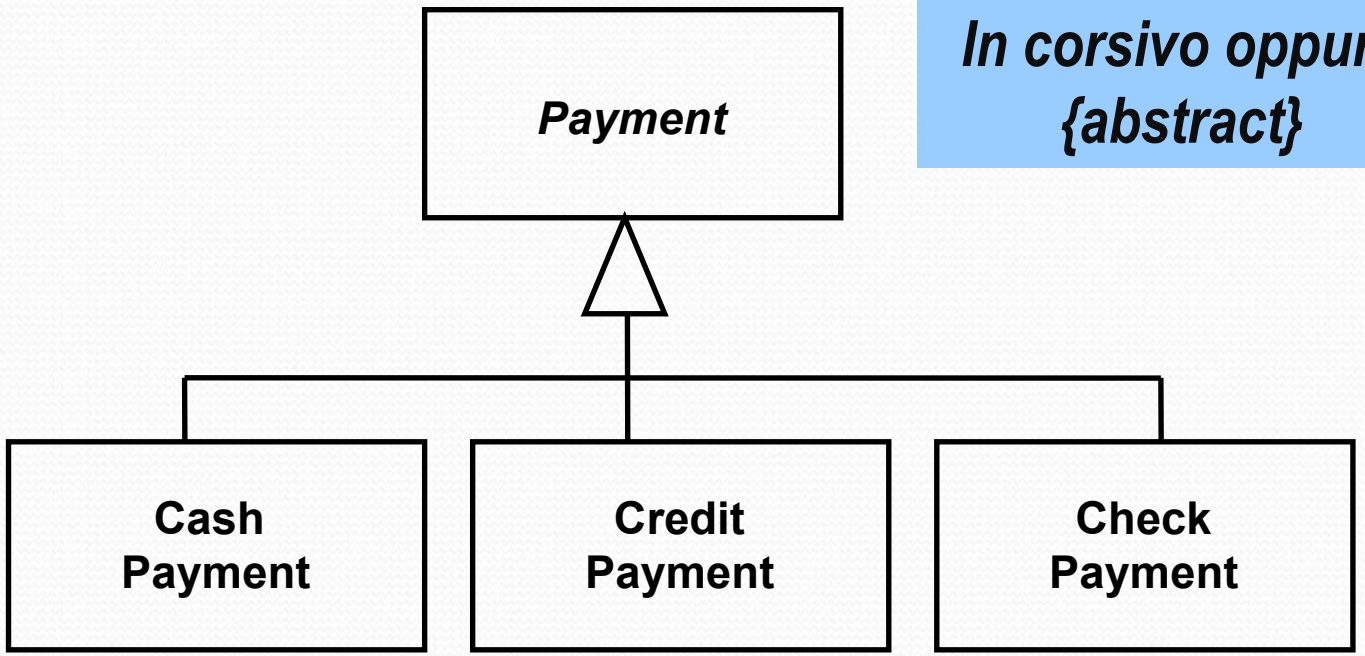


Keywords

Keyword	*Significato*	*Esempi*
<<actor>>	***Il classifier è un attore***	***Diagrammi di classe, sopra il nome della classe***
<<interface>>	***Il classifier è una interfaccia***	***Diagrammi di classe, sopra il nome della classe***
{abstract}	***Elemento astratto non può essere istanziato***	***Diagrammi di classe, sotto il nome della classe***
{ordered}	***Un insieme di oggetti con un determinato ordine***	***Diagrammi di classi, nelle association-end***

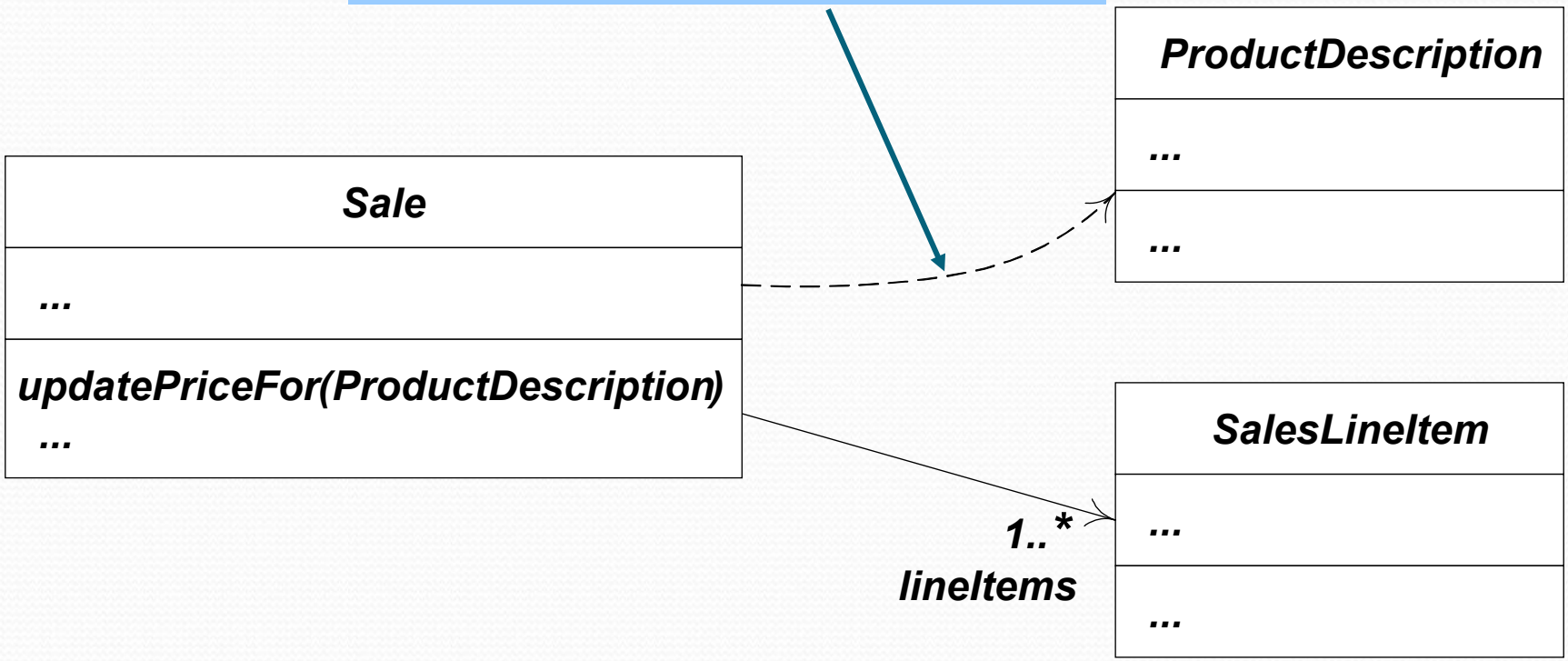
Generalizzazione

Classe astratta:
In corsivo oppure
{abstract}

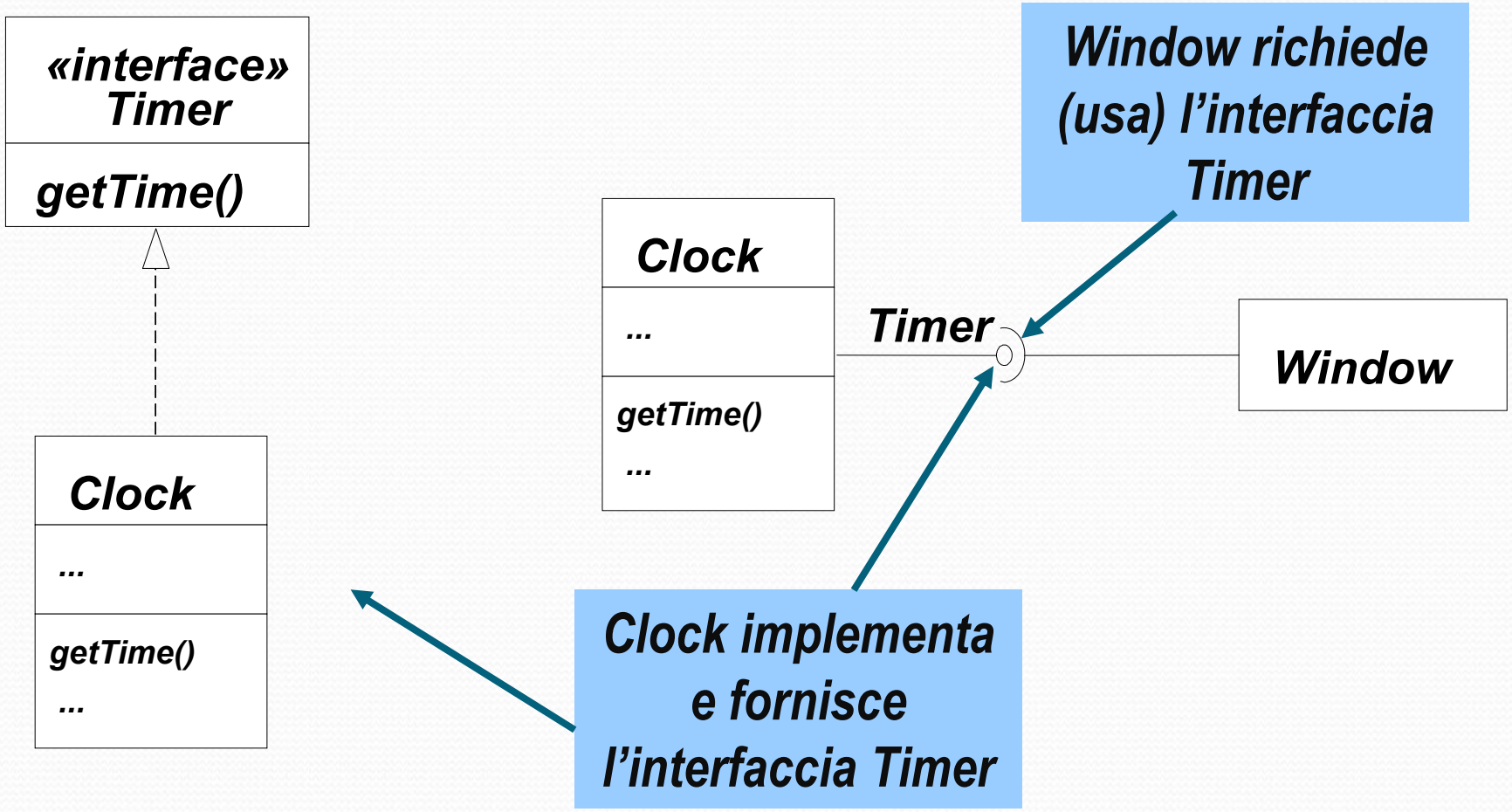


Dipendenze (=accoppiamento)

