

Corso di Laurea in Ingegneria Informatica

Corso di Ingegneria del Software

**Il linguaggio UML
Aspetti dinamici**

Sommario

Interaction Diagram

Sequence Diagram

Collaboration (Communication) Diagram

State Diagram

Activity Diagram

Riferimenti:

J.Arlow, I. Neustadt: UML 2 e Unified Process, II edizione,
McGraw-Hill. **Cap. 12, 13, 19**

Diagrammi di interazione

diagrammi di sequenza

diagrammi di comunicazione

Diagrammi di interazione (1/2)

Per aggiungere la dinamica comportamentale agli oggetti individuati (istanze dei classificatori), UML introduce i **diagrammi di interazione** nei quali modellare gli scenari dei casi d’uso tramite l’interazione tra le entità (concettuali o di progetto) individuate.

I diagrammi di interazione mostrano come le istanze dei classificatori interagiscono tra loro per realizzare un comportamento del sistema.

La simulazione dei Casi d’Uso con messaggi può rivelare l’esigenza di ulteriori classi non già specificate nel Diagramma delle Classi.

Diagrammi di interazione (2/2)

UML definisce quattro diagrammi che permettono di descrivere le interazioni (messaggi) tra oggetti:

- ◆ Diagrammi di Sequenza (Sequence Diagram)

- Enfatizzano la tempificazione e l'ordine dei messaggi scambiati tra le istanze.

- ◆ Diagrammi di Comunicazione (Communication Diagram)

- Diagrammi di collaborazione in UML 1.0
- Enfatizzano le relazioni strutturali tra gli oggetti e permettono di visualizzare rapidamente la collaborazione di un oggetto all'interazione.

- ◆ Diagrammi di Riepilogo di Interazione (Interaction Overview Diagram)

- Sono un caso speciale di diagrammi di attività.
- Mostrano come una interazione complessa è realizzata tramite un sottoinsieme di interazioni semplici.

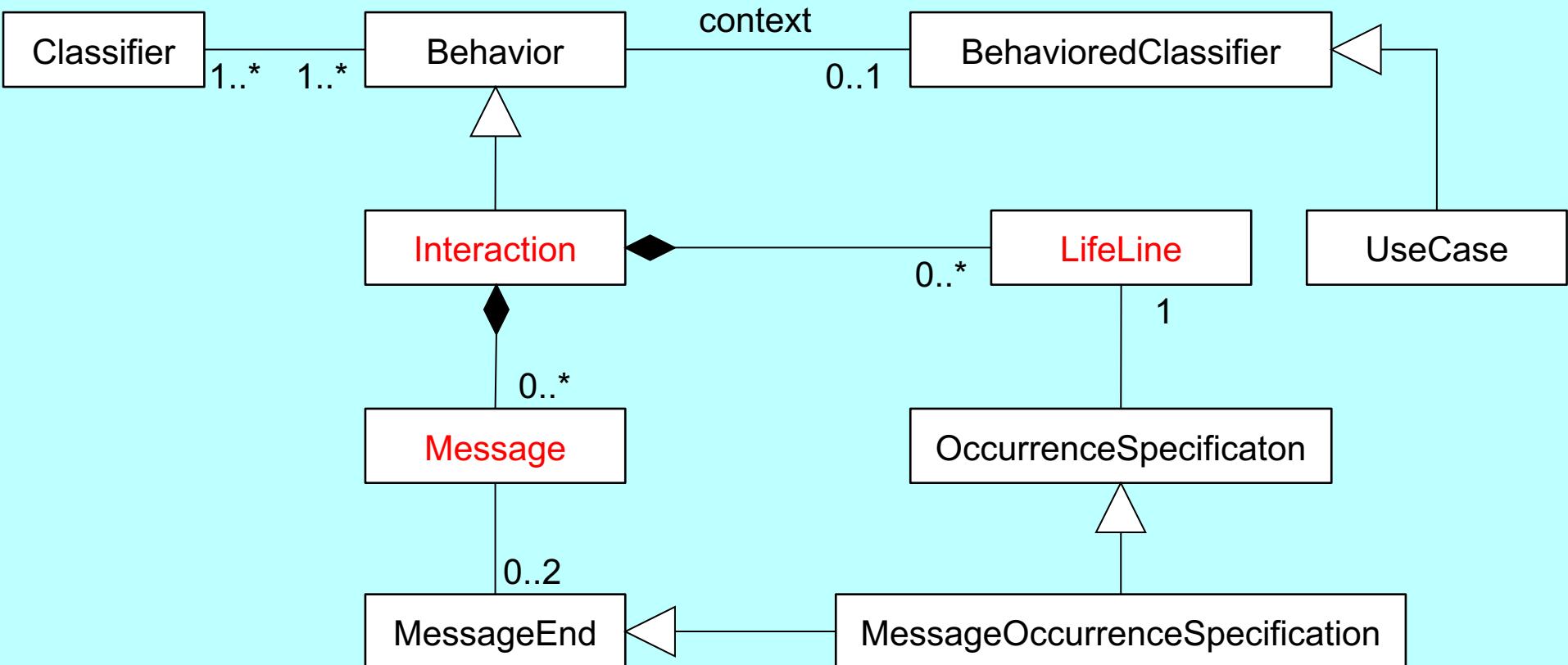
- ◆ Diagrammi di temporizzazione (Timing Diagrams)

- Pongono enfasi sui vincoli real-time cui le interazioni devono rispettare.

Interazioni

- ♦ Le interazioni sono unità di comportamento di un classificatore. Questo classificatore, conosciuto come **classificatore di contesto**, fornisce il contesto dell'interazione.
 - Le interazioni mostrano un scambio osservabile di informazione.
 - Ad esempio, se il classificatore di contesto è uno use case, una interazione mostra come il comportamento specificato dallo use case è realizzato dalle istanze dei classificatori (in questo caso, le classi di analisi) che si inviano messaggi l'un l'altro.
- ♦ I diagrammi di interazione catturano una interazione tramite:
 - LifeLine – mostrano i partecipanti in una interazione
 - Messaggi – mostrano la comunicazione tra i partecipanti

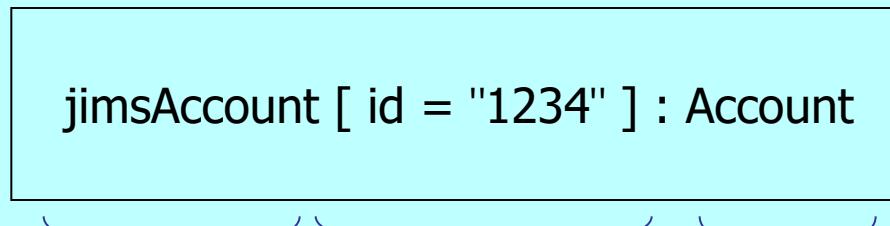
Metamodello UML diagrammi interazione



Le relazioni tra le Interazioni, i LifeLine, i Messaggi e il Classificatore di contesto sono mostrate da questo diagramma delle classi parziale del metamodello UML 2.0

Lifeline

- ❖ Una lifeline rappresenta un singolo partecipante in una interazione.
 - Permette di rappresentare come una istanza di un classificatore partecipa in una interazione



Nome

Usato per riferirsi alla
lifeline nell'interazione

Selettore

Una condizione booleana che
seleziona una specifica istanza

Tipo

Il classificatore di cui la lifeline
ne rappresenta una istanza

Le lifeline devono essere univocamente identificabili all'interno
di una interazione tramite il nome, il tipo o entrambi.

Messaggi (1/2)

- ♦ Un messaggio definisce una particolare comunicazione tra le lifeline in una interazione.
- ♦ La comunicazione può:
 - Invocare una operazione – messaggio di chiamata
 - Creare o distruggere una istanza – messaggio di creazione/distruzione
 - Inviare un segnale

Messaggi (2/2)

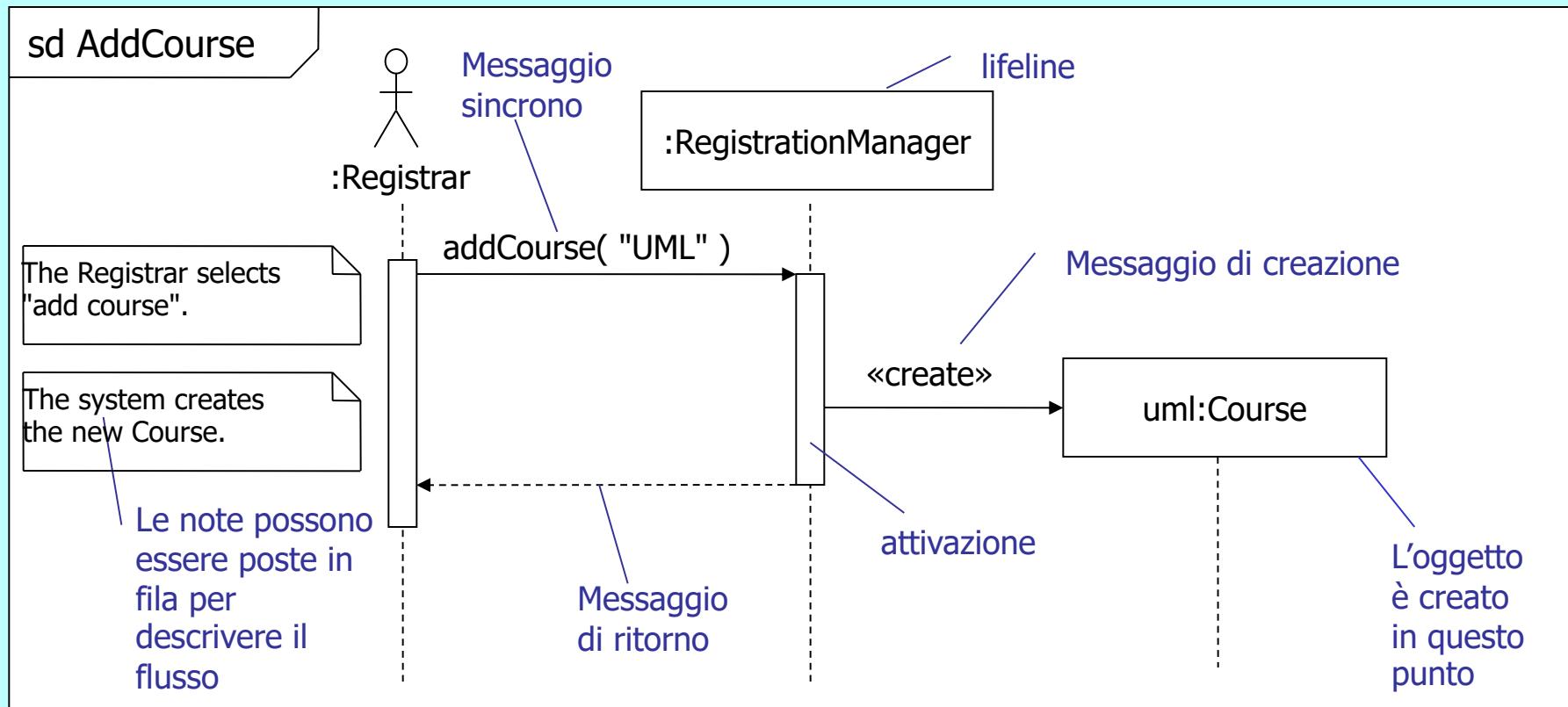
- ♦ Quando una lifeline riceve un messaggio di chiamata, ci deve essere una corrispondente operazione nel classificatore della lifeline.
 - UML permette in primo luogo (fase di analisi) di rilassare questo vincolo per avere modelli più dinamici e flessibili
 - I messaggi possono invocare operazioni in maniera sincrona e asincrona
- ♦ Una lifeline si attiva quando esegue un messaggio. Col procedere del tempo l'attivazione si sposta tra le lifeline descrivendo il flusso di controllo

Tipi di messaggi

Rappresentazione Sender	Receiver	Tipo	Semantica
aMessage(aParameter)		Messaggio Sincrono	Il chiamante attende che il ricevente completi
aMessage(aParameter)		Messaggio Asincrono	Il chiamante invia un segnale al ricevente e non attende il completamento
		Messaggio di ritorno	Il ricevente completa una chiamata sincrona e restituisce il controllo al chiamante
<<create>> ctor()		Creazione	Il chiamante crea il target (invocando, ad esempio, un costruttore)
<<destroy>>		Distruzione	Il chiamante distrugge il ricevente
		Messaggio trovato	Il messaggio è inviato esternamente all'ambito dell'interazione
		Messaggio perso	Il messaggio falisce nel raggiungere la destinazione

Diagrammi di sequenza (1/2)

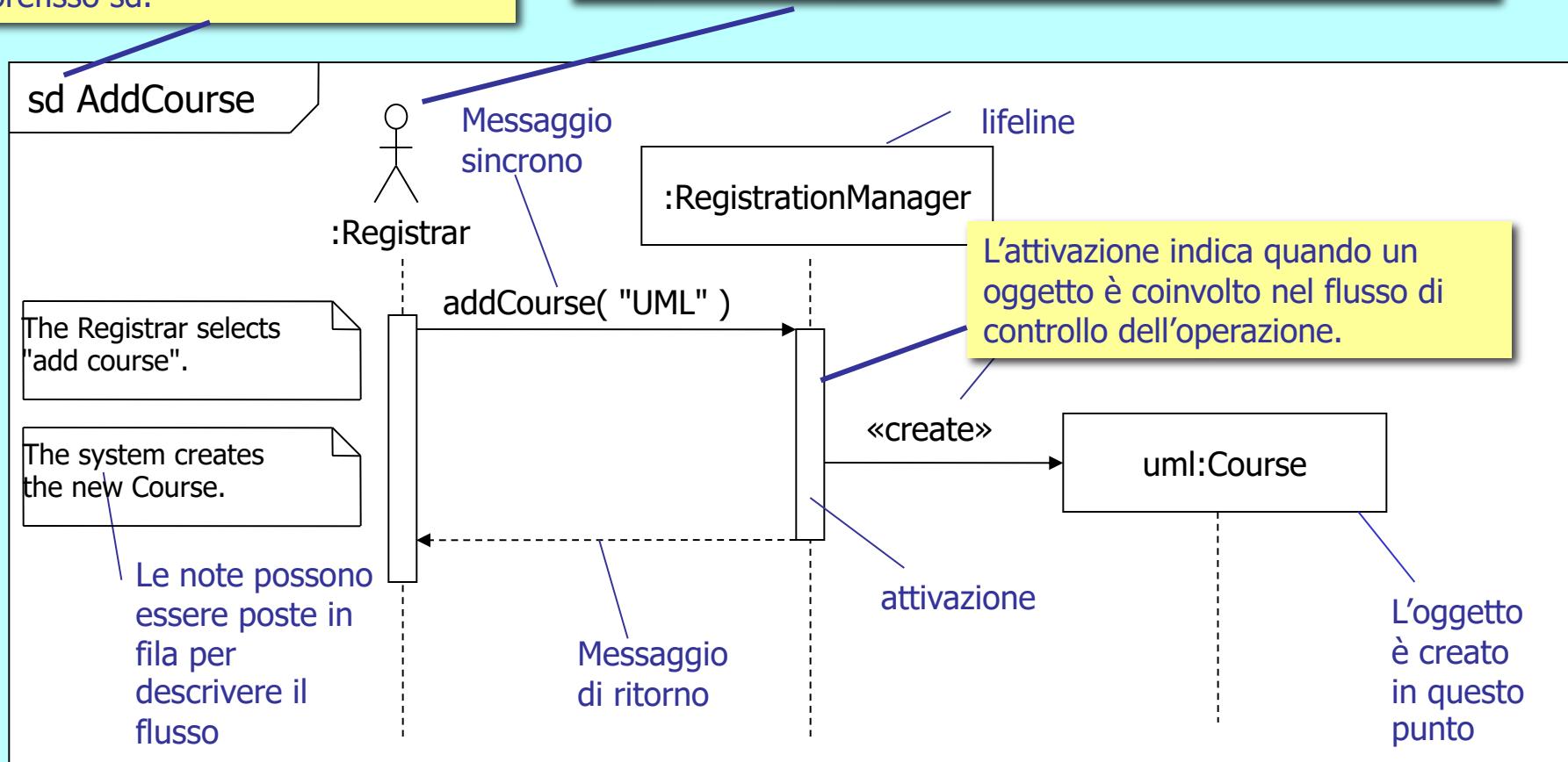
- ◆ Mostrano le interazioni tra le lifeline come una sequenza ordinata temporalmente di eventi.