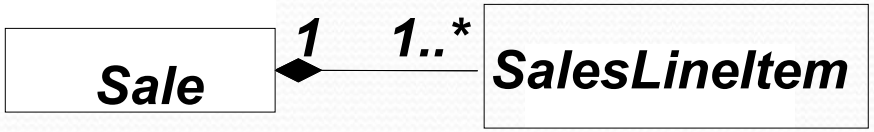
Sale ha visibilità verso ProductDescription, quindi una dipendenza



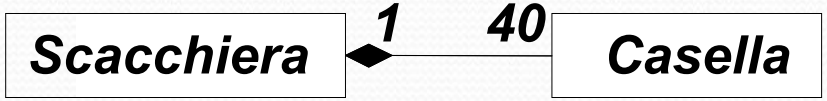
Interfacce


«interface»
Timer
getTime()
Clock
...
getTime()
...
Clock
...
getTime()
...
Timer
Window
Window richiede (usa) l'interfaccia Timer
Clock implementa e fornisce l'interfaccia Timer

Composizione



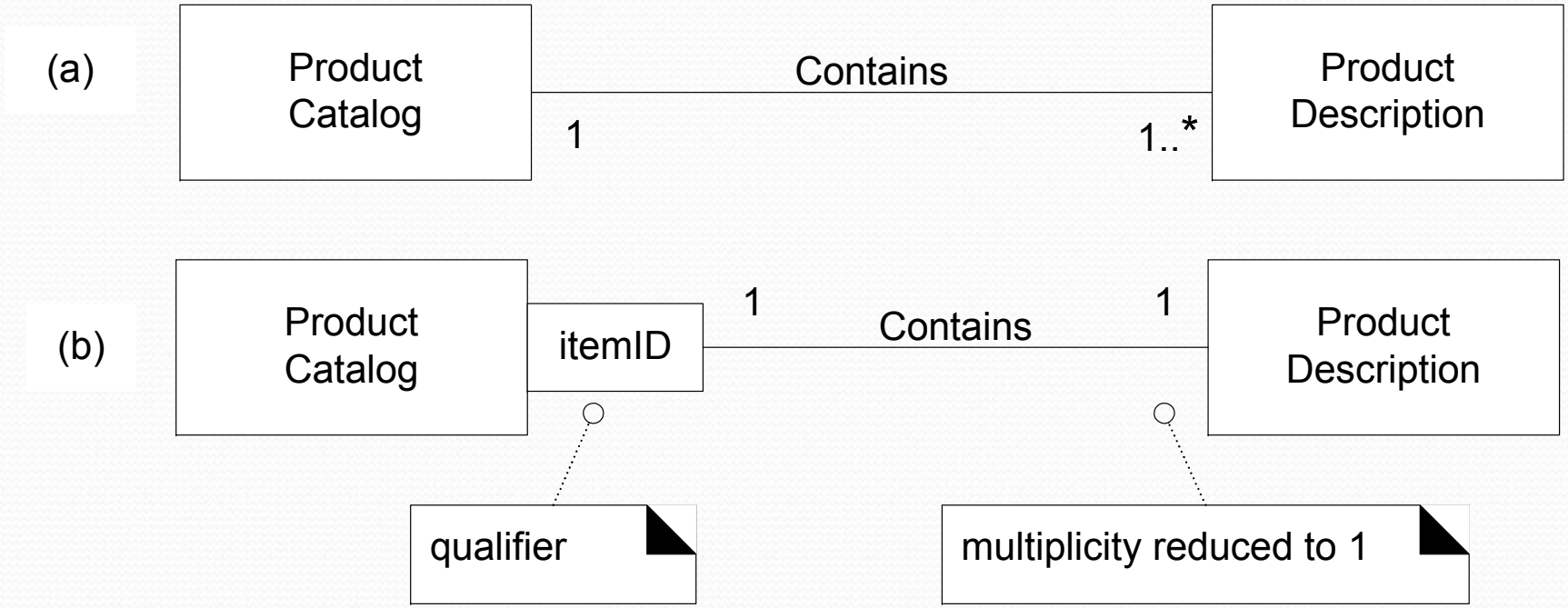
Sale è la unica classe che ha responsabilità sulla creazione/distruzione delle istanze SalesLineItem



Una istanza Casella può far parte di solo una scacchiera

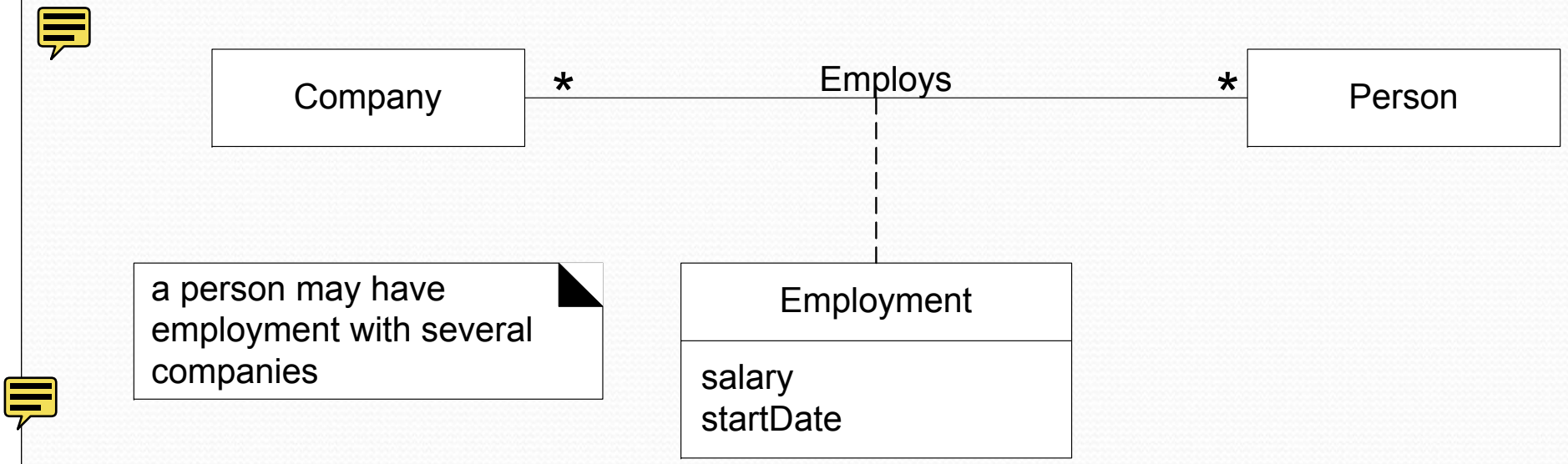
Associazioni qualificate

- ***Si usa** per selezionare un oggetto da un insieme di oggetti correlati in base ad una chiave (es. oggetti di una HashMap).*



Classi associazione

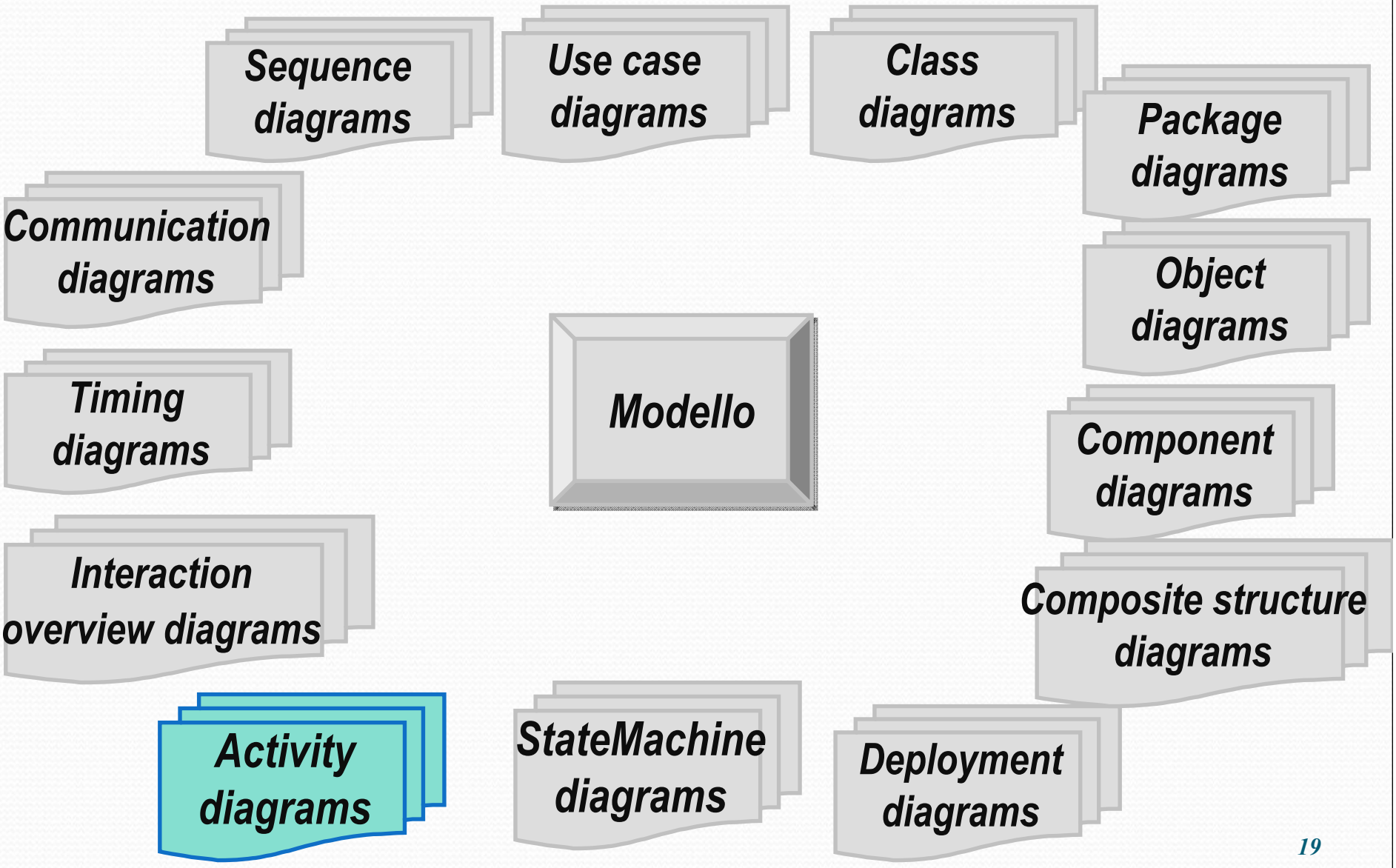
- ***Si tratta una associazione come se fosse una classe (con attributi)***



Classi Singleton

ServicesFactory	***1***
instance:ServicesFactory ***accountingAdapter: IAccountingAdapter*** ***inventoryAdapter: IInventoryAdapter*** ***taxCalculatorAdapter: ITaxCalculatorAdapter***	
getInstance(): ServiceFactory ***getAccountingAdapter(): IAccountingAdapter*** ***getInventoryAdapter(): IInventoryAdapter*** ***getTaxCalculatorAdapter(): ITaxCalculatorAdapter***	

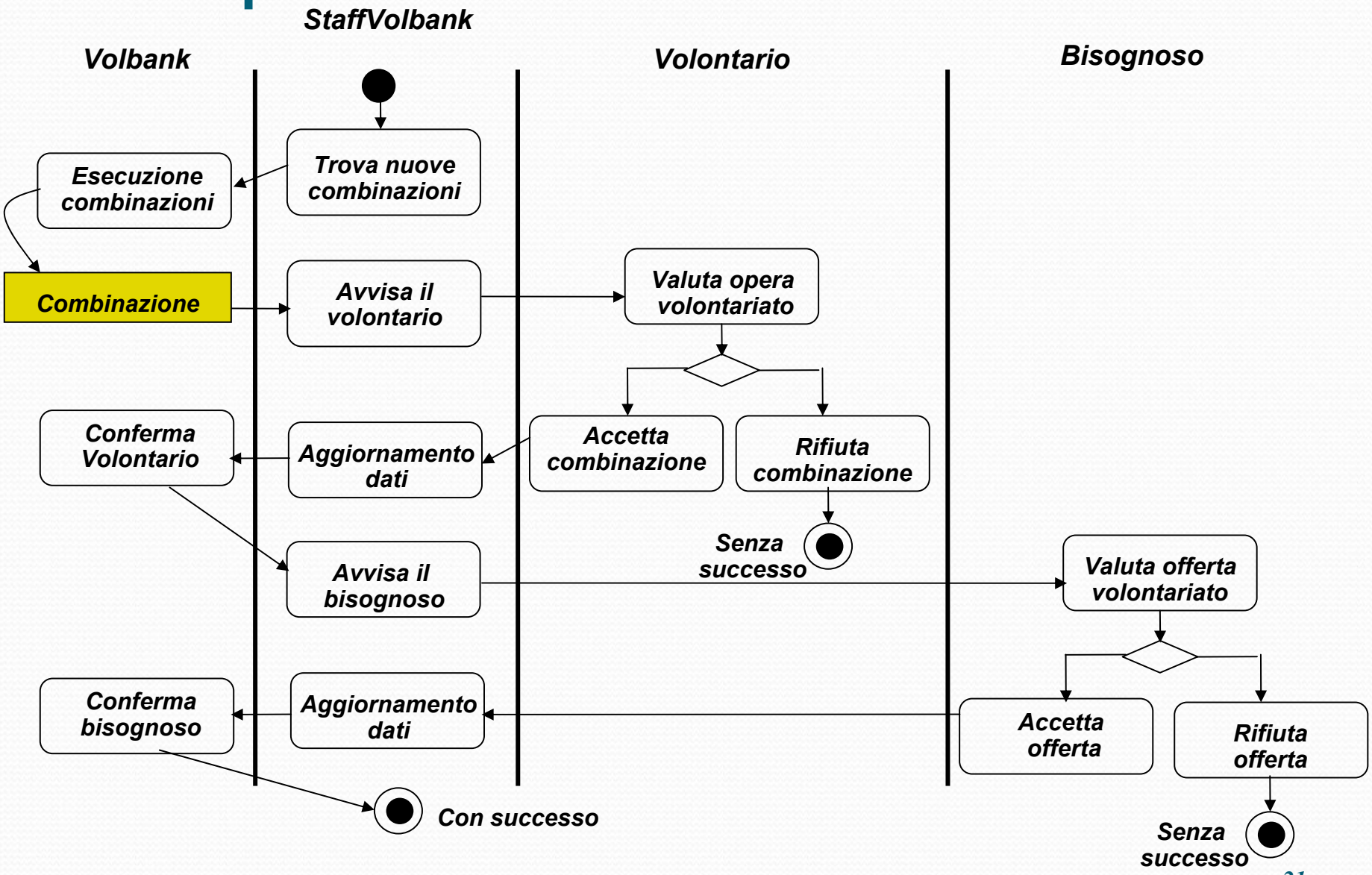
Modellazione con UML



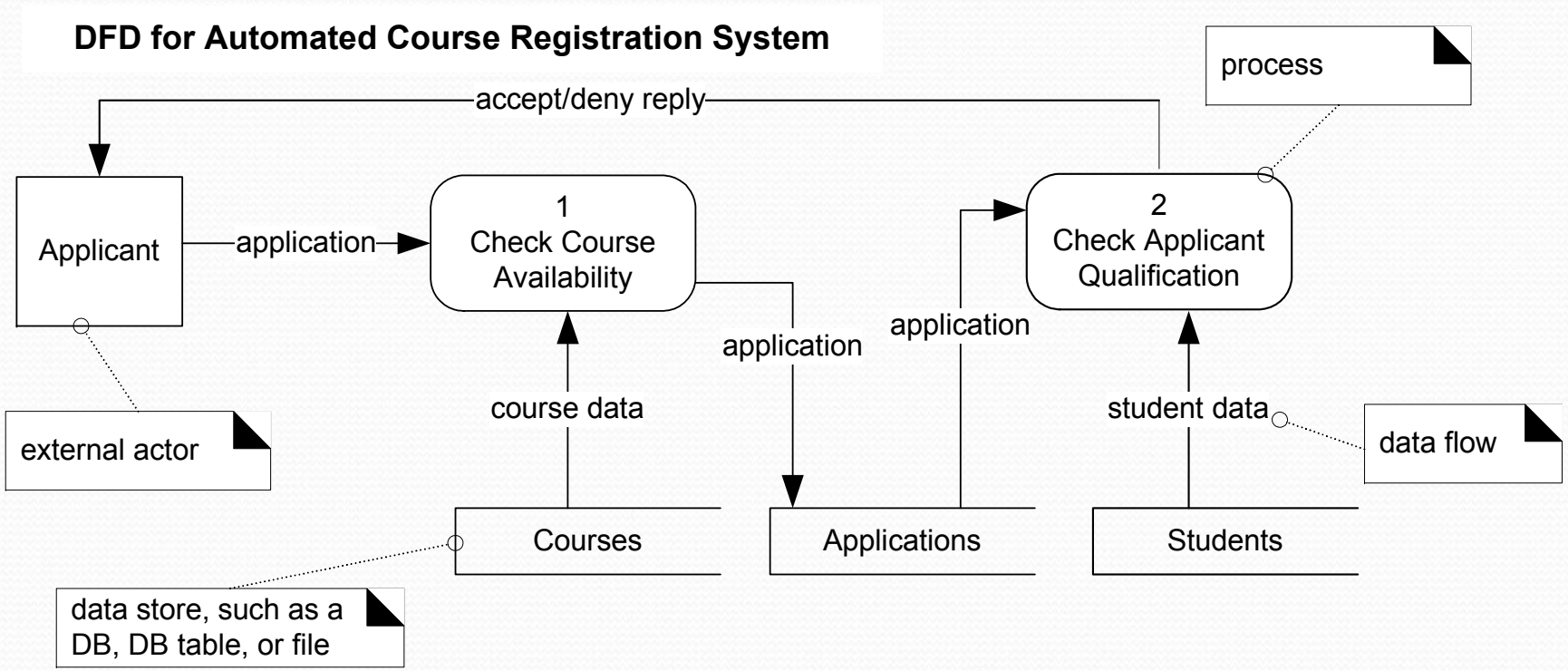
Diagrammi di attività UML

- Rappresentano attività sequenziali e parallele di un processo
- Si possono usare per modellare
  - Workflows – flussi di attività
  - Dataflows – flussi dei dati al posto dei DFD
  - Algoritmi complessi
- Modellazione del business in UP
  - **Business Object Model** (superinsieme del modello di dominio) modella il funzionamento del business in termini di diagrammi di classe, di sequenza e di attività