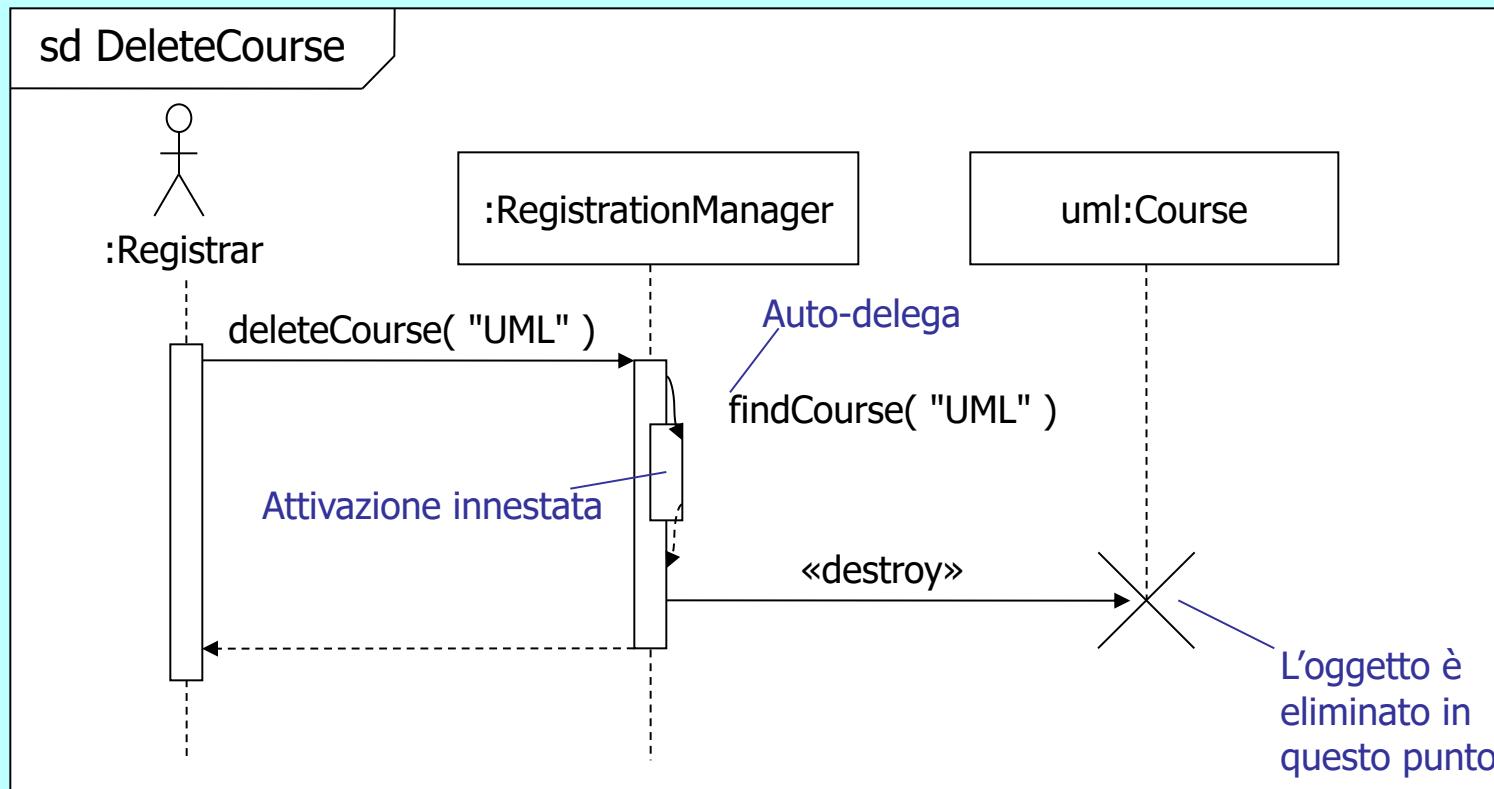
Diagrammi di sequenza (1/2)

▲ Mostrano le interazioni
Tutti i diagrammi di interazione
(non solo i sequence) hanno
prefisso sd.

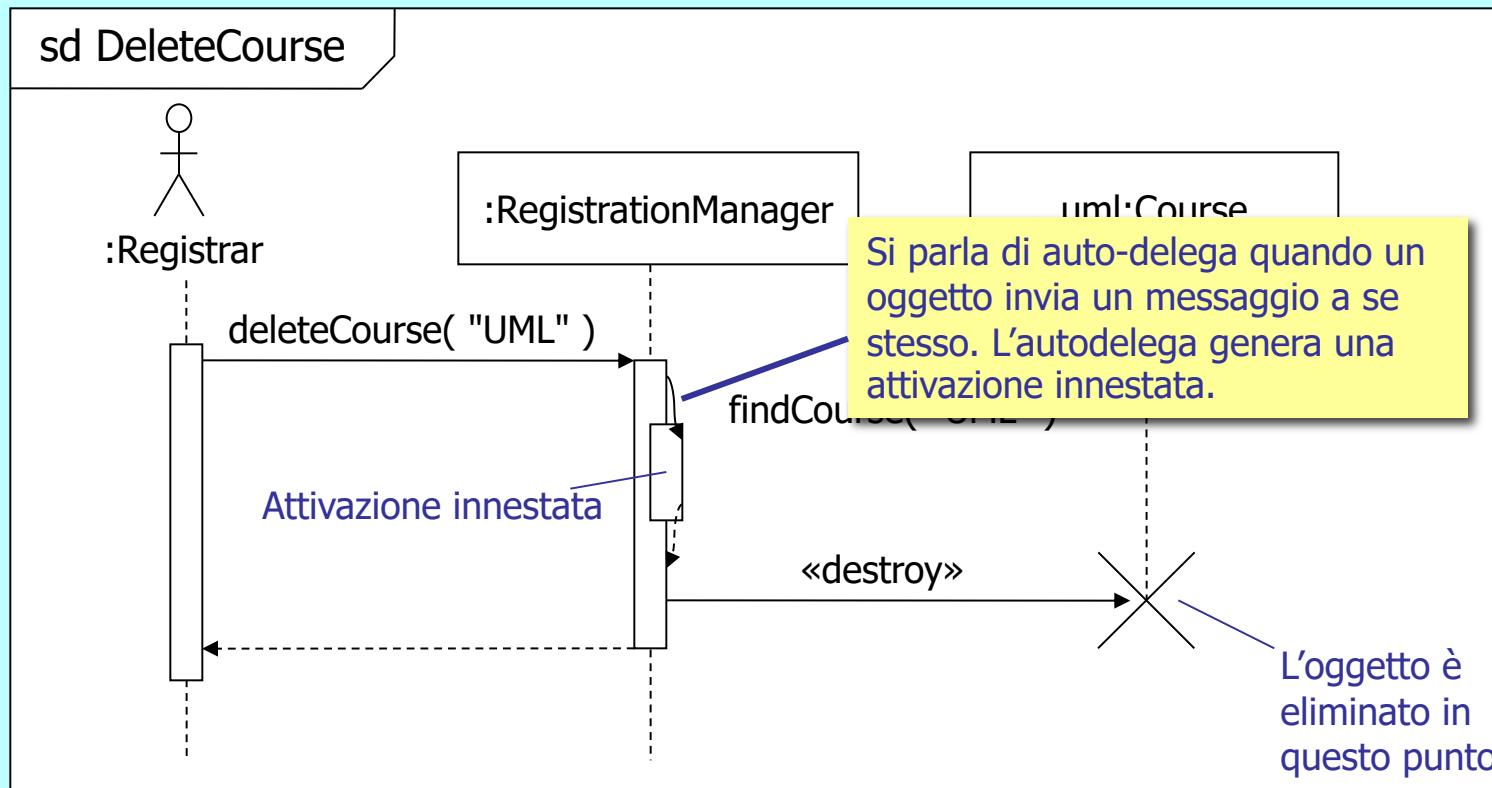
Gli attori possono partecipare all'interazione. In sede di progettazione i diagrammi di interazione possono essere arricchiti specificando gli elementi dell'interfaccia grafica con cui gli attori interagiscono.



Diagrammi di sequenza (2/2)

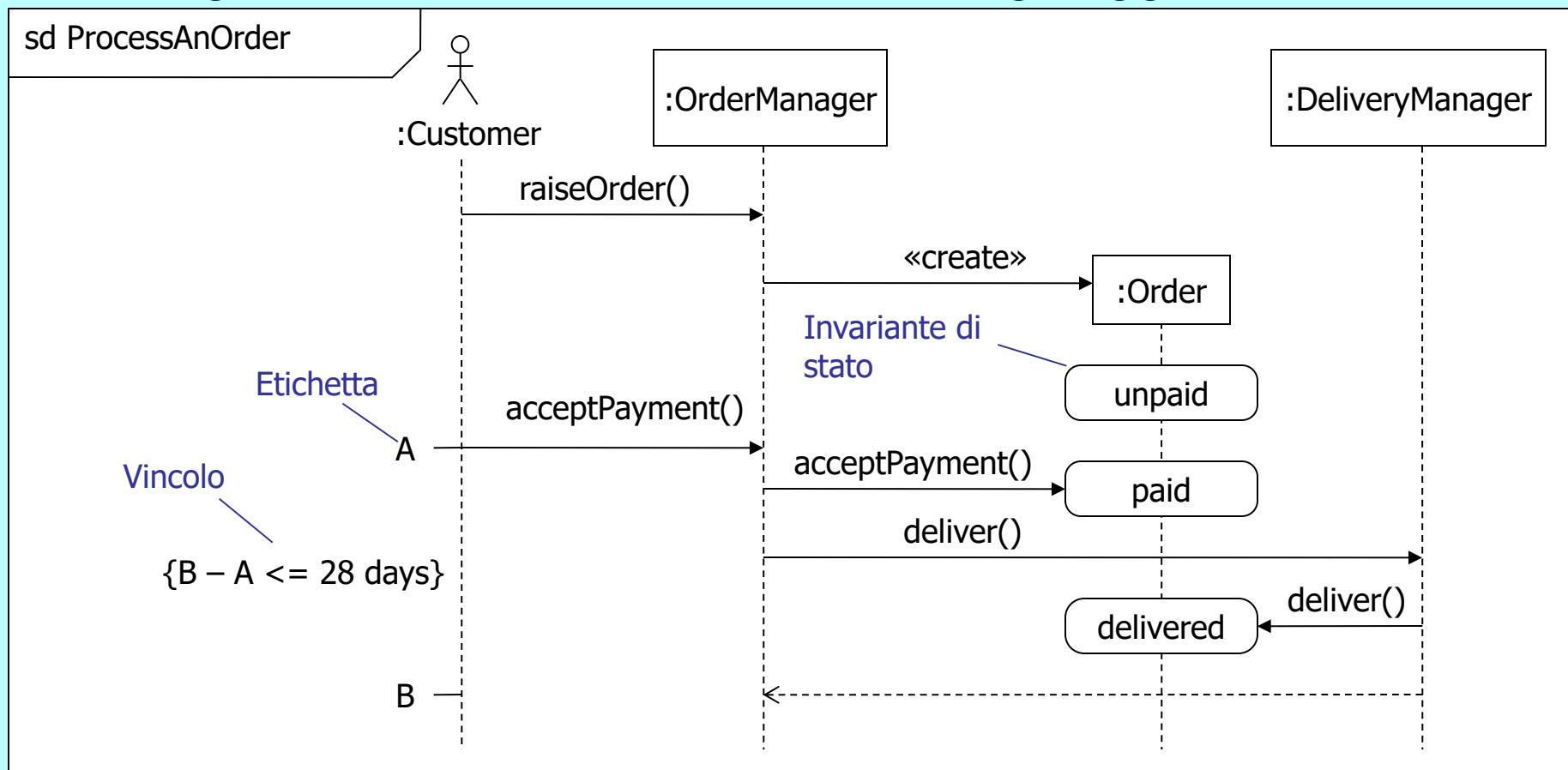


Diagrammi di sequenza (2/2)



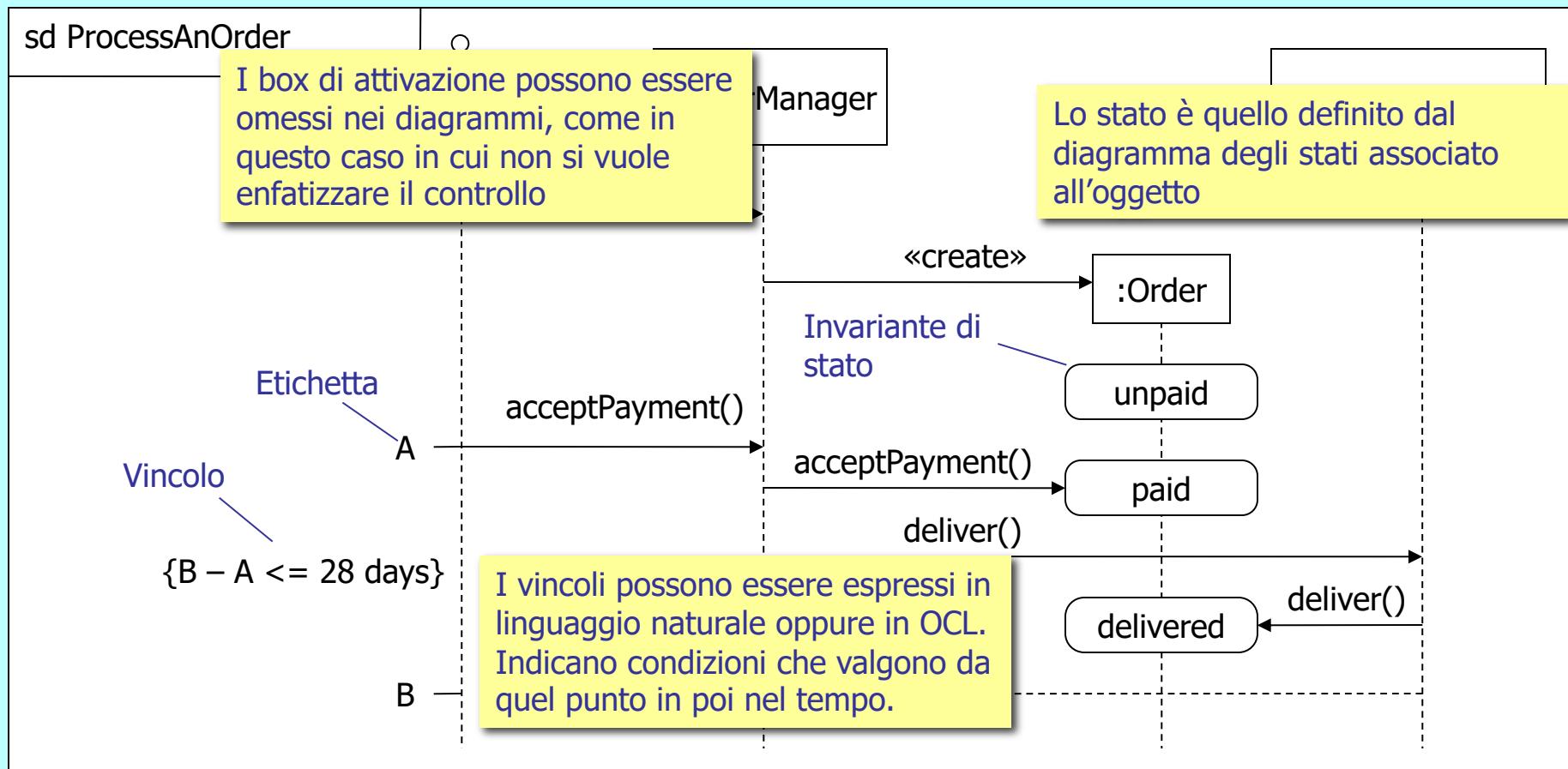
Diagrammi Sequenza – Aspetti avanzati

- UML permette di arrichire gli SD specificando vincoli nei diagrammi e invarianti sullo stato degli oggetti.