Esempio in VolBank

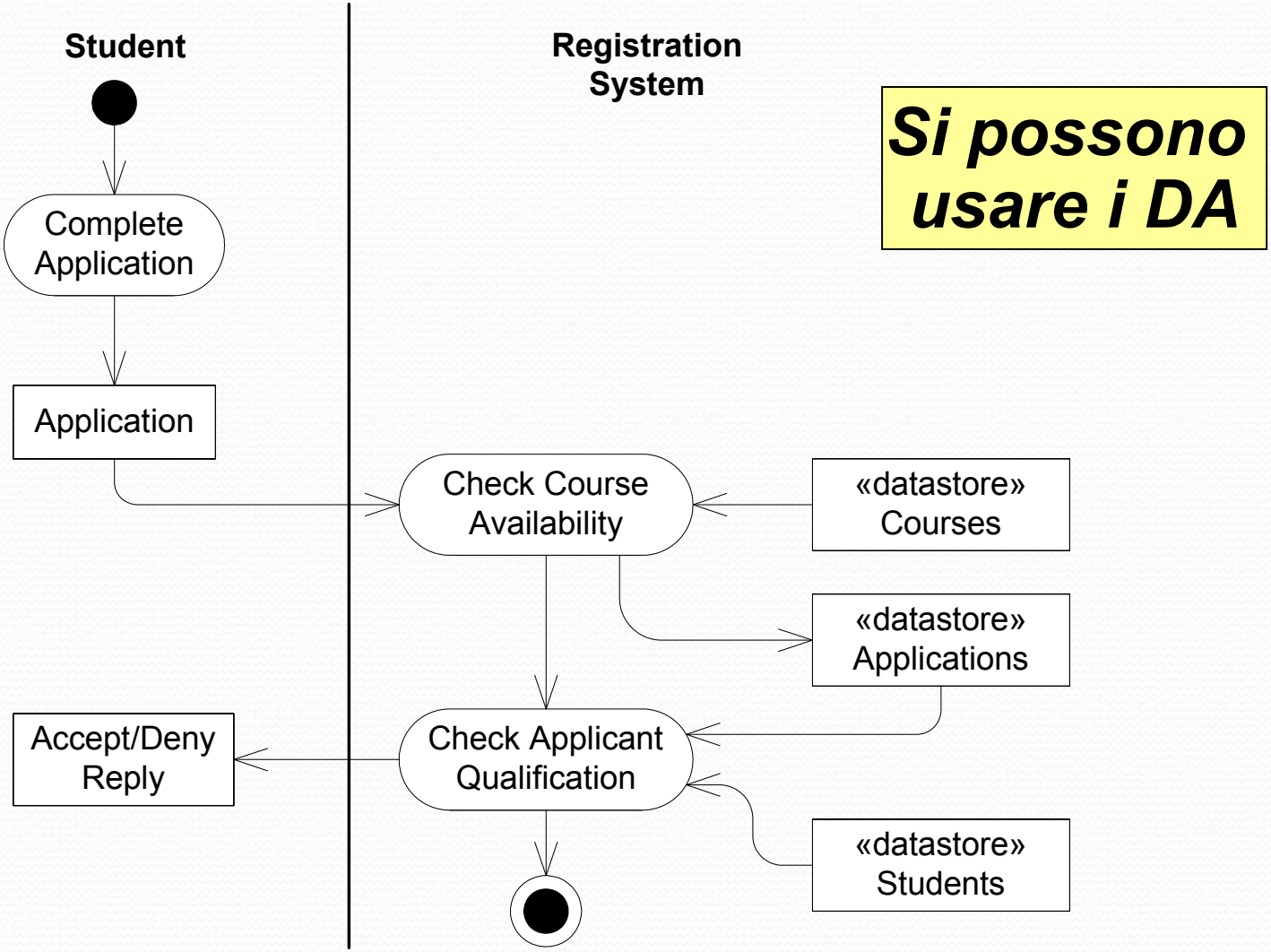


Diagrammi attività e DFD - I



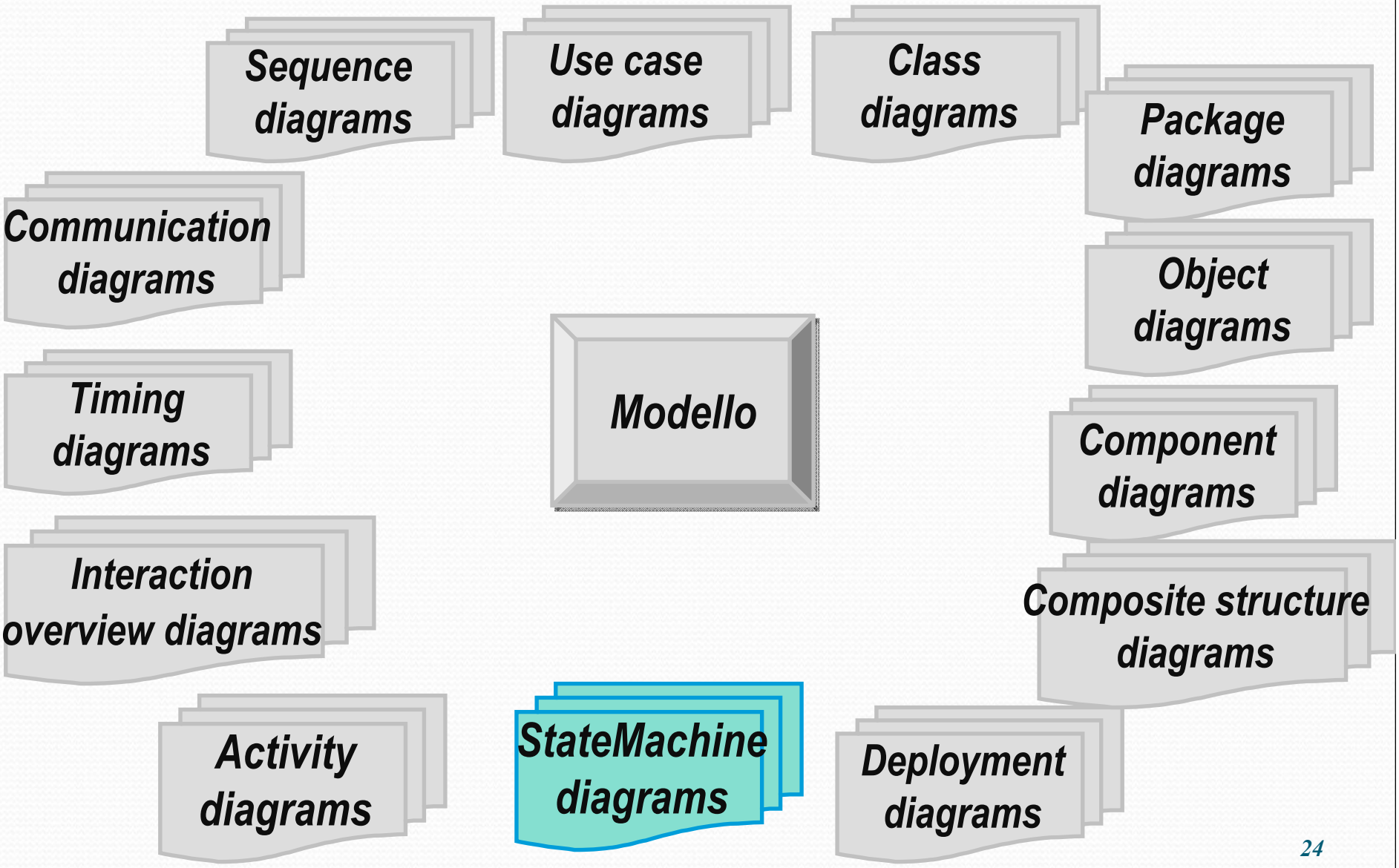
In UML non ci sono DFD

Diagrammi attività e DFD - II



Si possono usare i DA

Modellazione con UML



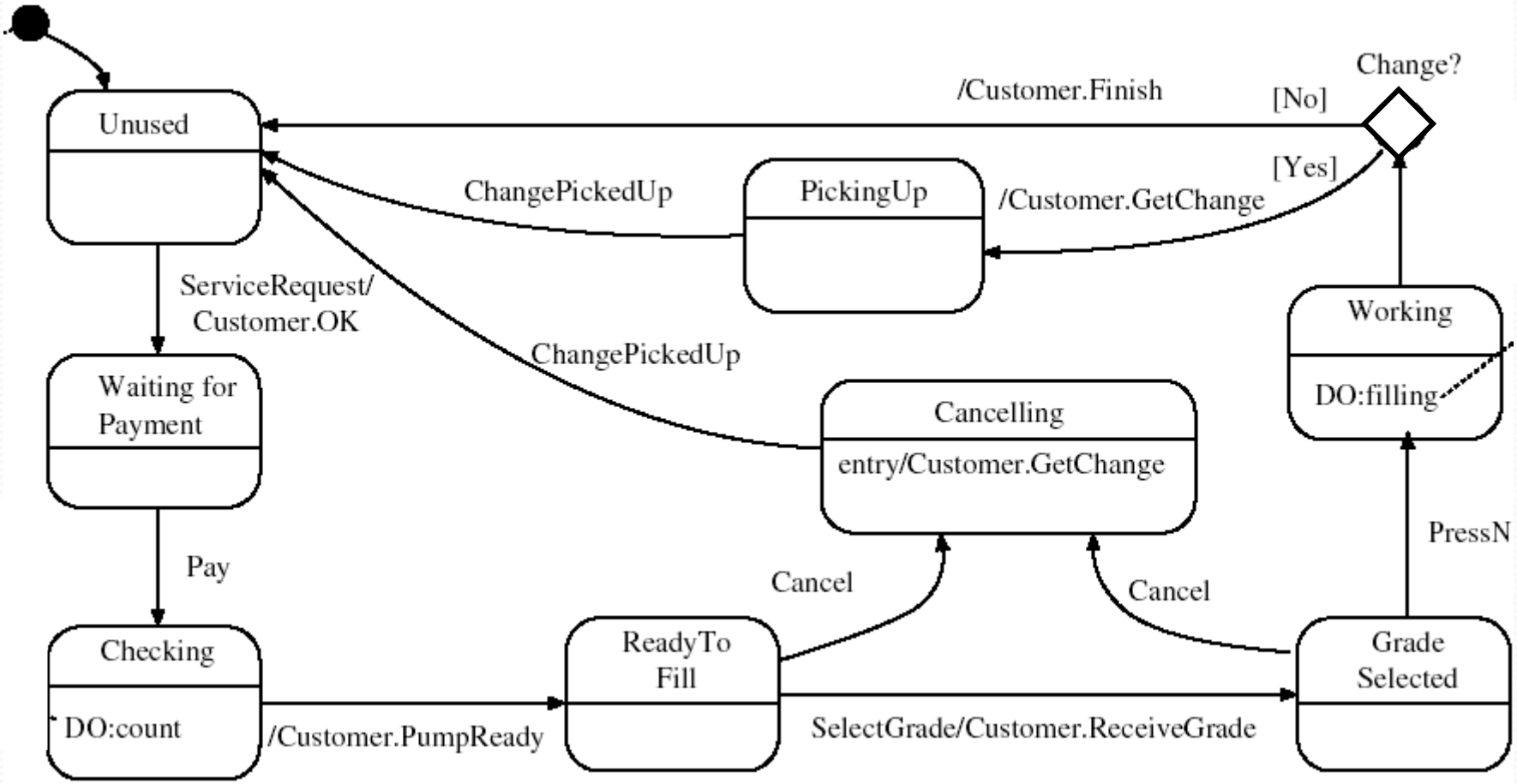
Macchine a stati UML

- Servono per modellare aspetti dinamici del sistema
  - Eventi, stati di un oggetto, il suo comportamento come reazione ad un evento (trigger)
- Usare le macchine a stati per modellare oggetti
  - Il cui comportamento dipende dallo stato (es. valori degli attributi)
  - con un comportamento complesso
- Due approcci
  - Modellazione del comportamento reattivo di un oggetto
  - Modellazione di sequenze corrette delle operazioni (specifica protocolli/linguaggi)

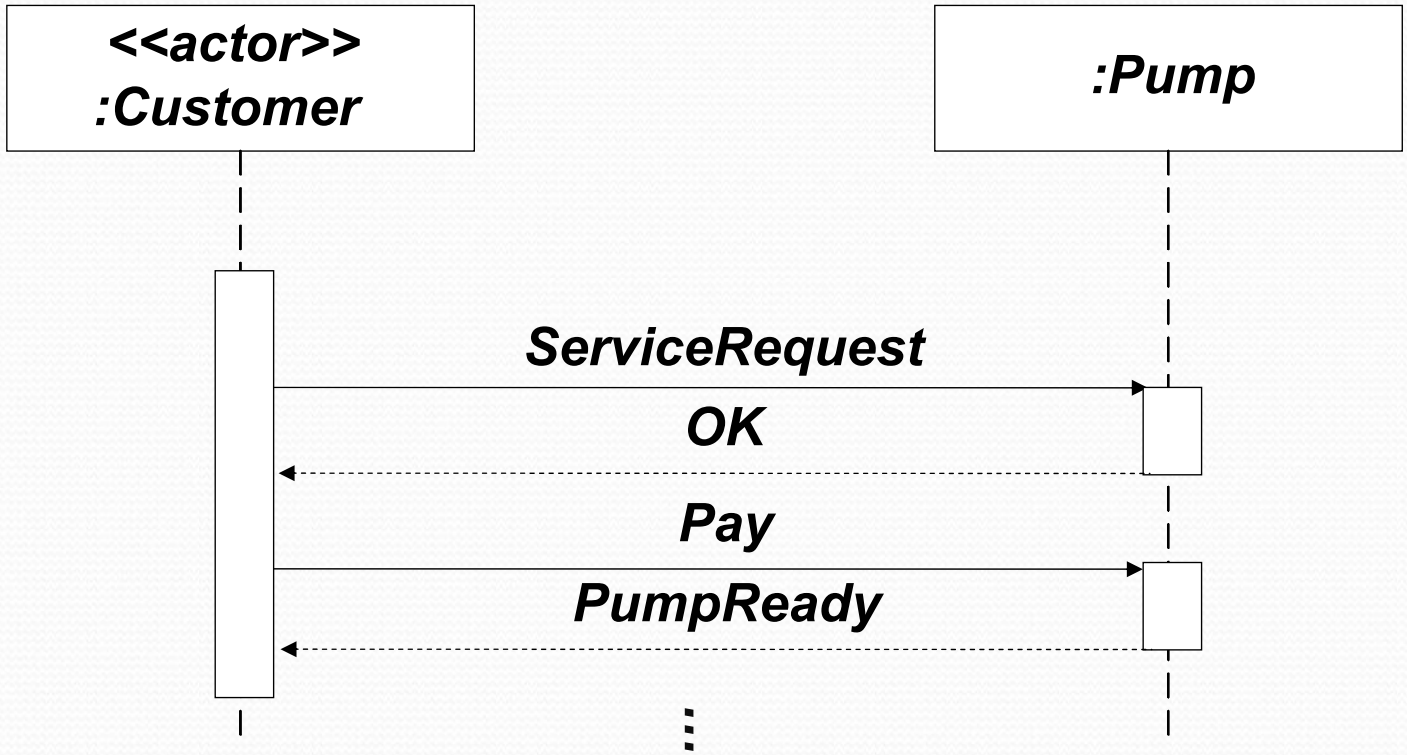
Modellazione oggetti reattivi

- Apparati fisici controllati dal software (sistemi integrati)
  - Telefono, forno a microonde, sistemi di controllo negli aerei,etc.
- Transazioni e oggetti di business
  - vendite, ordine pagamenti, etc.
- Oggetti che cambiano ruolo
  - Una persona, ogni ruolo che assume si rappresenta come stato