Diagrammi Sequenza – Aspetti avanzati

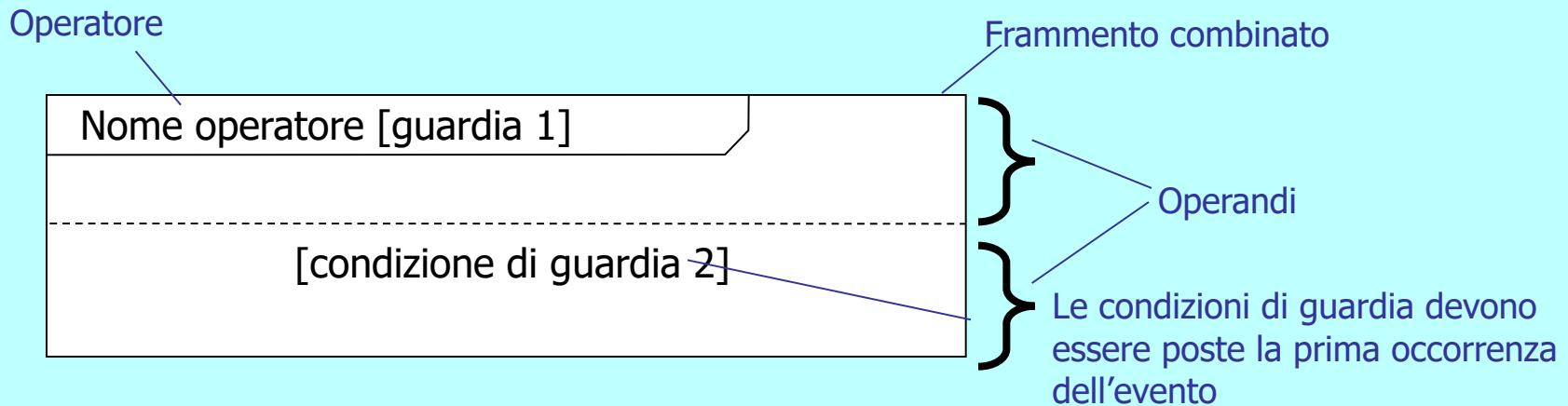
- UML permette di arrichire gli SD specificando vincoli nei diagrammi e invarianti sullo stato degli oggetti.



Operatori e frammenti combinati

- ♦ I SD possono essere divisi in *frammenti combinati*. I frammenti combinati hanno:

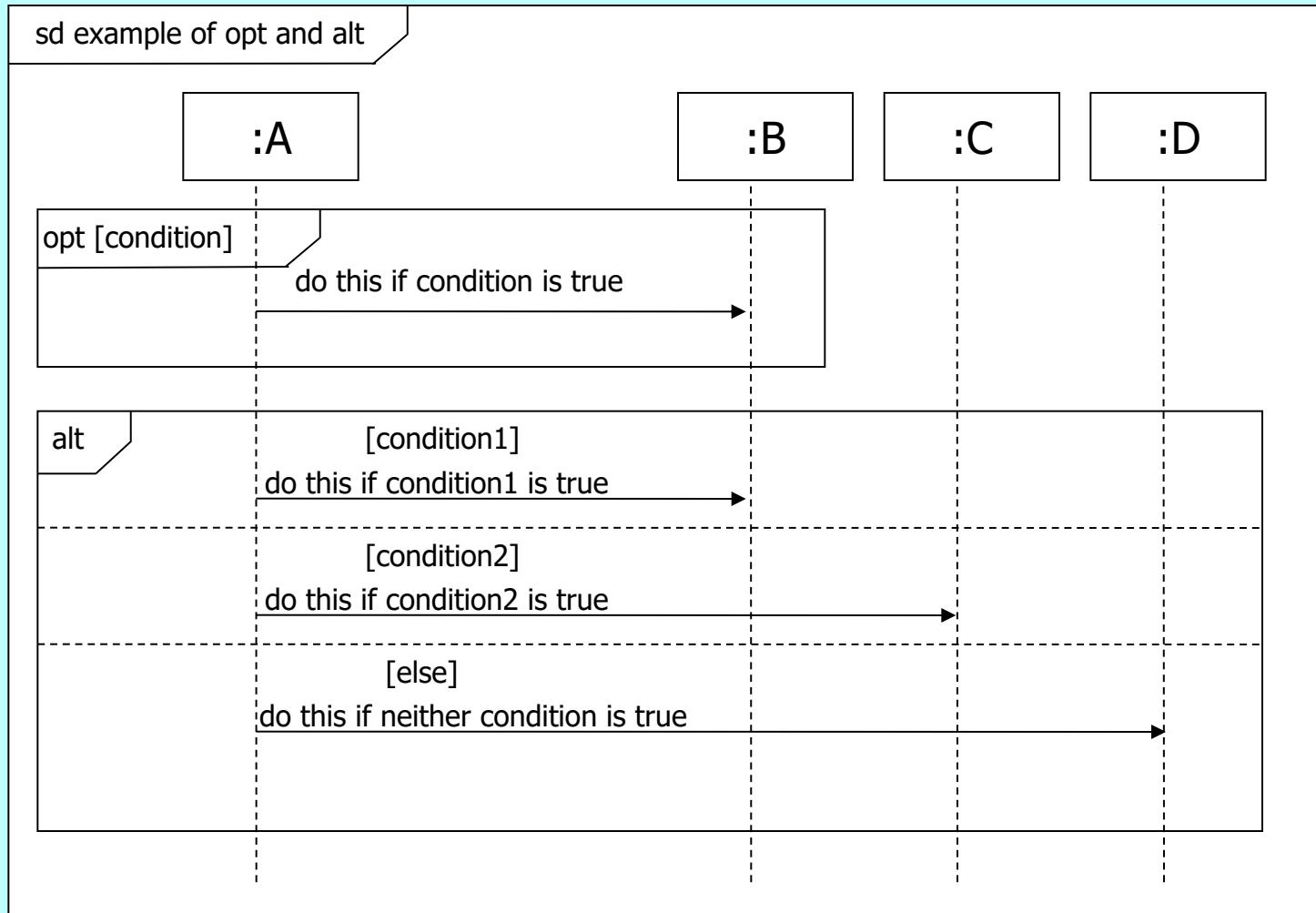
- Un *operatore*
 - ♦ determina come vengono eseguiti i suoi operandi
- Uno o più operandi
- Zero o più condizioni di guardia
 - ♦ Stabiliscono se i loro operandi devono essere seguiti.
 - ♦ Ogni operatore può avere la propria guardia o c'è ne una per tutti.



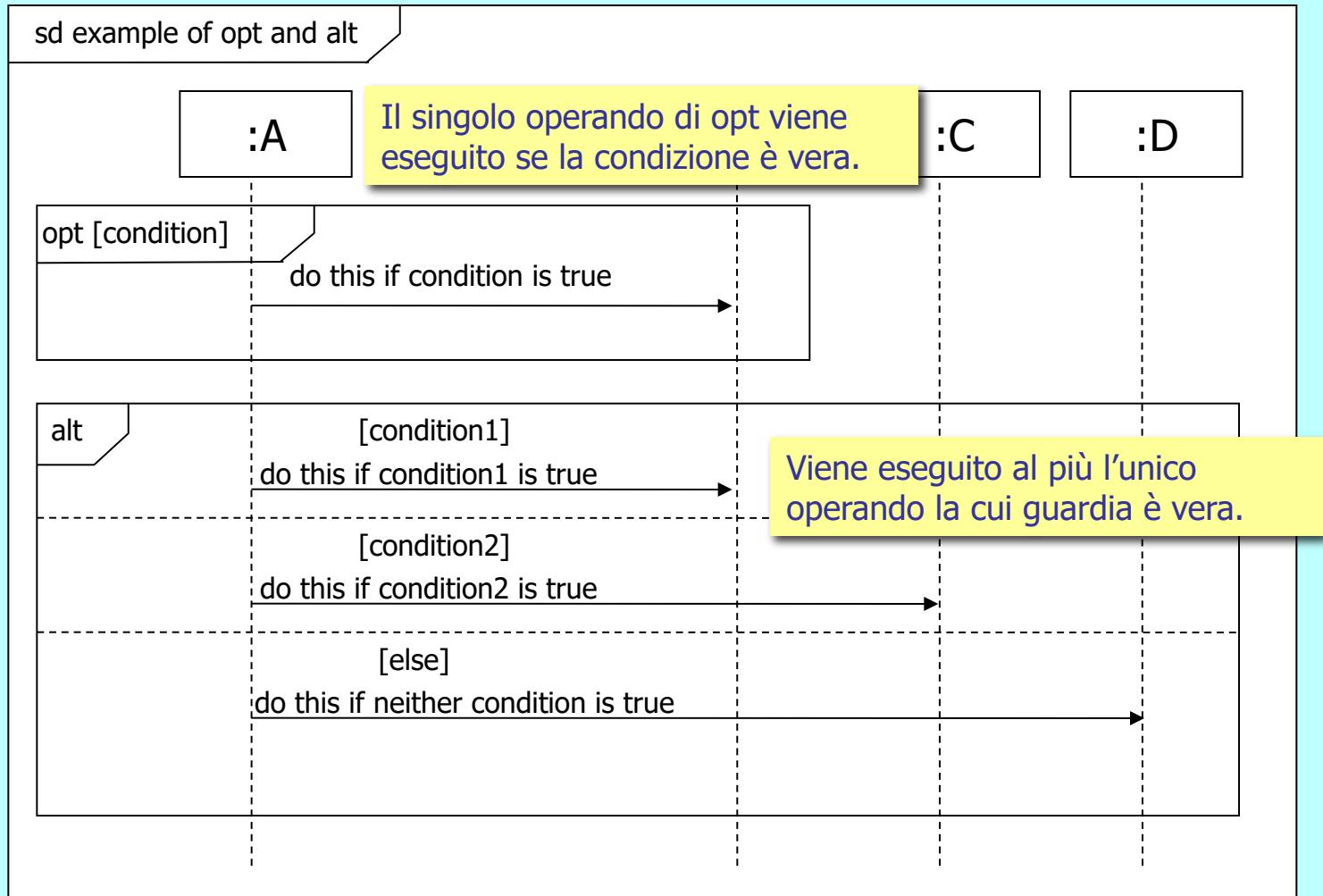
Operatori di frammento più usati

Operatore	Nome completo	Semantica
opt	Option	C'è un singolo operando che viene eseguito se la sua condizione è vera (if then... endif)
alt	Alternatives	Viene eseguito l'operando la cui condizione è vera. Ammette else come condizione di guardia. (if then ... elseif ... else ... endif)
loop	Loop	Sintassi: loop min, max [condizione] Esegue loop min volte. Poi, se e fin quando [condizione] è vera, continua ad iterare fino a max.
break	Break	Se la guardia è vera l'operando è eseguito e si interrompe il resto dell'interazione del frammento che lo racchiude.
ref	Reference	Il frammento combinato rimanda ad un'altra interazione

Frammenti combinati: opt e alt



Frammenti combinati: opt e alt

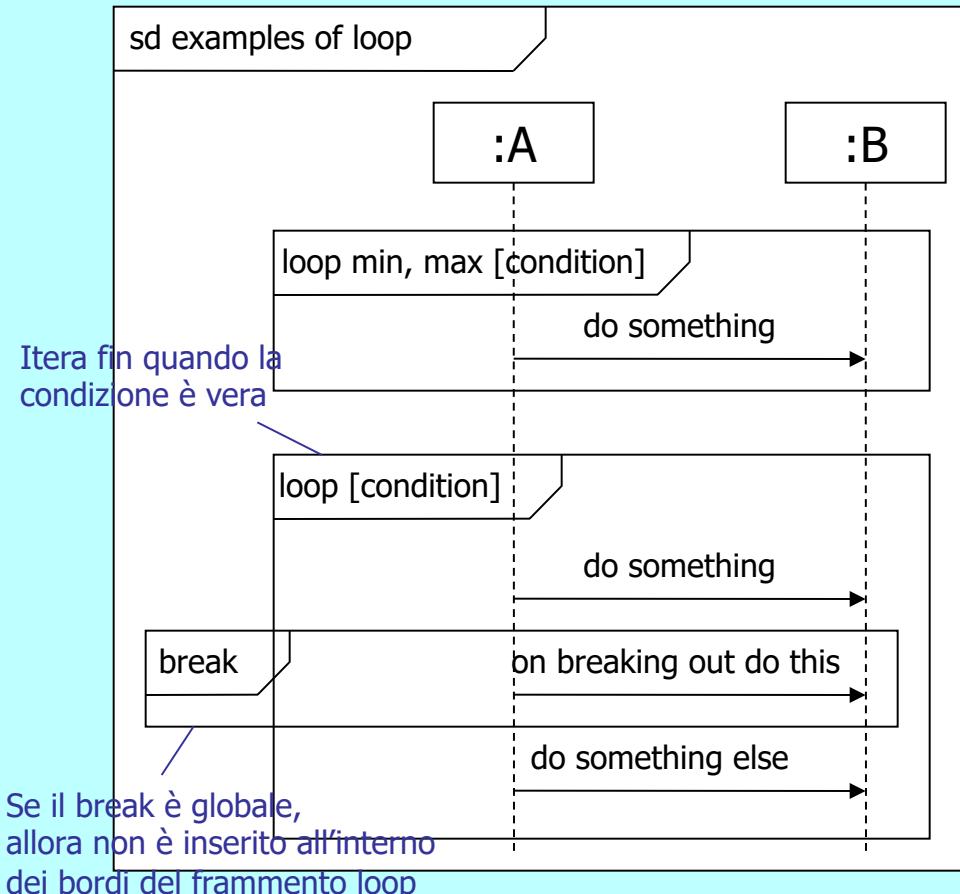


Frammenti combinati: loop e break

Loop itera min volte e poi fino a (max-min) fin quando la condizione è vera.

Costrutto	Sintassi
while (true) {}	loop / loop *
for (i=0;i<n;i++) {}	loop n
while (G) {}	loop [G]
do { ... } while (G);	loop 1,* [G]
foreach (e in V) {}	loop [for each e in V]
foreach (instance of Class) {}	loop [for each instance:Class]

Poiché il frammento break interrompe il loop è esterno ai suoi bordi.

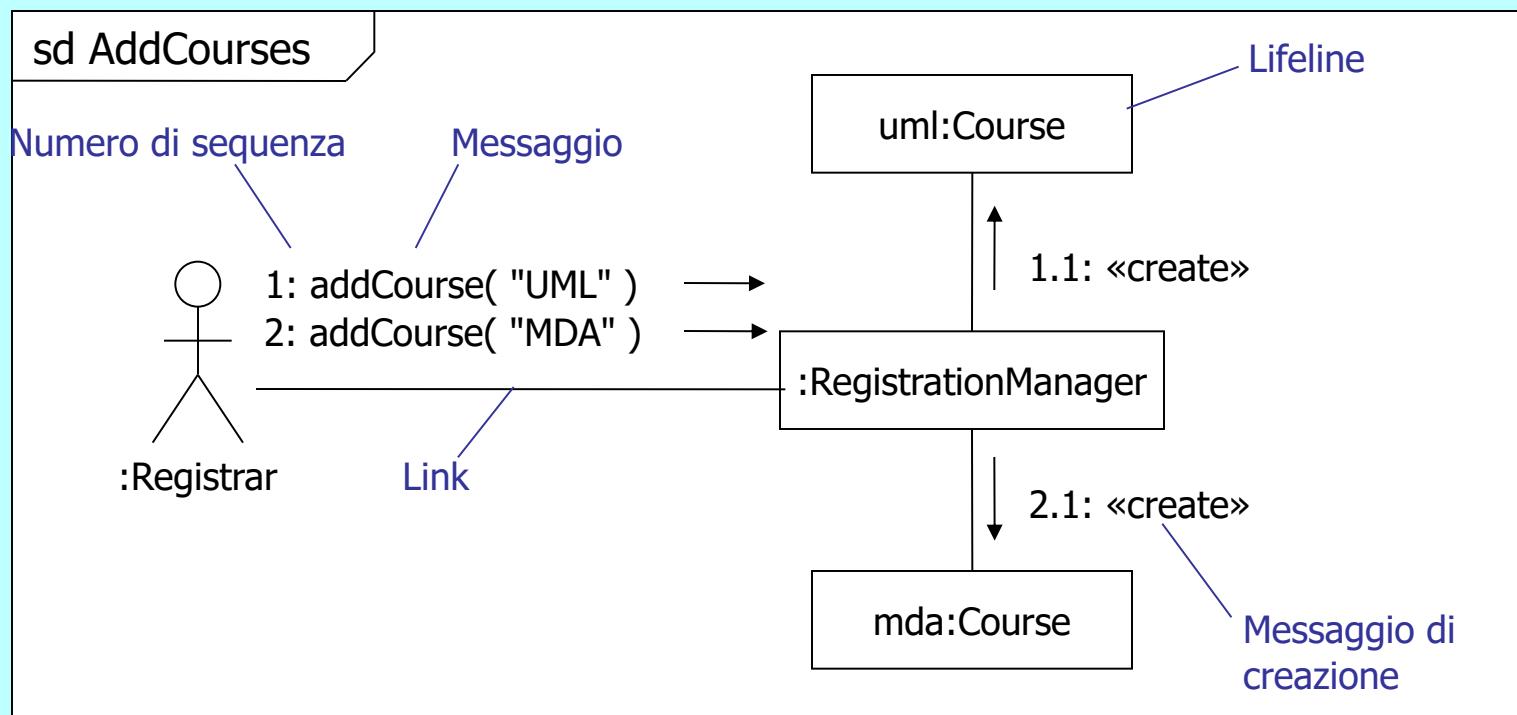


Diagrammi di comunicazione

- ♦ I diagrammi di comunicazione enfatizzano gli aspetti strutturali di un'interazione, mostrando come si collegano le LifeLine.
 - Possono considerarsi casi particolari di diagrammi degli oggetti
- ♦ La differenza di sintassi è che i diagrammi di collaborazione non hanno le *code* ma le lifeline sono connesse da collegamenti
 - I collegamenti forniscono i canali di comunicazione per i messaggi
 - La sequenza temporale è indicata dalla numerazione gerarchica di ogni messaggio

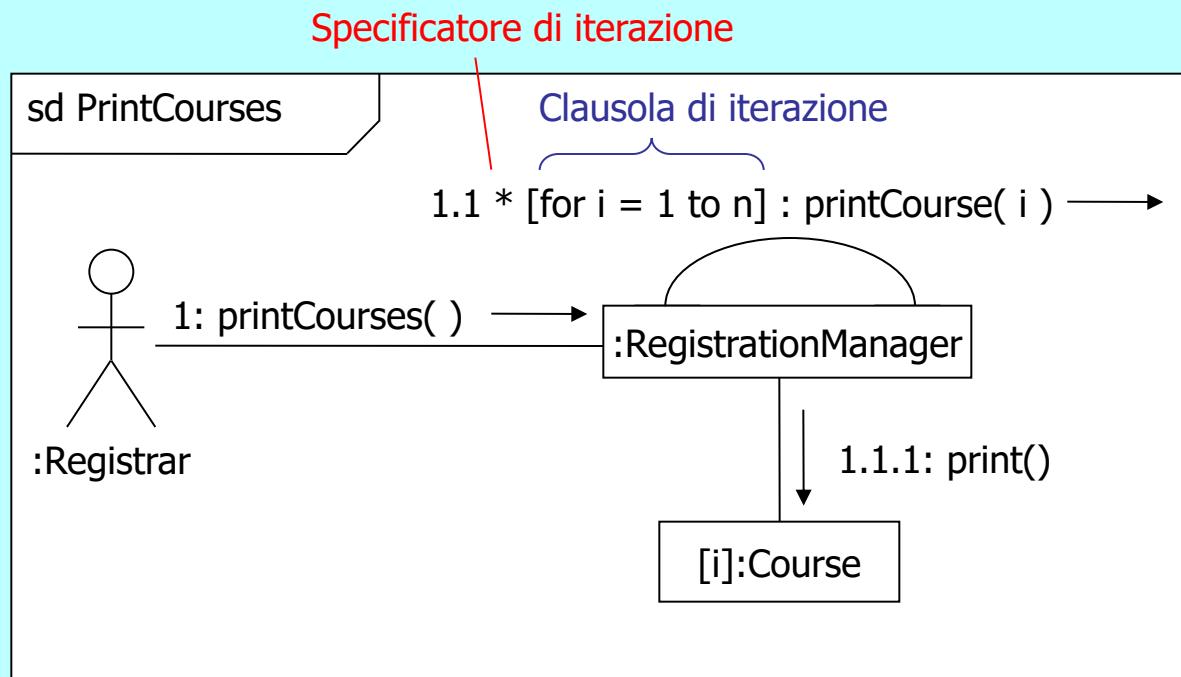
Diagrammi di comunicazione

- Il tempo non è una dimensione separata, la sequenza dei messaggi deve essere determinata utilizzando i numeri di sequenza.



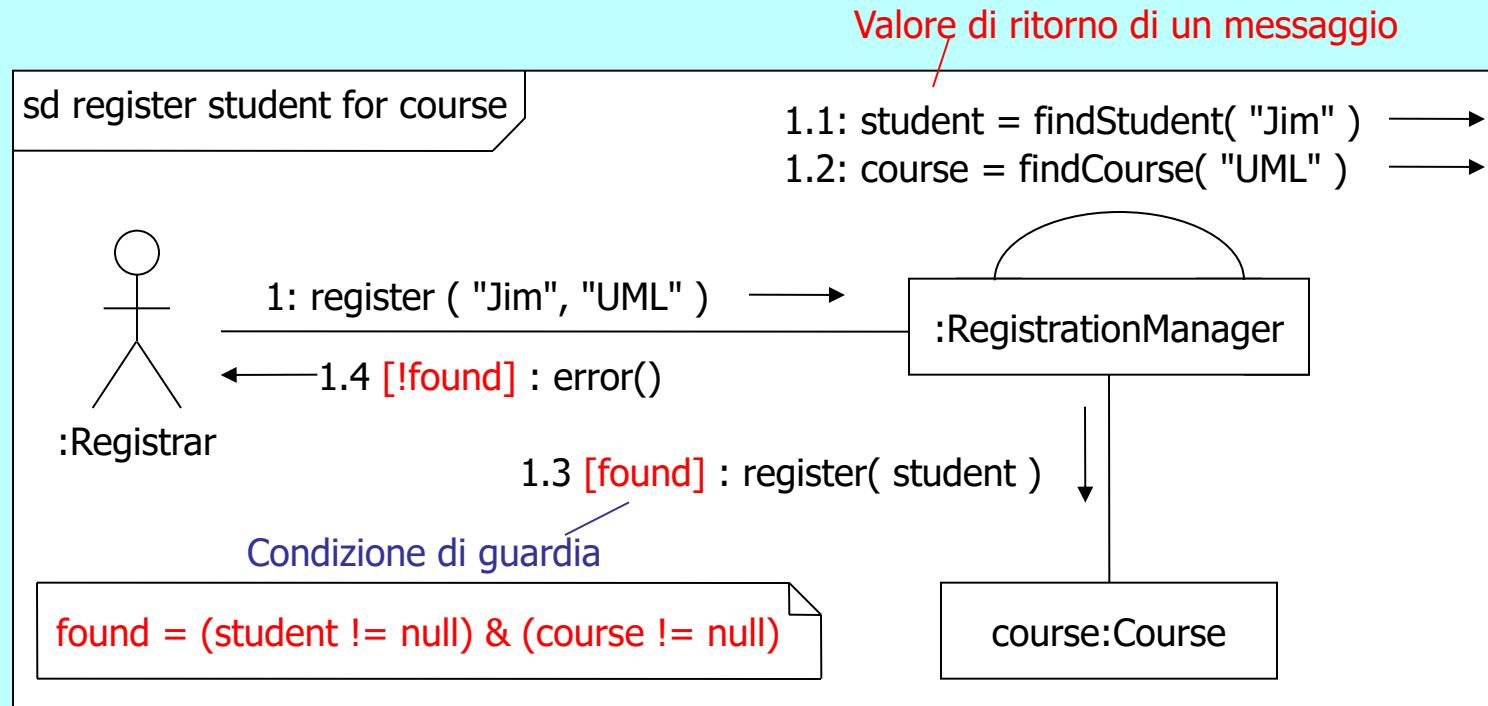
Diagrammi di comunicazione: Iterazione

- ◆ L'iterazione è mostrata usando uno specificatore di iterazione e una clausola opzionale
- ◆ Lo specificatore di iterazione “*” indica che i messaggi sono inviati sequenzialmente. Si usi “*//” per rappresentare un invio in parallelo.



Diagrammi di comunicazione: Ramificazione

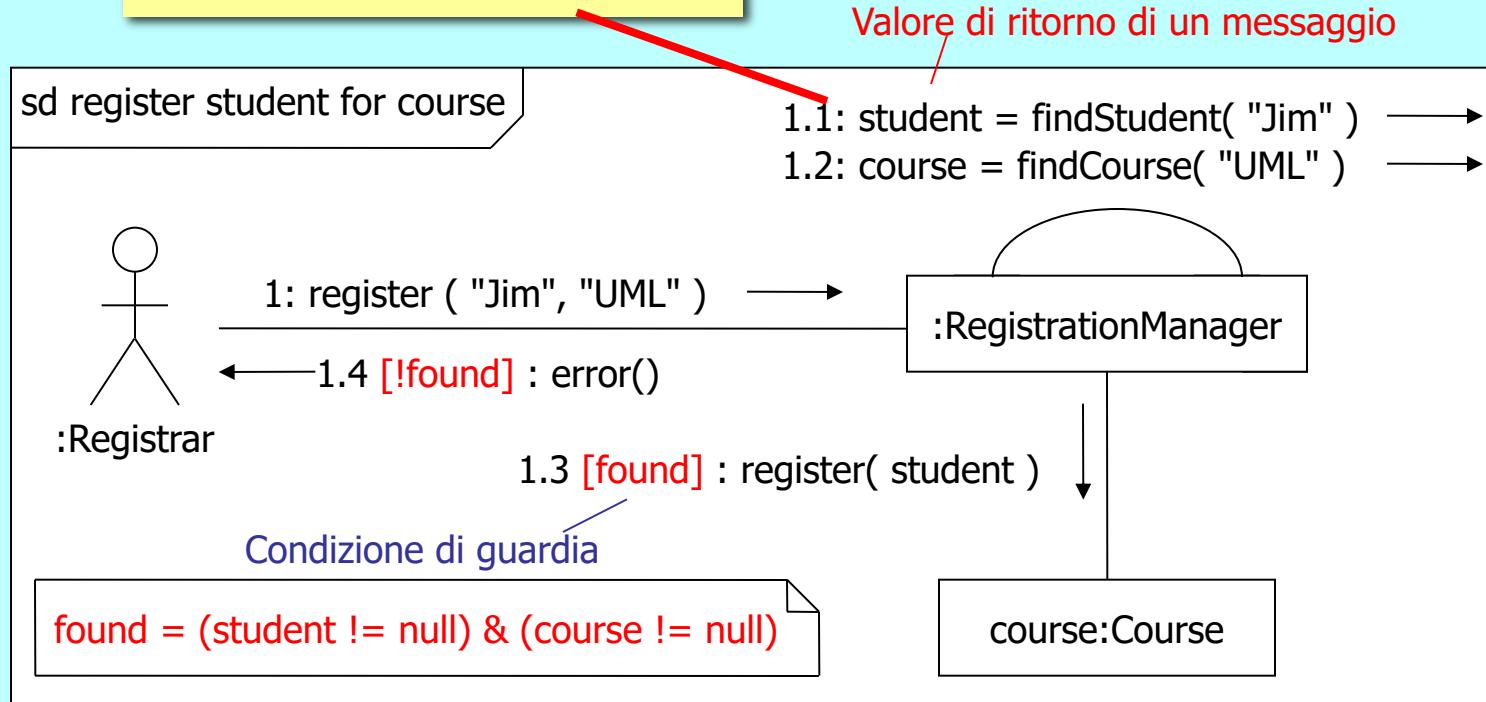
- La ramificazione è modellata aggiungendo condizioni di guardia ai messaggi. Il messaggio è inviato solo se la condizione di guardia è vera.
 - Poiché le ramificazioni rendono difficile la lettura sono da limitare a casi semplici.



Diagrammi di comunicazione: Ramificazione

- La ramificazione è modellata aggiungendo condizioni di guardia ai messaggi. Il messaggio è inviato solo se la condizione di guardia è vera.

- Poiché Nei diagrammi di interazione i messaggi possono assegnare il valore di ritorno (in generale i parametri di (in)out) a degli attributi con la notazione mostrata la lettura sono da limitare a casi semplici.



Confronto tra diagrammi di interazione

I Sequence Diagram e i Collaboration Diagram esprimono informazioni simili ma in modo differente.

♦ I Diagrammi di sequenza

- Enfatizzano la sequenza temporale relativa dei messaggi
- Sono i diagrammi di interazione più espressivi
- Sono indicati per specificare gli scenari di realizzazione dei casi d'uso.

♦ Diagrammi di comunicazione

- Permettono di rendere esplicito lo scambio di messaggi sulla base delle relazioni strutturali
- Mostrano le interazioni tra gli oggetti e sono più indicati per evidenziare tutti gli effetti su un dato oggetto e per la progettazione procedurale

Come scegliere fra sequence o collaboration diagram?

I **Sequence diagrams** rendono esplicito l' ordinamento temporale delle interazioni. Hanno analogie con gli scenari degli *use case*, perciò i sequence diagrams sono la scelta naturale quando si dettaglia un modello dell' interazione a partire da uno *use case*.

I **Collaboration diagrams** hanno meno spazio per dettagli. Sono preferibili quando si deriva un diagramma di interazione da un *class diagram*. Sono utili per la realizzazione iniziale dei casi d' uso e validare class diagrams.