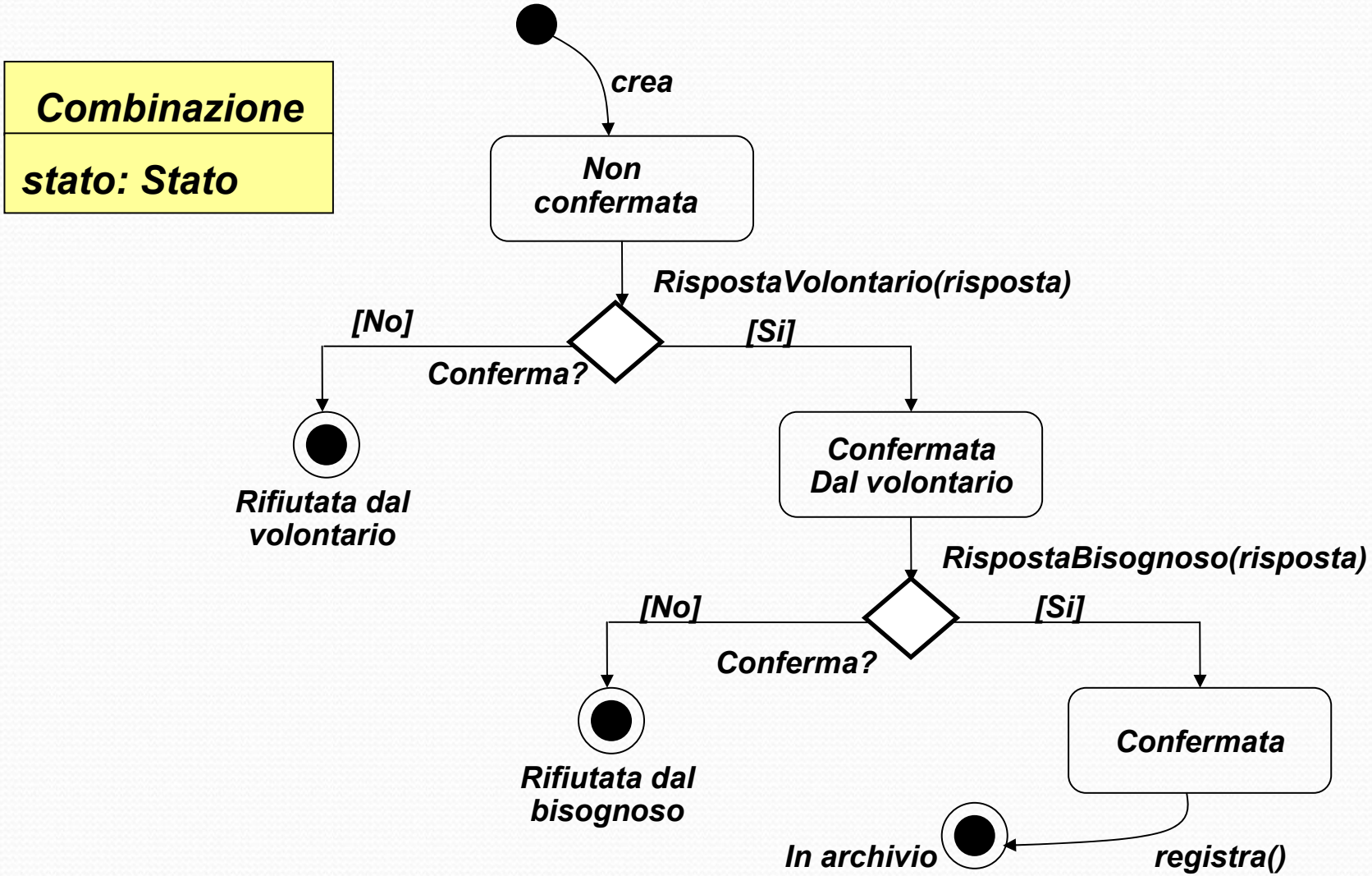
Es: Pompa di benzina



Relazione con i diagrammi di interazione (es.pompa di benzina)



Esempio in Volbank



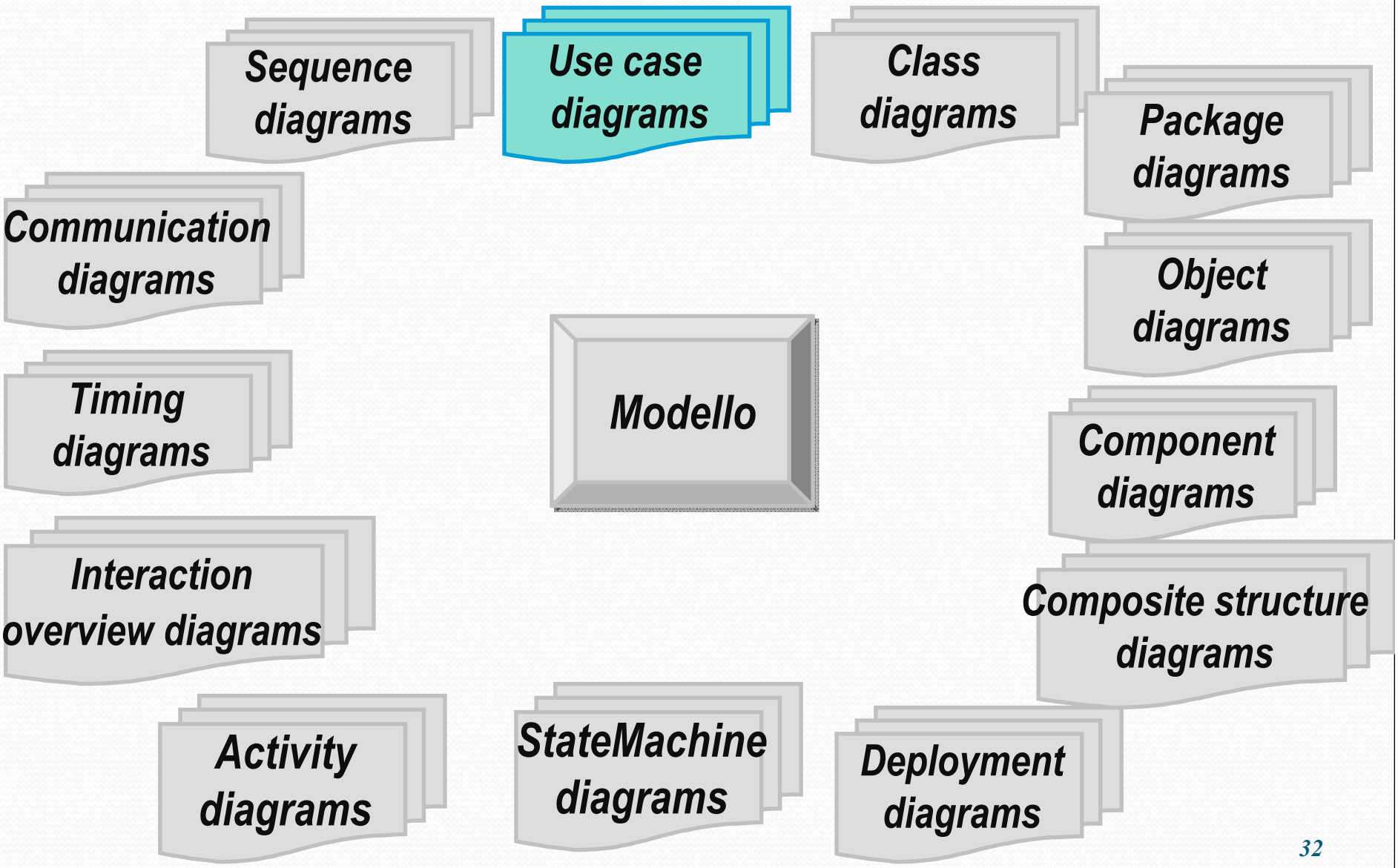
Modellazione di protocolli e sequenze corrette

- Protocolli di comunicazione
- Flusso o navigabilità finestre UI
- Controllori di flusso UI, sessioni
- Operazioni di sistema dei casi d'uso
- Gestione eventi in una finestra UI
- ...

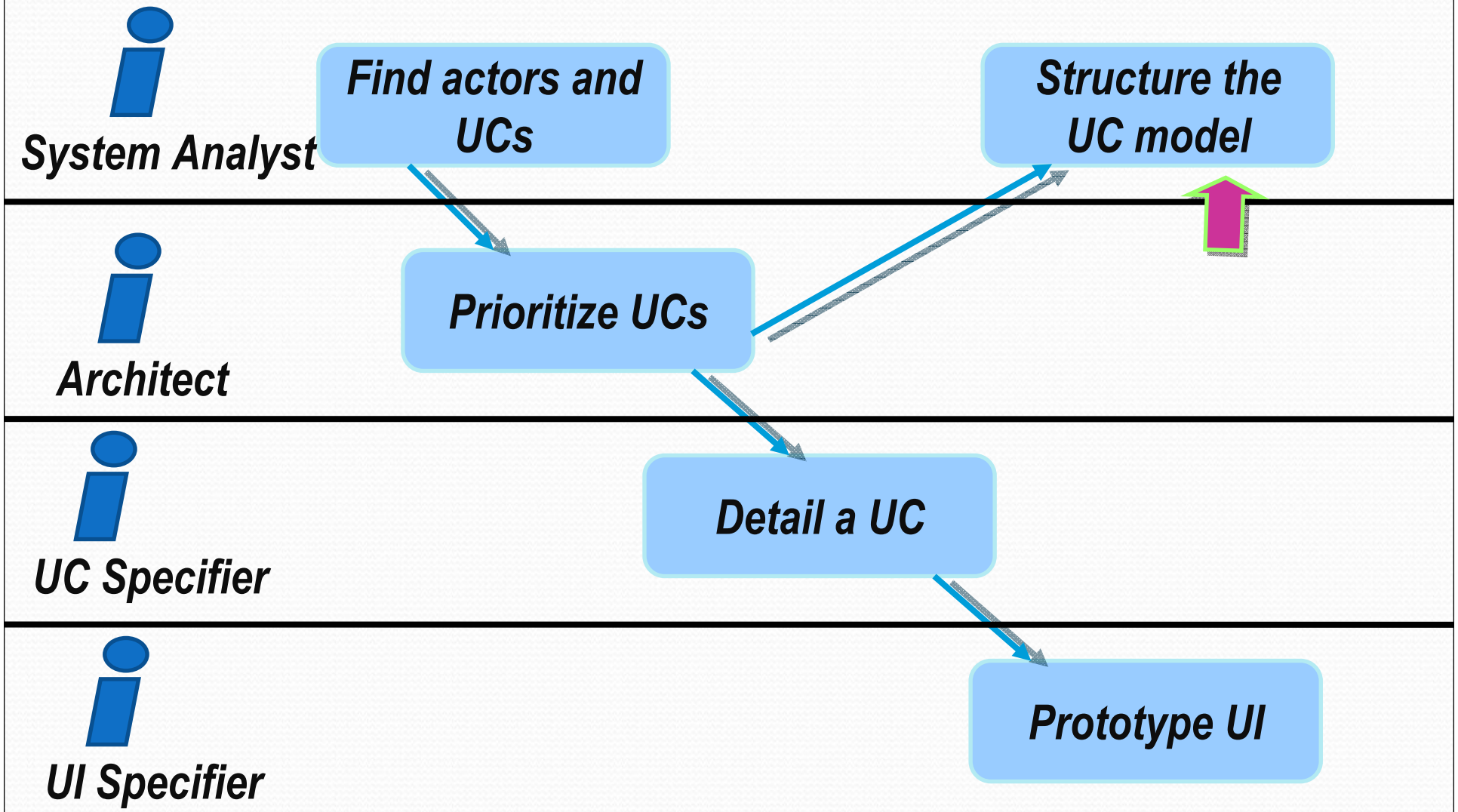
Macchine a stati in UP

- Non c'è un modello UP "modello a stati"
- Qualunque elemento di qualunque modello UP (modello dei casi d'uso, modello di dominio, modello di progetto, etc.) può includere una macchina a stati