Diagrammi di stato

State Diagram

- ◆ Lo **state diagram** di un oggetto rappresenta le sequenze di stati, le risposte e le azioni, che l' oggetto attraversa durante la sua vita in risposta agli stimoli ricevuti.
- ◆ Gli **state diagrams** permettono di rappresentare il comportamento dinamico di **macchine a stati UML**.
- ◆ Una macchina a stati UML modella il ciclo di vita di una **entità reattiva** come una macchina a stati finiti. Un entità reattiva:
 - Risponde a eventi esterni (eventi che nascono fuori dal contesto dell'entità)
 - Può generare e rispondere a eventi interni
 - Ha un ciclo di vita definito modellabile con una successione di stati, transizioni ed eventi
 - Il comportamento può dipendere dalla propria storia

Macchine a stati del comportamento e del protocollo

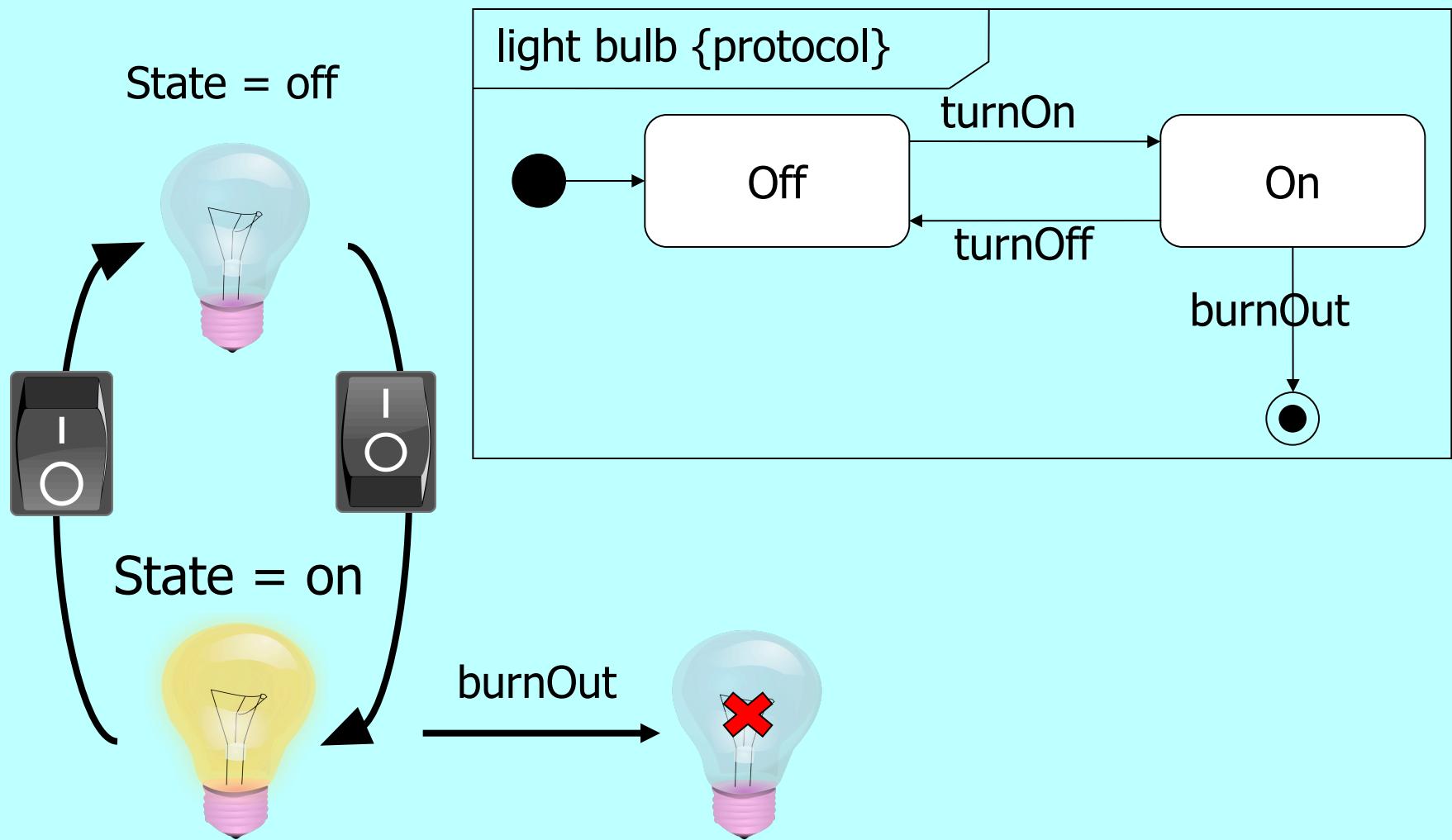
- ◆ UML prevede due tipi di macchine a stati
 - Macchine a stati **del comportamento**
 - ◆ Utilizzano stati, transizioni ed eventi per definire il *comportamento* dell'entità reattiva modellata (white-box)
 - Macchine a stati **del protocollo**
 - ◆ Utilizzano stati, transizioni ed eventi per definire il *protocollo* dell'entità reattiva modellata (black-box)
 - ◆ Possono descrivere anche entità che non abbiano implementazione

Diagramma di stato

Gli oggetti, istanze di una classe, sono assimilati a entità reattive a stati finiti.

- ◆ Le risposte a sollecitazioni esterne non dipendono soltanto dalla particolare sollecitazione, ma anche dalla *storia dell'oggetto*.
- ◆ La storia è sintetizzata nello stato in cui l' oggetto si trova: i valori correntemente assunti dagli attributi e dalle relazioni dell' oggetto sono la sintesi della vita dell' oggetto nel sistema.
- ◆ Rappresentare il comportamento interno di un oggetto (cosa succede all' invocazione dei suoi metodi) significa
 - esplicitare gli stati che un oggetto può attraversare
 - gli eventi che generano la transizione tra stati
 - le attività svolte in corrispondenza dei singoli stati

Diagramma di stato: esempio



Stati

- ♦ Uno **stato** di una macchina del **comportamento** può contenere zero o più azioni **e attività** che l' oggetto deve svolgere quando si trova in quello stato.
 - Una macchina a stati del protocollo non ha azioni né attività
- ♦ Ogni azione è associata a una transizione interna attivata da un evento
- ♦ Una **transizione interna** modella un evento che produce effetti senza alterare lo stato. Esistono due transizioni speciali di **ingresso** e **uscita**.



sintassi dell'azione: nomeEvento/ azione

sintassi dell'attività: esegui/ attività

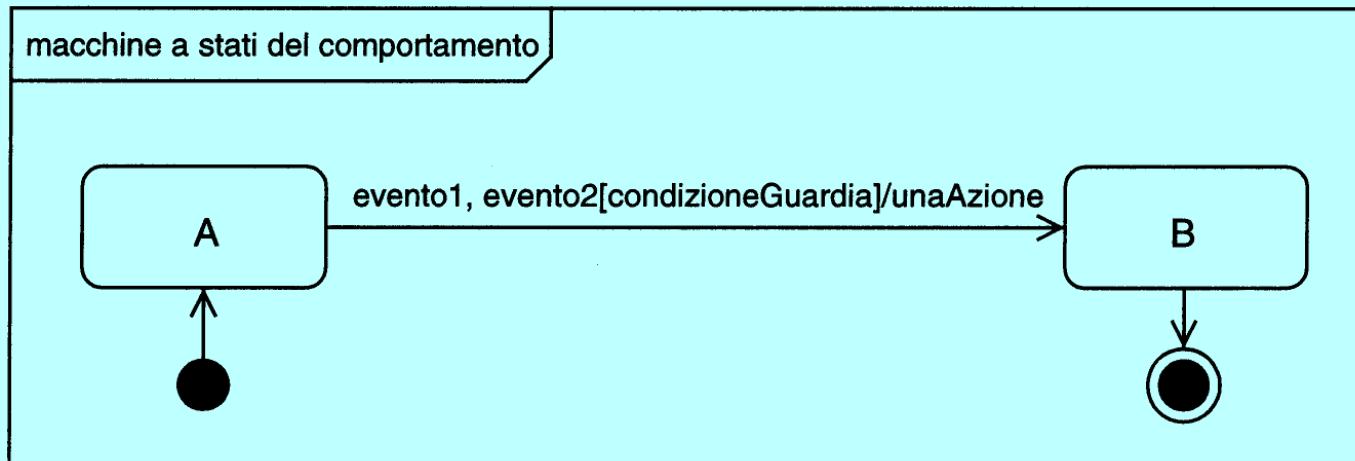
Azioni e attività

- ◆ La differenza semantica tra **azione** ed **attività** è correlata alla differente durata temporale delle due elaborazioni associate
- ◆ **L'azione** è associata ad un'elaborazione "veloce" (istantanea in relazione al tipo di sistema) ed atomica cioè **non interrompibile**
- ◆ **L'attività** è invece associata ad un'elaborazione più "complessa" che potrà essere interrotta da un evento generatosi al verificarsi di una o più condizioni logiche.
- ◆ Sia **I' attività** (associata allo stato) sia **le azioni** (associate alle transizioni – interne ed esterne) sono associate a qualche metodo di una classe.

Transizioni

Le **transizioni** (interne ed esterne) in un macchina a stati del comportamento sono caratterizzate:

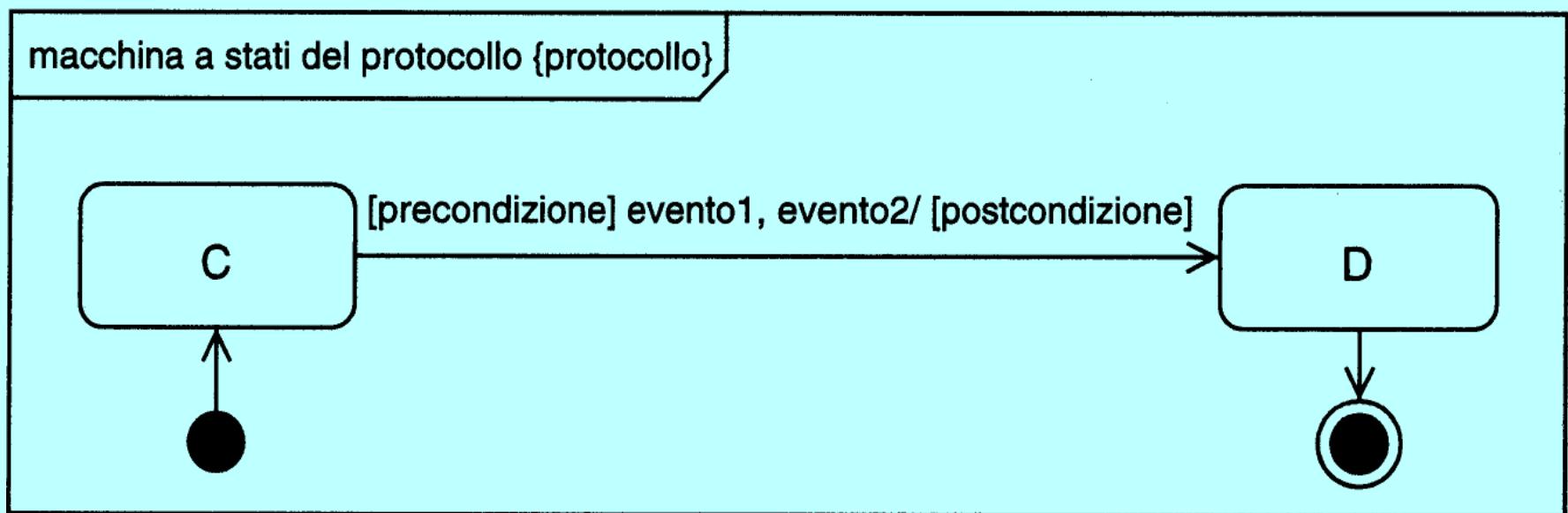
- ◆ dall'**evento** che genera la transizione
- ◆ dall' eventuale condizione logica (**guardia**) sotto la quale si genera l' evento
- ◆ dall' **azione** che dovrà essere effettuata in seguito alla transizione.



State Diagram

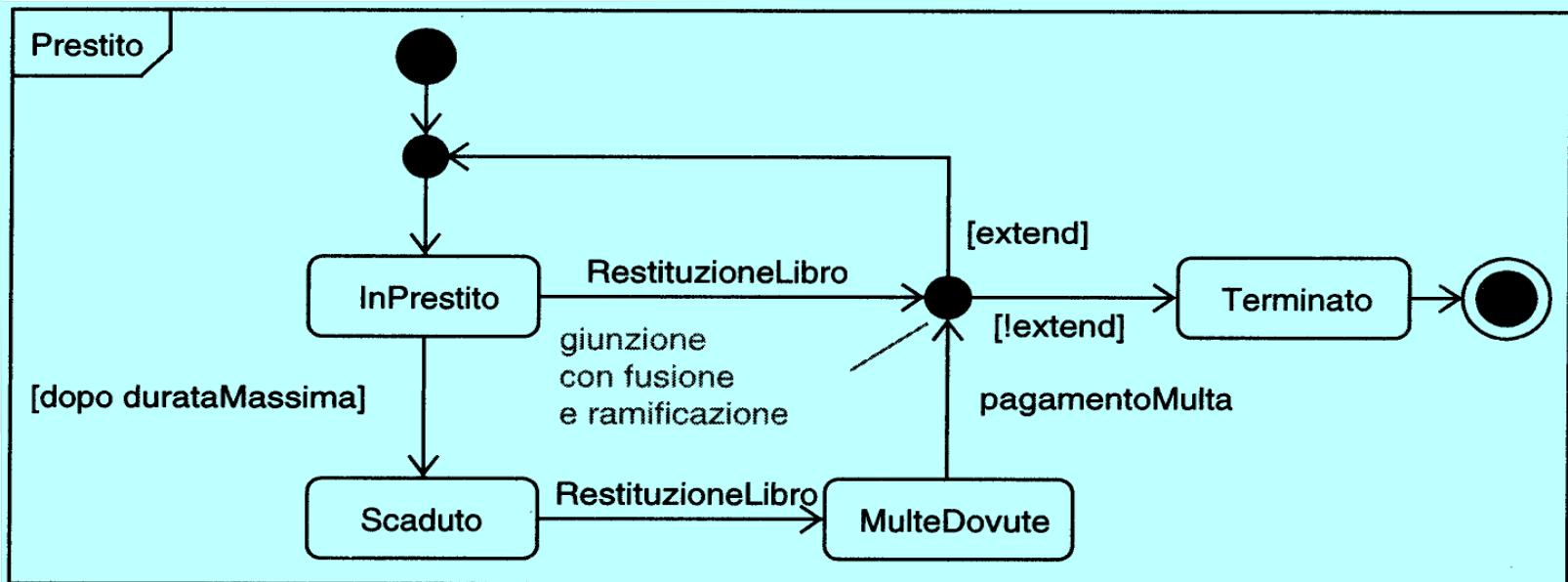
Le **transizioni** (interne ed esterne) in un macchina a stati del protocollo

- Non posseggono azioni
- Posseggono **precondizioni** e **postcondizioni** invece delle condizioni di guardia



State Diagram

- ◆ Una transizione che **non** ha alcun evento è detta **automatica**
- ◆ Le condizioni logiche delle **guardie** devono essere mutuamente esclusive, per cui una sola transizione potrà avvenire in un determinato momento
- ◆ Le transizioni possono essere connesse tramite **pseudo-stati** di **giunzione** e di **ramificazione**

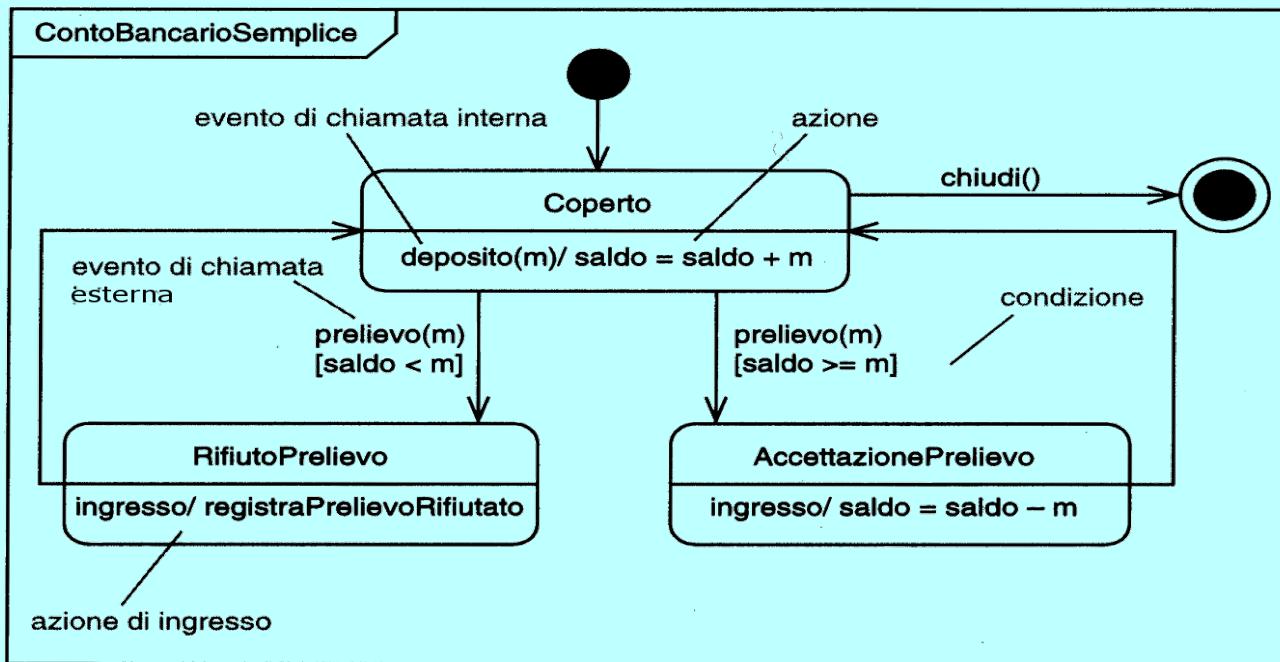


Eventi

- ♦ Un **evento** è la **specifica di un' occorrenza** di interesse che ha una **collocazione nel tempo e nello spazio**.
- ♦ Gli eventi attivano transizioni (interne ed esterne) nelle macchine a stati
- ♦ Esistono **quattro tipi di eventi** con diverse semantiche
 1. Evento di chiamata
 2. Evento di segnale
 3. Evento di variazione
 4. Evento temporale

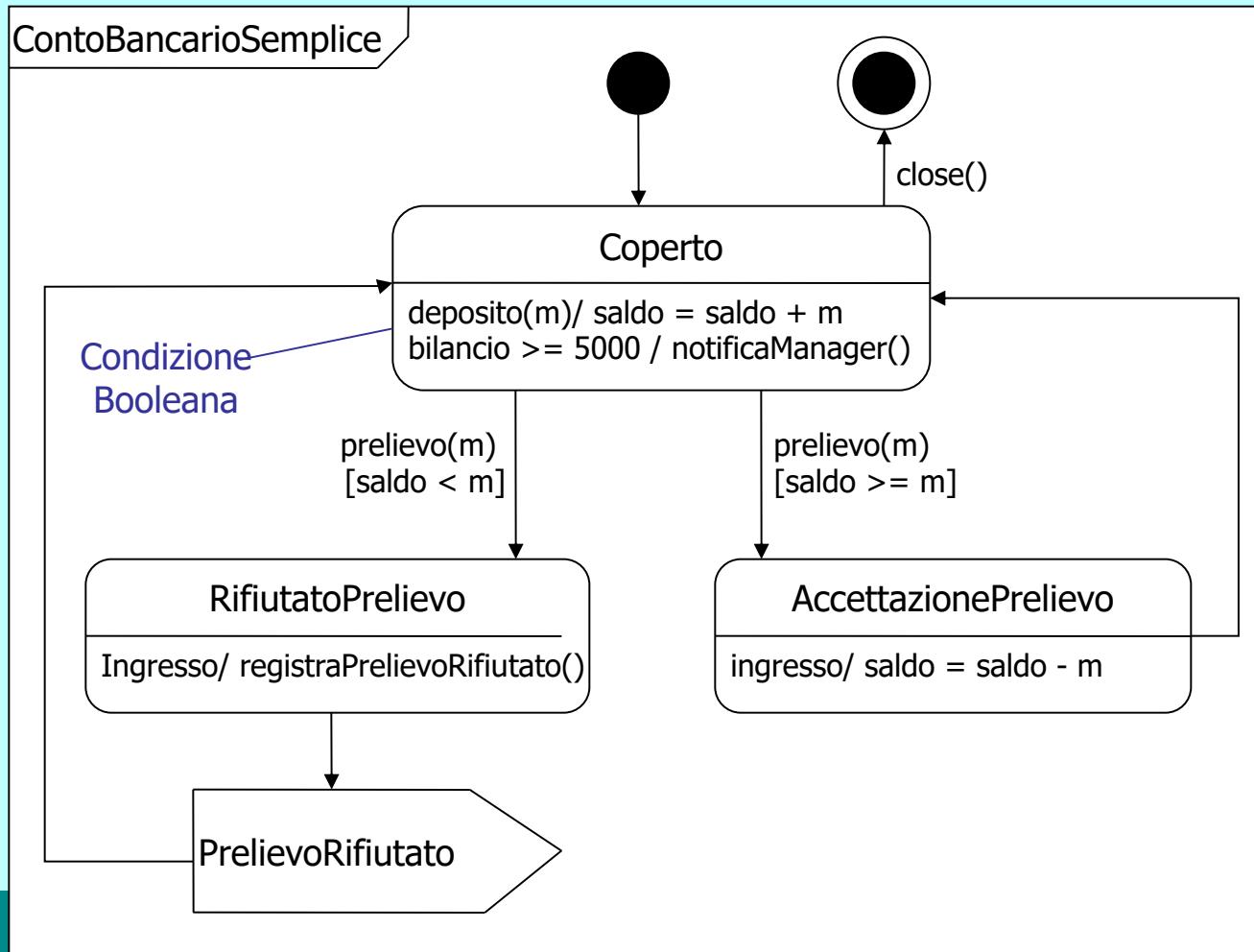
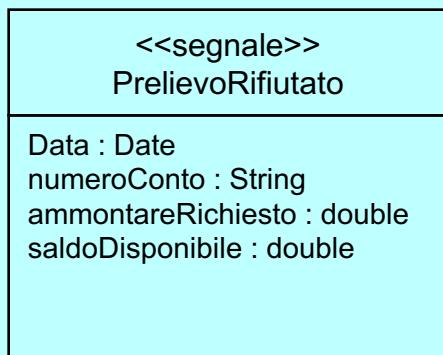
Eventi di chiamata

- Un **evento di chiamata** è l' invocazione di una specifica operazione
- La ricezione dell' evento attiva l' operazione.
- È possibile associare ad un evento di chiamata sequenze di azioni separate da punto e virgola



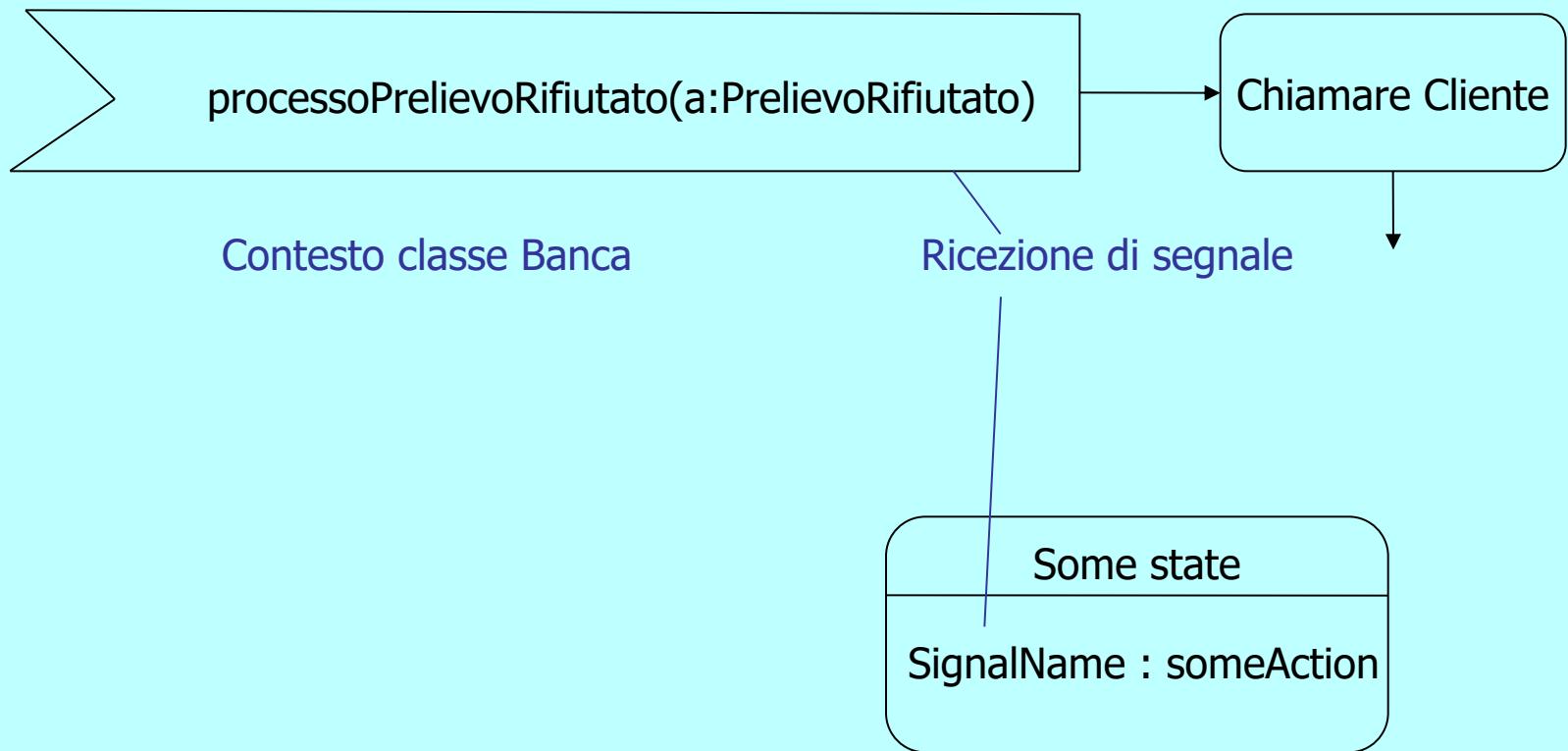
Eventi di segnale

- Un **segnale** è un pacchetto di informazioni inviato in modo **asincrono** da un oggetto a un altro. Esso viene modellato come una classe con stereotipo «**segnale**»



Eventi di segnale

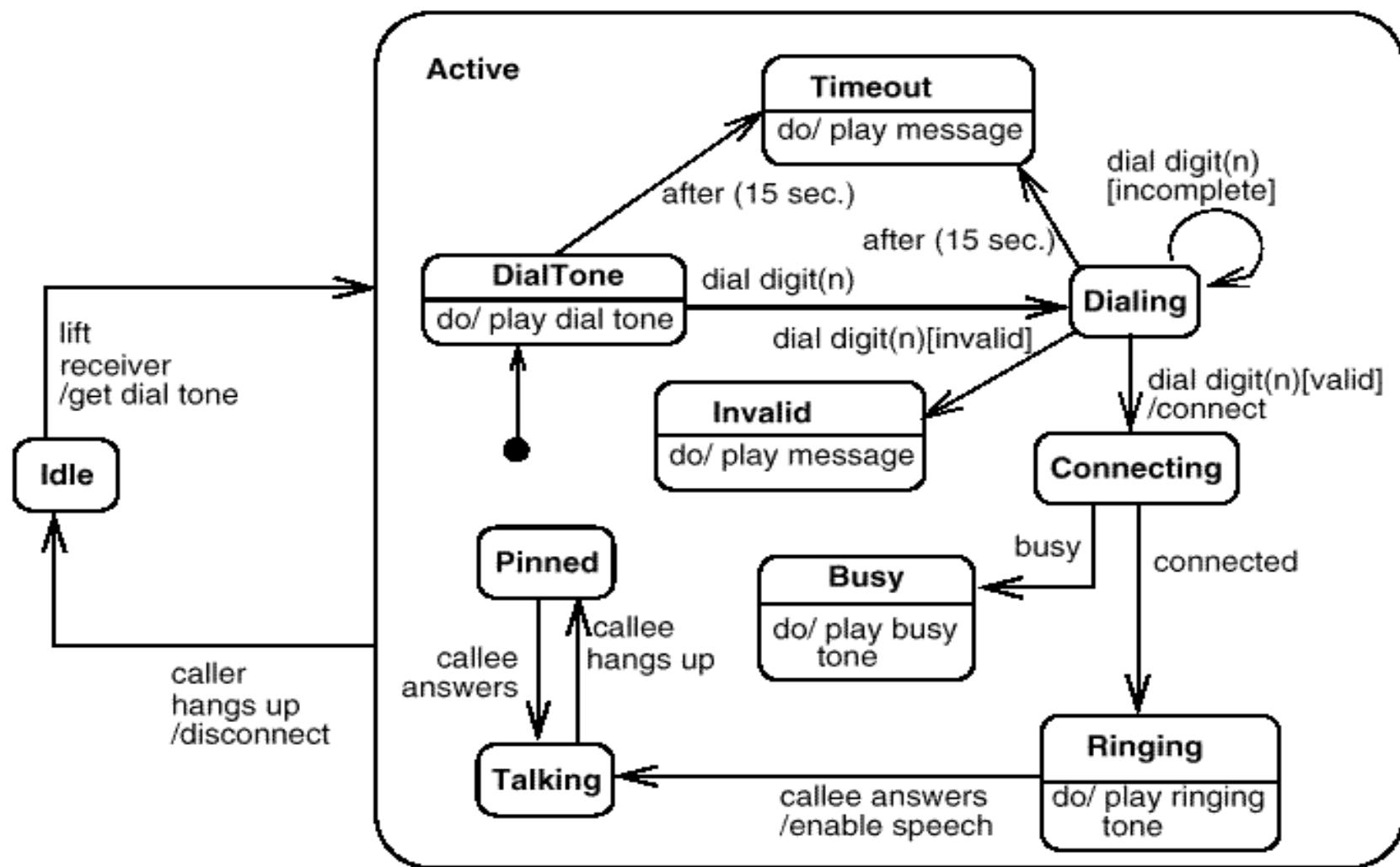
- ♦ Un **evento di segnale** occorre alla **ricezione** di un segnale



Eventi di variazione e temporali

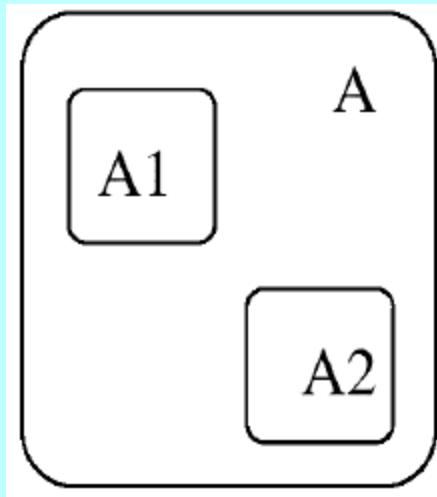
- ♦ Un **evento di variazione** occorre al trasformarsi di un predicato dal valore falso a vero
- ♦ Gli **eventi temporali** sono indicati tramite le parole chiave **quando** e **dopo**. Esse specificano un determinato **momento** in cui viene attivato l' evento.