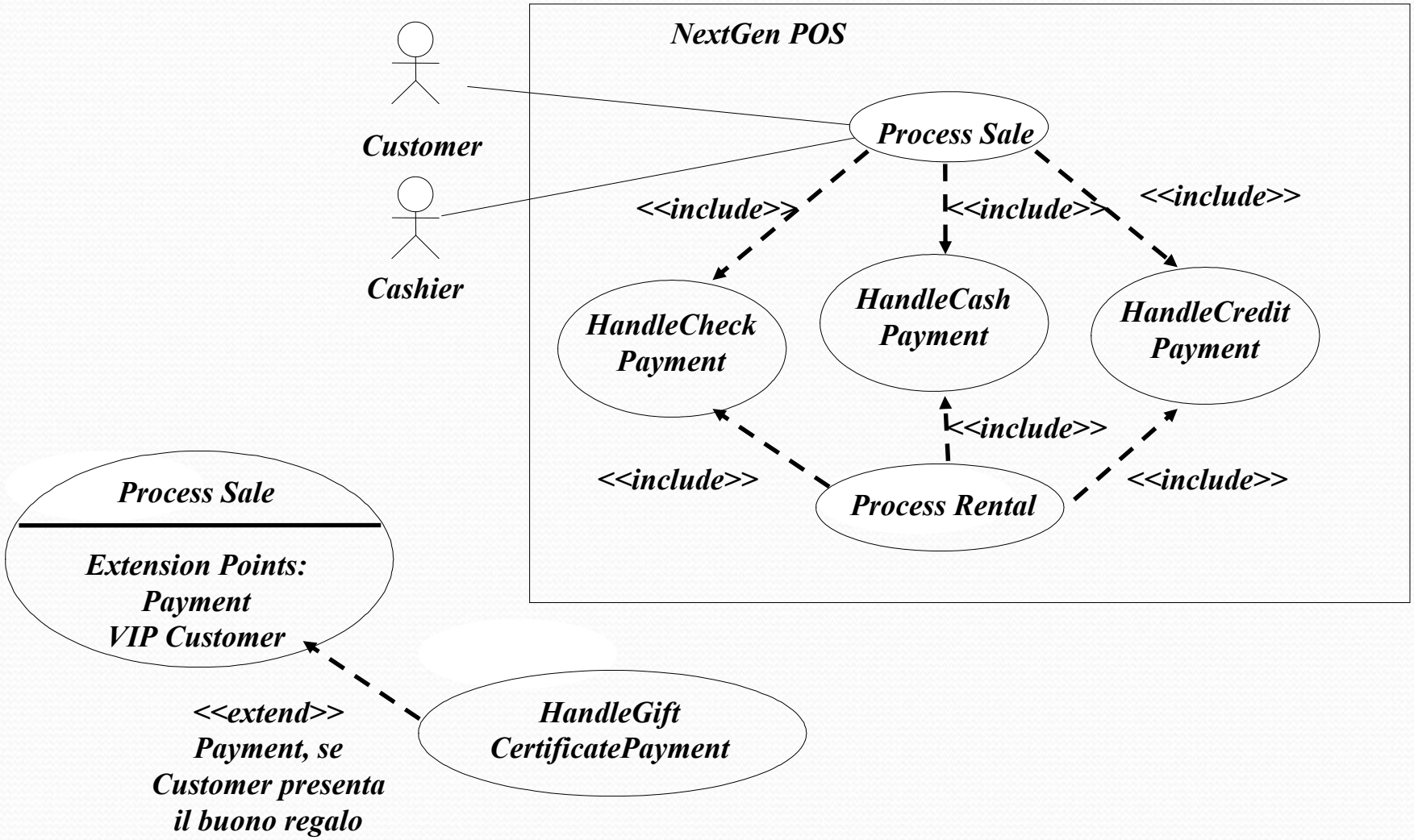
Modellazione con UML



Flusso delle attività (con UC)



Correlare i casi d' uso - I



Correlare i casi d'uso - II

- Usare **<<include>>** quando ci sono comportamenti (nei flussi principale/alternativi) comuni a diversi casi d'uso oppure per casi d´uso complessi
  - Il caso d'uso incluso è a livello di sotto-funzione

(UC Process Sale)

Scenario principale:

1. Il Cliente arriva alla cassa PoS

2. ..

7. Il Cliente paga e gestisce il Pagamento

Estensioni:

7b. Pagamento con carta di credito: HandleCreditPayment

7b. Pagamento con assegno: HandleCheckPayment

.

Correlare i casi d'uso - III

- Usare **<<extend>>** quando il caso d´uso non deve essere modificato (es.versione stabile a fine iterazione)
  - Il caso d'uso incluso è a livello di sotto-funzione
  - Indicare il punto di estensione

(UC Process Sale)

Punti di estensione:

Cliente VIP, passo 1. Pagamento, passo 7.

Scenario principale:

1. Il Cliente arriva alla cassa PoS

2. ..

7. Il Cliente paga e gestisce il Pagamento

(UC HandleGiftCertificatePayment)

Punti di estensione: Pagamento in Process Sale

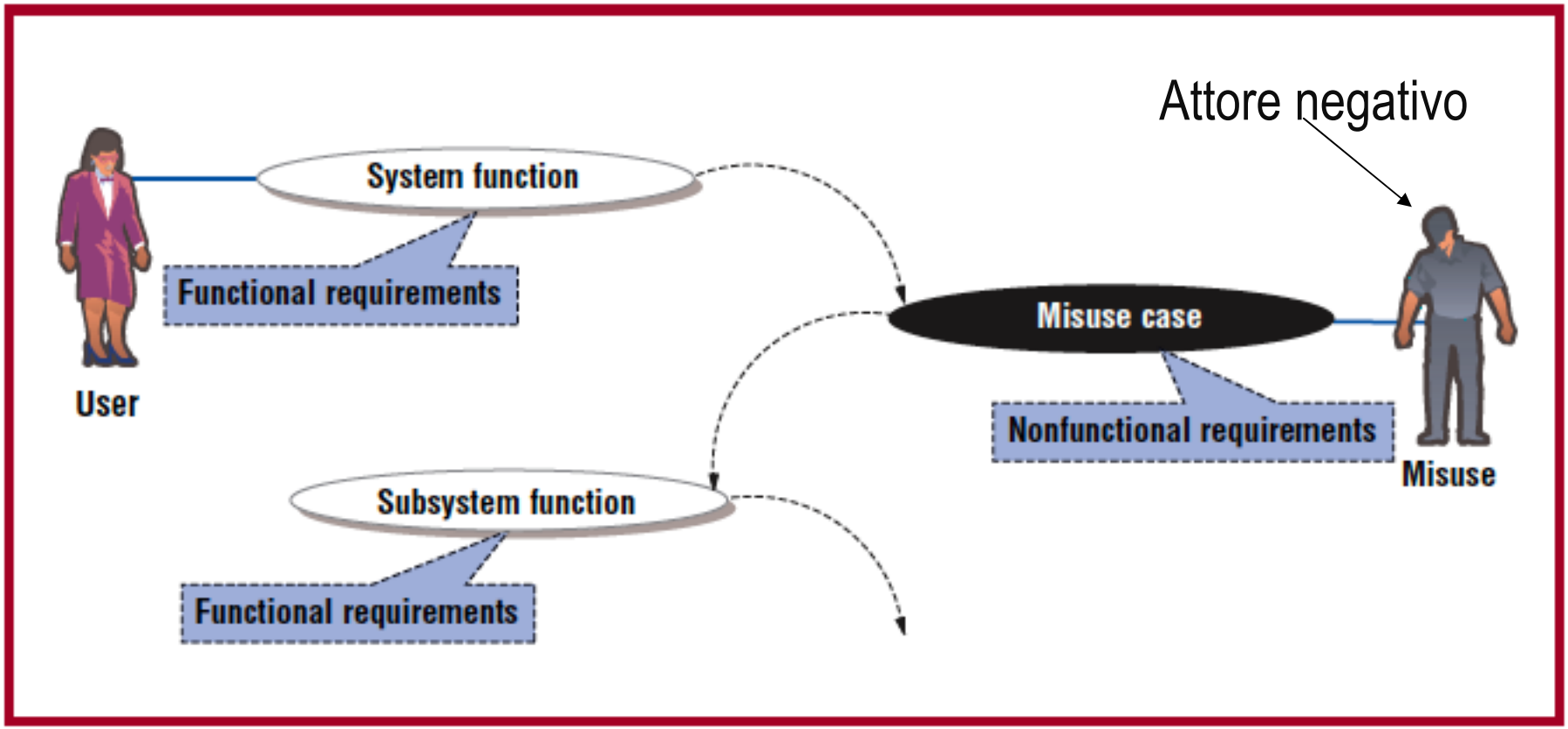
Livello: sotto-funzione

Scenario principale:...