Esempio



Stati composti

Stato composto: l'annidamento (*nesting*) di stati permette di dettagliare il comportamento interno di uno stato tramite sottostati:

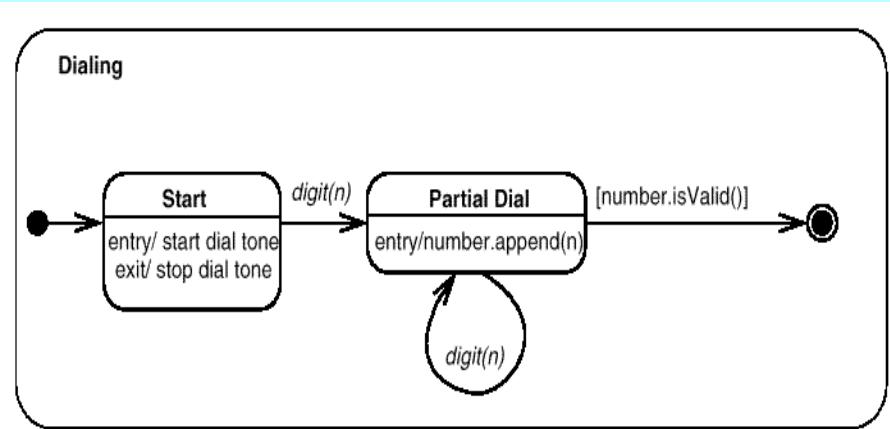


A: superstato

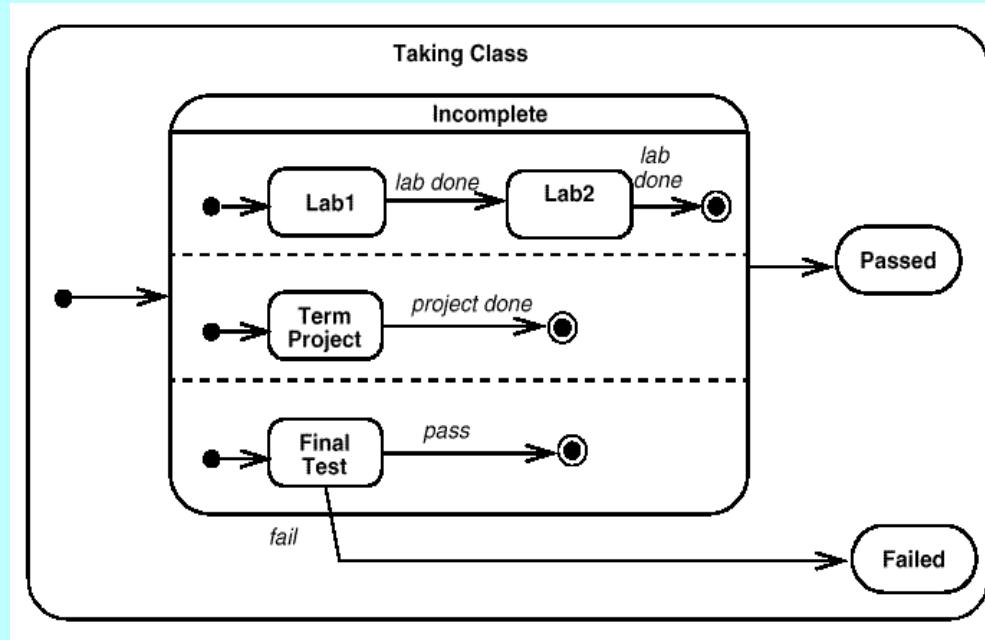
A1,A2 : sottostati

Stati composti

Uno stato può essere decomposto usando una AND-relationship in sotto-stati concorrenti oppure usando una OR-relationship in sotto-stati mutuamente esclusivi.



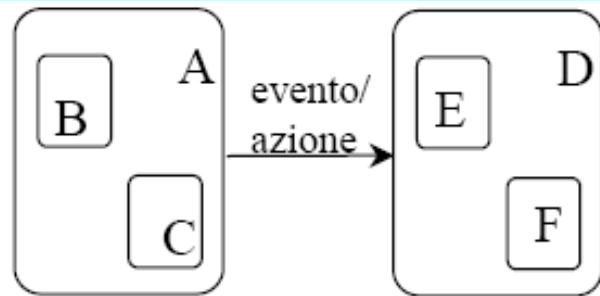
Sotto-stati sequenziali



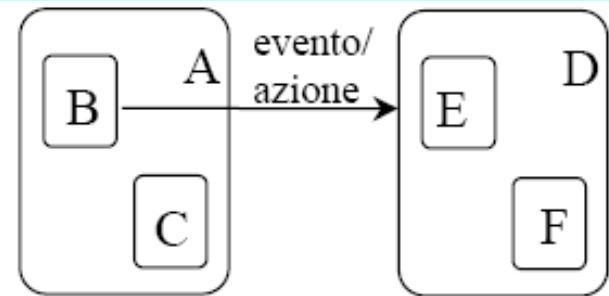
Sotto-stati concorrenti

Transizioni tra stati con stati composti

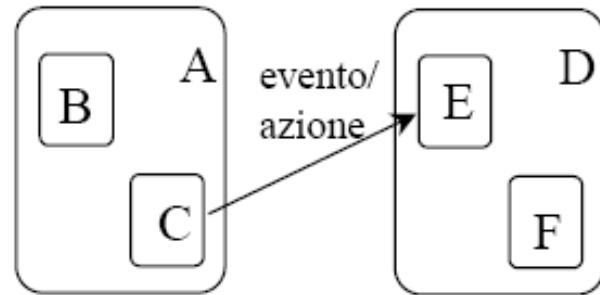
Esempi di transizioni



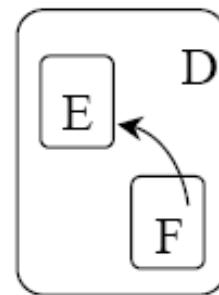
- 1) da un sottostato di A
al sottostato di default di D



- 2) dal sottostato B al sottostato
di default di D



- 3) dal sottostato C al sottostato E



- 4) dal sottostato E al sottostato F

Esempio

aggiorna disponibilità

**stato
iniziale**



prenota biglietto

Acquisisci prenotazione

Transizione
di stato

verifica prenotazione

stato

verificato e completato

scadenza termini di pagamento

annullato

spedito (OR)

via posta ordinaria

via posta elettronica

pagamento ricevuto

pagato

spedizione biglietto
al cliente

dopo una settimana

stato finale

dopo una settimana

spedito

Diagrammi di attività

Diagrammi di attività

I diagrammi di attività (**activity diagrams**) sono “diagrammi di flusso O-O”, che permettono di modellare un processo

- ♦ Sono molto utili nelle prime fasi dell’analisi, per es. per modellare il flusso di un caso d’uso, o il flusso tra più casi d’uso
- ♦ Sono anche utili per descrivere i *business processes* (*flussi di lavoro di business*) esistenti all’interno della realtà da modellare
- ♦ Questi processi possono essere, infatti, rappresentati tramite un certo numero di attività sequenziali o parallele.
- ♦ Servono anche a modellare i dettagli di un’operazione o di un algoritmo

Attività: può essere considerata come un’attività svolta da un essere umano o da una entità hardware o software, come un metodo di una classe

Diagrammi di attività

Il contenuto informativo di questi diagrammi è molto vicino a quello espresso dalle reti di Petri: si ha una visione del parallelismo intrinseco dei processi, delle condizioni abilitanti e delle sincronizzazioni.

Per questo motivo i diagrammi possono rivelarsi particolarmente utili nel caso di sistemi concorrenti.

Questo tipo di diagramma non indica (se non in sistemi poco complessi) gli “attori” responsabili delle attività.

Diagrammi di attività

È possibile associare un' attività a qualunque elemento di modellazione, al solo fine di modellare il comportamento di tale elemento.

Le attività sono tipicamente associate a:

- ◆ Casi d' uso
- ◆ Classi
- ◆ Interfacce
- ◆ Componenti
- ◆ Collaborazioni
- ◆ Operazioni

Diagrammi di attività e casi d' uso

- ♦ I diagrammi di attività possono essere utilizzati in fase d' analisi dei requisiti in associazione agli Use Case individuati.
 - I casi d' uso possono essere descritti come un insieme di scenari alternativi.
 - Gli scenari, d' altra parte, sono intrinsecamente sequenziali e quindi mal si prestano alla rappresentazione di attività complesse esprimibili anche in termini paralleli
 - *Utilizzando i diagrammi delle attività è possibile rappresentare graficamente uno Use Case in un tutt'uno (più scenari paralleli che si sincronizzano).*

Stati e transizioni

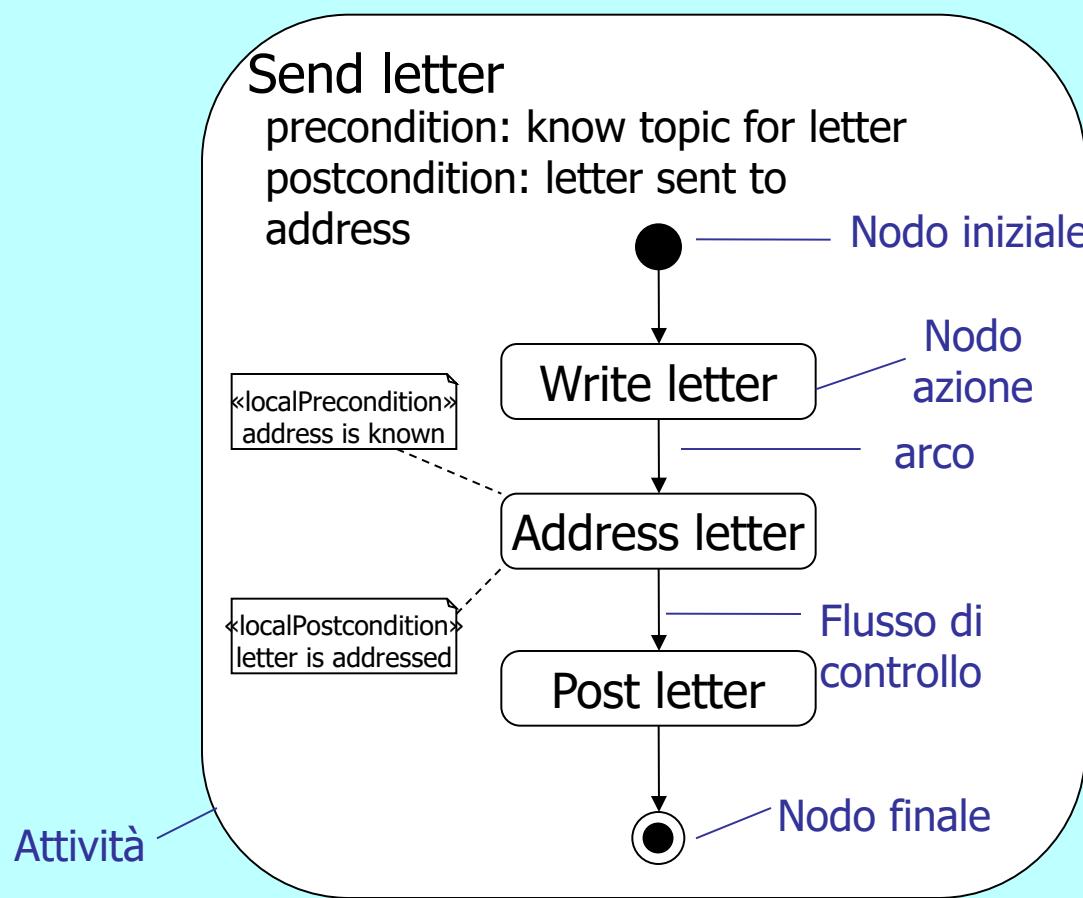
- ❖ Una attività è una **rete di nodi connessi da archi**, ove sono presenti:
 - nodi azione
 - ◆ Rappresentano unità discrete di lavoro atomiche all' interno dell' attività.
 - nodi controllo
 - ◆ Controllano il flusso attraverso l' attività.
 - nodi oggetto
 - ◆ Rappresentano oggetti usati nell' attività.
- ❖ Gli archi rappresentano il flusso attraverso l' attività; vi sono:
 - flussi di controllo
 - flussi di oggetti

Semantica delle attività

- ◆ I diagrammi di attività modellano il comportamento usando il *token game*. I token fluiscono dai nodi di controllo iniziali a quelli di controllo finali eseguendo i nodi azione con una dinamica analoga alle reti di Petri.
- ◆ I nodi azione sono eseguiti quando:
 - Esiste un token simultaneamente su ciascun arco entrante
 - I token in ingresso soddisfano tutte le pre-condizioni locali del nodo azione
- ◆ Il nodo di controllo iniziale indica il punto in cui l' esecuzione inizierà quando l' attività viene invocata.
 - Se esistono più nodi iniziali in una attività, i flussi cominciano simultaneamente in tutti i nodi iniziali e vengono eseguiti in concorrenza

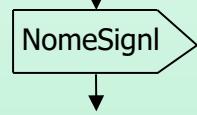
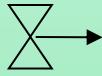
Semantica delle attività

- Quando un nodo azione ha terminato l'esecuzione, la post-condizione locale viene controllata; se è soddisfatta, il nodo emette token su tutti i suoi archi uscenti



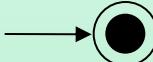
Nodi azione

◆ Esistono quattro tipi di nodi azione:

<p>Nodo azione di chiamata</p> 	<p>Può attivare un'attività (simbolo a rastrello), o un comportamento del contesto o un'operazione.</p> 
<p>Azione invio segnale</p> 	<p>Invia un segnale in modo asincrono (il mittente non aspetta la conferma di segnale ricevuto).</p> <ul style="list-style-type: none">Può accettare parametri di input necessari per creare il segnale.
<p>Azione accettazione evento</p> 	<p>Aspetta gli eventi individuati dal suo oggetto proprietario ed emette l'evento sul suo arco di uscita.</p> <ul style="list-style-type: none">È attivato quando riceve un token sul suo arco in entrata.Se non ci sono archi in entrata inizia quando l'attività corrente inizia.
<p>Nodo accetta evento temporale</p>  <p>Espressione temporale</p>	<p>Risponde al tempo, genera eventi temporali secondo la sua espressione temporale.</p>

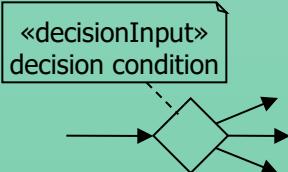
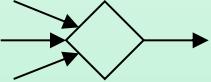
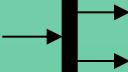
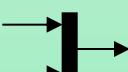
Nodi di controllo (1/2)