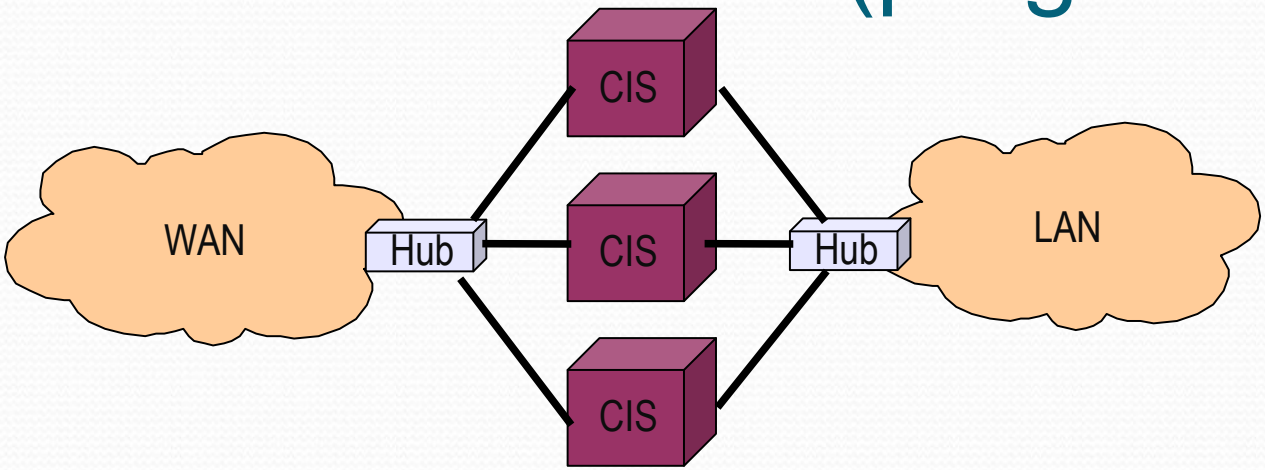
Misuse cases (I)

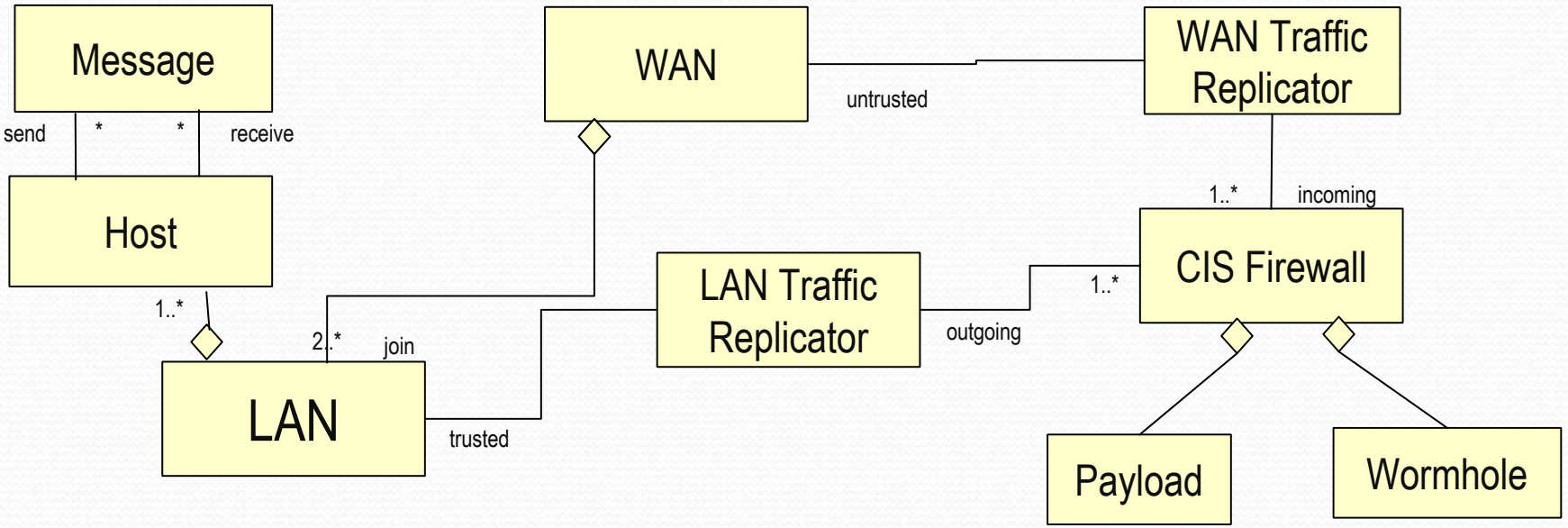
- Casi d'uso per catturare requisiti non funzionali
  - sicurezza
  - affidabilità
- Uso di stereotipi per identificare
  - attori negativi (hackers, agenti esterni che possono minacciare il sistema)
  - casi di utilizzo improprio del sistema
  - possibili contro-misure per mitigare gli effetti (requisiti funzionali derivati)

Misuse case (II)

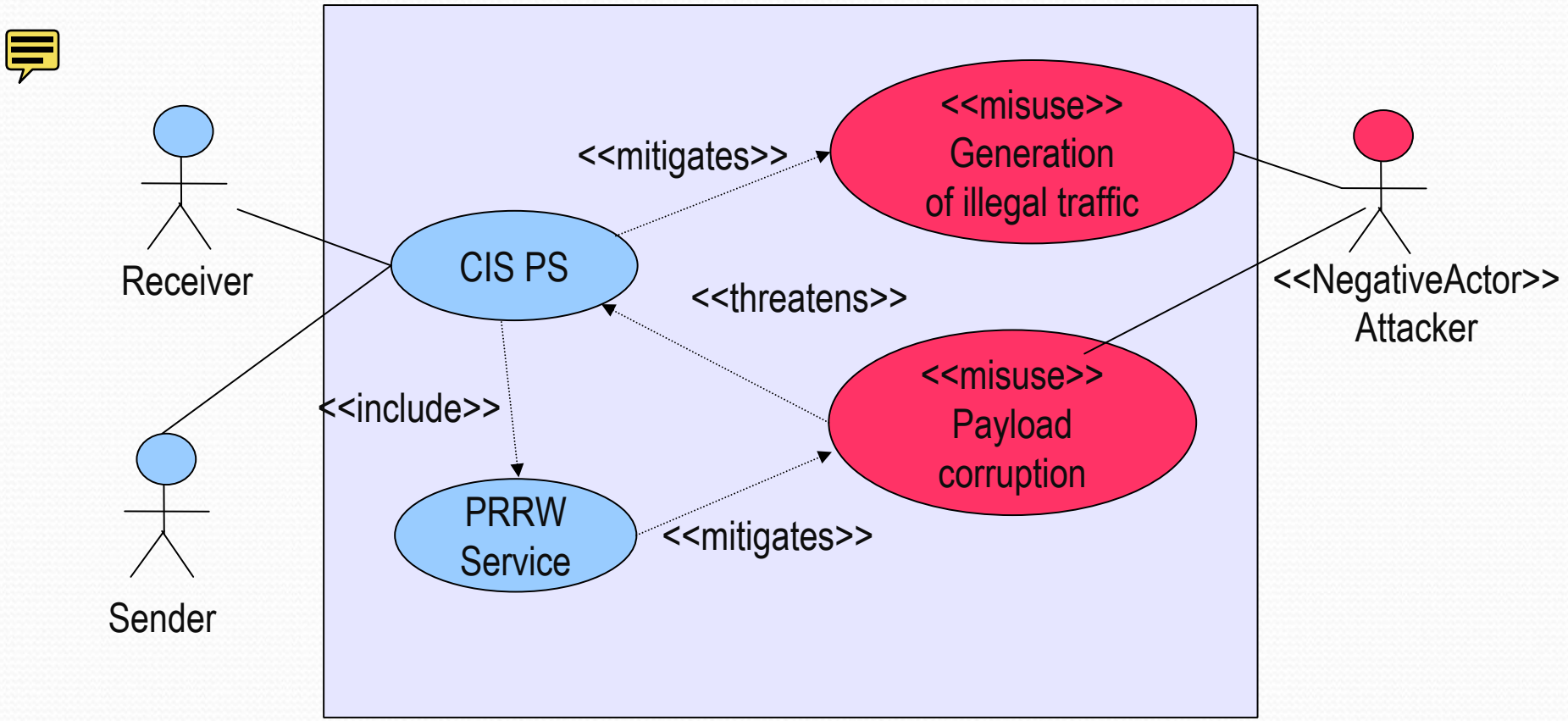


Caso di studio CIS (progetto CRUTIAL)





Esempio di misuse case



CIS PS: descrizione dettagliata

UC Name	CIS Protection Service
Scope	SCADA
Main Actors	Sender (computer from the WAN), Receiver (computer of the protected LAN)
Success guarantee	The correct message is eventually delivered The illegal message is not delivered
Main scenario	A message is sent by Sender to Receiver 1. It arrives to the CIS Firewall 2. Each CIS Firewall checks if it satisfies the security policy and votes 3. The CIS firewalls agree upon a final judgement (majority voting) 4. The message is correct and the CIS Firewall leader forwards it to the Receiver
Alternate scenarios	4.a The message is illegal, then it is not delivered
Special Reqs	A1. The CIS PS should be available 99.99% of the time R1. The MTBF shall be at least 6 months
Relationships	CIS includes PRRW Service, Payload Corruption threatens CIS PS, CIS PS mitigates Generation of illegal traffic

Payload Corruption: descrizione dettagliata

MUC Name	Payload Corruption
Scope	CIS PS
Main Actors	Attacker: Outside and Inside Threats
Success guarantee	The Payload evaluates as “correct” an illegal message or it evaluate as “illegal” a correct message (FM1), or it is subject to a temporary omission (FM2)
Main Scenario (Outside Threat)	The Attacker identifies the WAN traffic replicator as potential target • The Attacker sniffs the network traffic • The Attacker gets an unauthorized access to an host in the LAN • The Attacker install a malicious logics in the accessed host • The hosted Payload behaves in an unpredicted manner.
Special Reqs	F1. At most f Payloads can be concurrently corrupted F2. f should be se according to the expected rate of fault occurrence
Relationships	Payload Corruption threatens CIS PS