◆ Gestiscono il flusso di controllo di un'attività

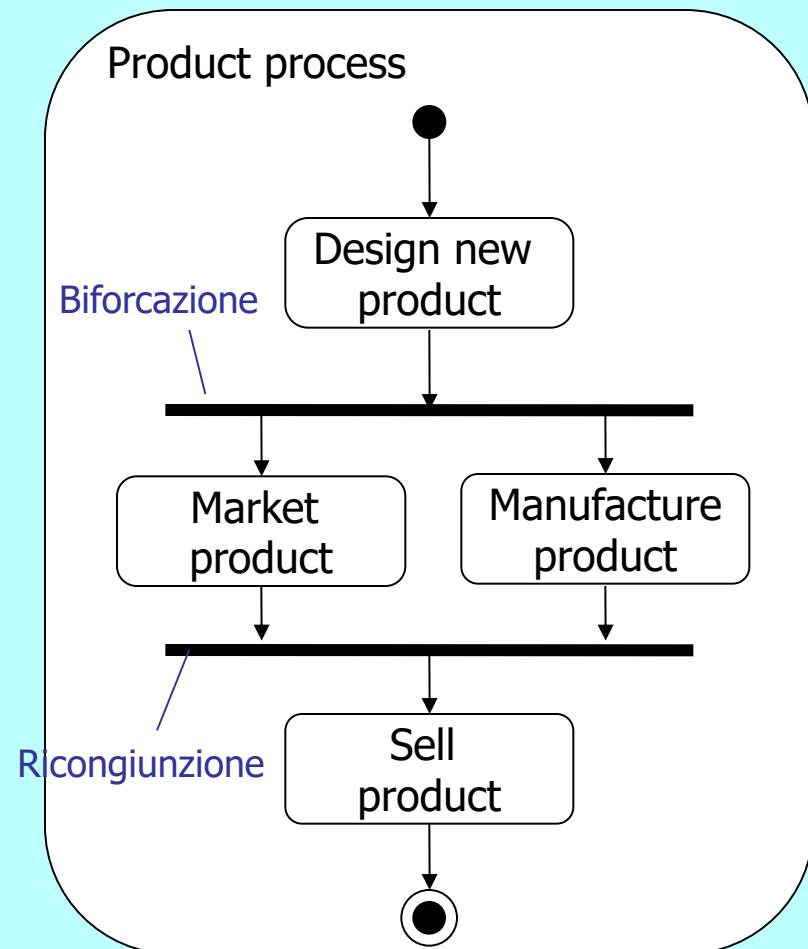
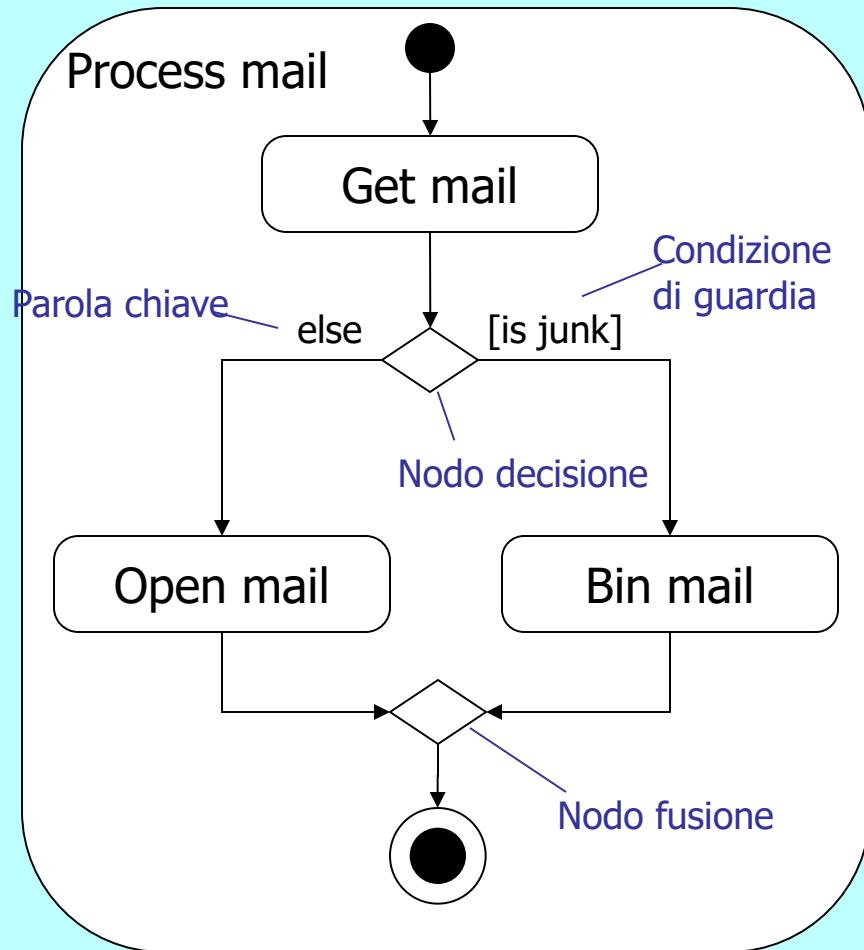
Nodo iniziale 	Indica dove inizia il flusso quando viene invocata un'attività. <ul style="list-style-type: none">• Se ci sono più nodi iniziali i flussi cominciano simultaneamente• I flussi possono essere anche iniziati da un'azione accettazione evento o da un nodo parametro di attività.
Nodo finale 	Termina un'attività. <ul style="list-style-type: none">• Il primo nodo finale ad essere attivato ferma tutti i flussi all'interno di un'attività
Nodo finale del flusso 	Termina un flusso specifico all'interno di un'attività senza influenzare gli altri flussi.

Nodi di controllo (2/2)

♦ Gestiscono il flusso di controllo di un'attività

<p>Nodo decisione</p> <p>«decisionInput» decision condition</p> 	<p>Il token attraversa l'unico arco in uscita la cui condizione di guardia è vera.</p> <ul style="list-style-type: none">Se più condizioni sono vere il comportamento è indefinitoSi possono associare note con stereotipi <<inputDecisione>> come condizione delle guardie
<p>Nodo fusione</p> 	Copia token in ingresso nel suo unico arco in uscita.
<p>Nodo biforcazione</p> 	Divide il flusso in più flussi concorrenti (fork).
<p>Nodo ricongiunzione {join spec}</p> 	Sincronizza più flussi concorrenti (join).

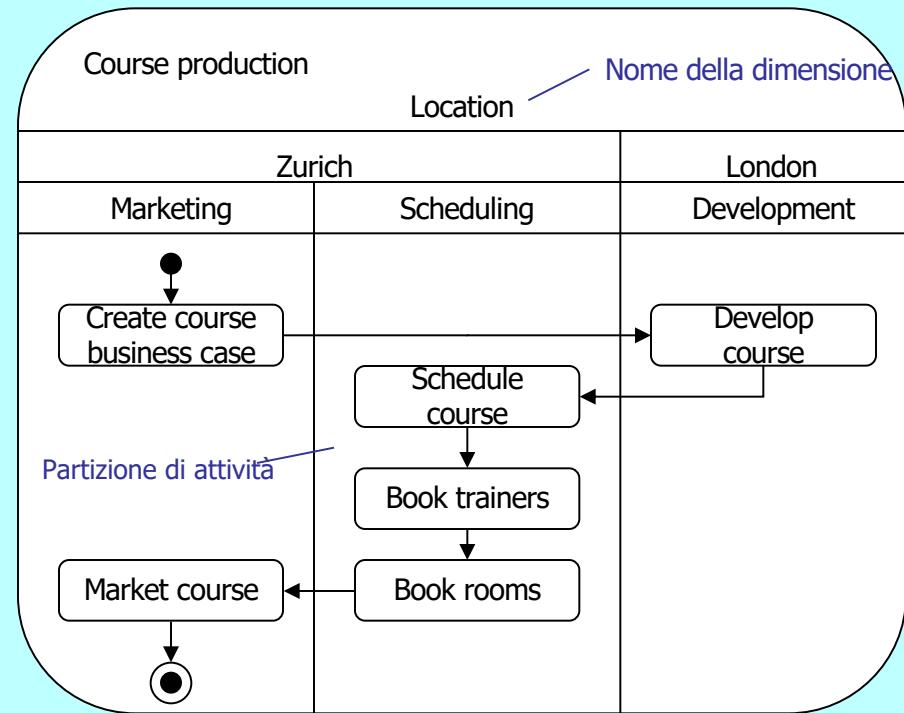
Esempi



Partizioni delle attività (swimlanes)

Per facilitare la lettura, è possibile suddividere le attività in partizioni usando linee verticali, orizzontali o curve.

- ◆ Le partizioni (chiamate corsie o swimlane) rappresentano raggruppamenti di alto livello di azioni correlate
- ◆ Le corsie non hanno una semantica intrinseca, la decide il modellatore definendo il nome delle dimensioni di partizionamento



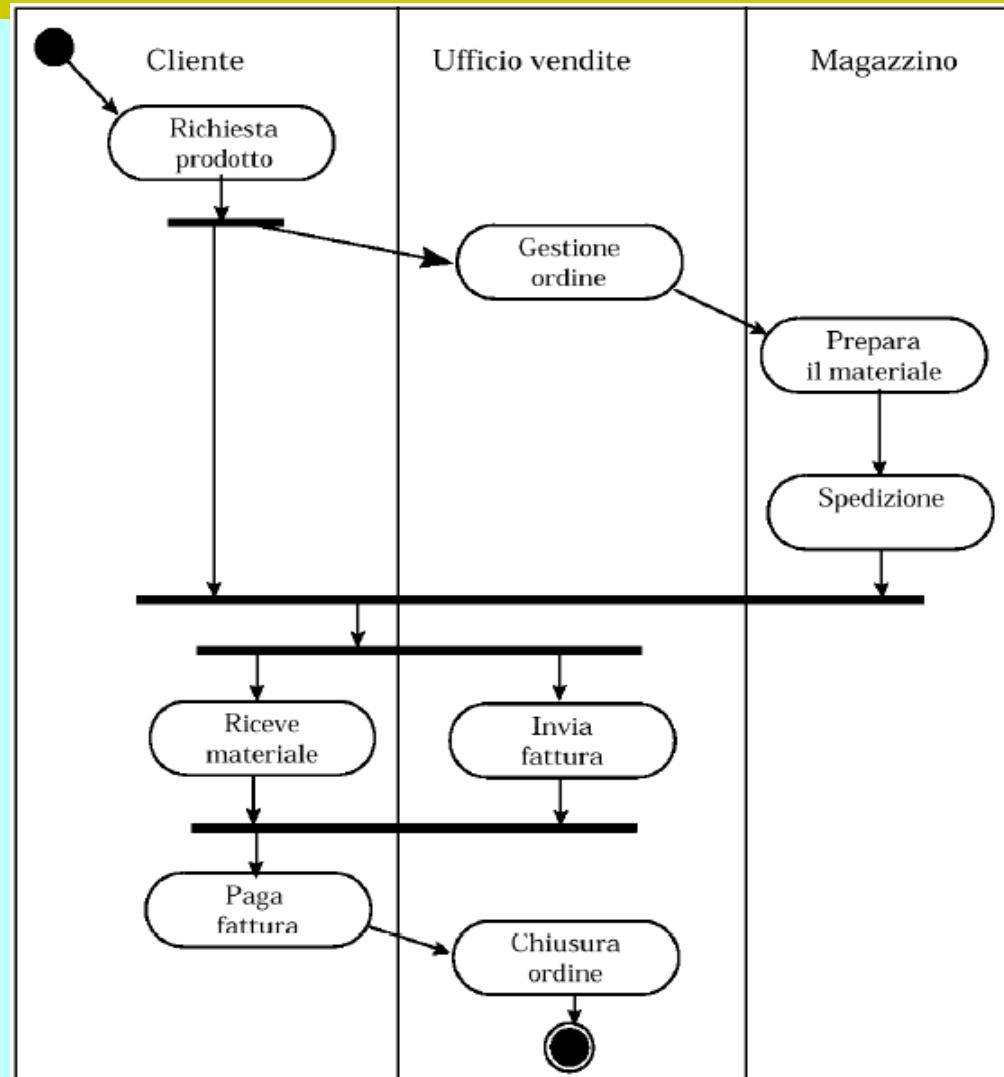
Partizioni delle attività (swimlanes)

È possibile rappresentare quali sono le classi responsabili per le attività utilizzando le partizioni

Si divide il diagramma in zone verticali mediante linee tratteggiate, ognuna delle quali è di responsabilità di un attore o classe

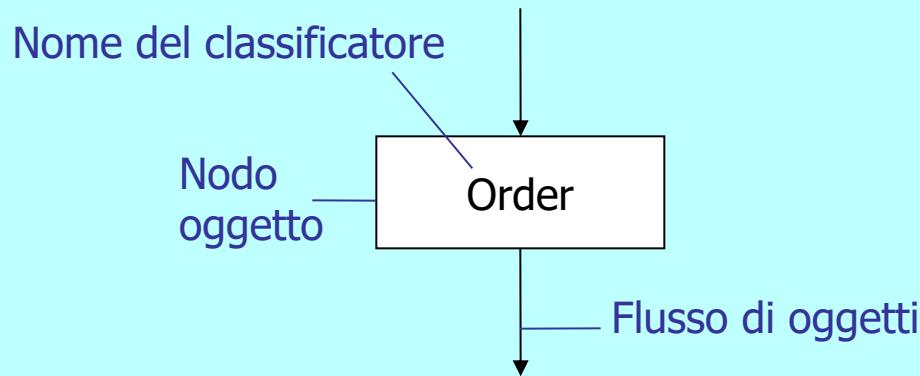
L'oggetto responsabile per l'esecuzione di una azione può essere rappresentato visualizzando la sua corsia del tempo di vita e posizionando le azioni di cui è responsabile lungo questa corsia.

Possono esserci differenti corsie per differenti oggetti della stessa classe o di classi diverse.



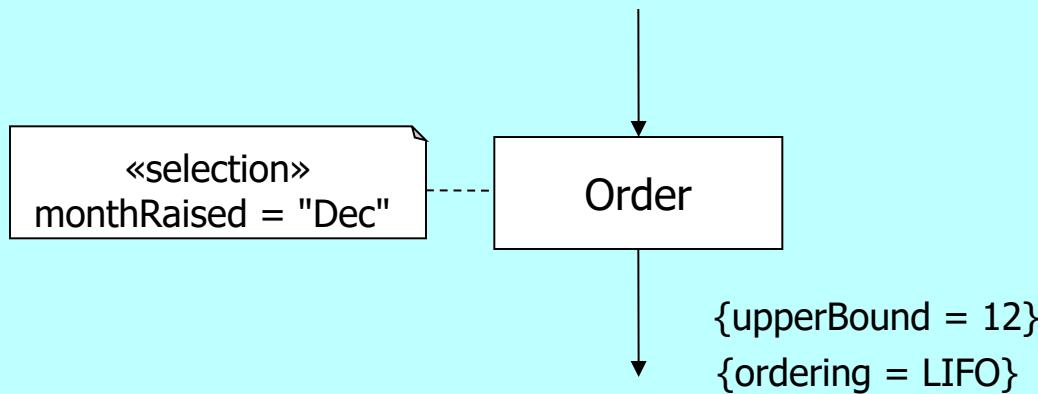
Nodi oggetto (1/3)

- ♦ I nodi oggetto indicano che sono disponibili istanze di un particolare classificatore (o delle sue sottoclassi) in un punto specifico dell'attività
 - Gli archi in entrata e in uscita di nodi oggetto sono *flussi di oggetti*.
 - Rappresentano la creazione, il movimento e il consumo di oggetti all'interno dell'attività



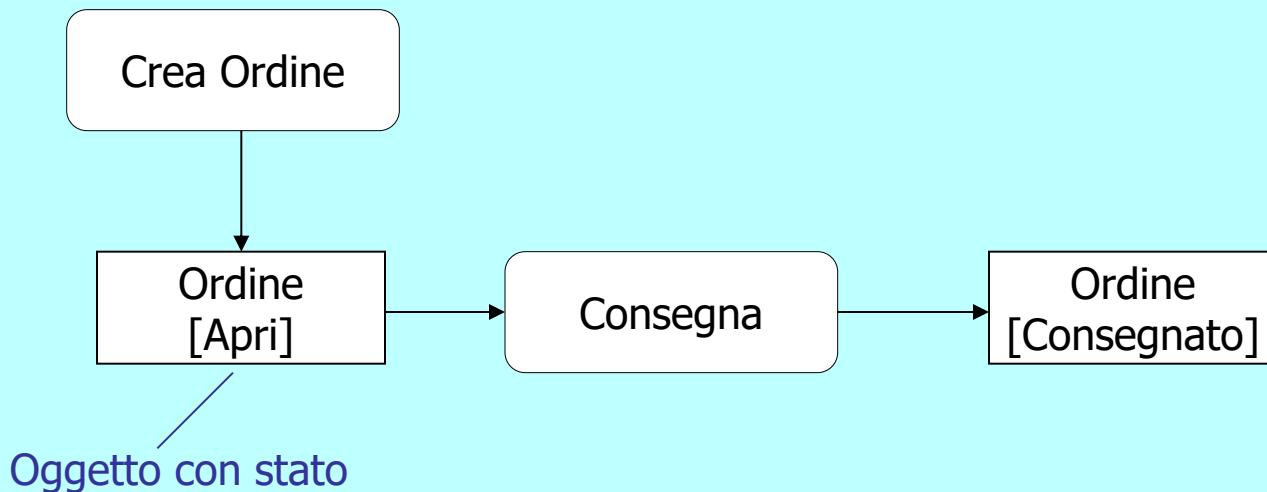
Nodi oggetto (2/3)

- ◆ Differentemente dai nodi azione, i nodi oggetto:
 - Offrono solamente un token su tutti i loro archi in uscita e questi competono per riceverlo.
 - Fungono da buffer, cioè punti dell'attività in cui i token possono accumularsi. È possibile specificare il numero massimo di token nel buffer e la politica di ordinamento dei token (es.: FIFO, LIFO).
 - Possono avere un comportamento di selezione degli oggetti presentati agli archi in entrata secondo un criterio definito dal modellatore tramite una nota con stereotipo <<selezione>>.



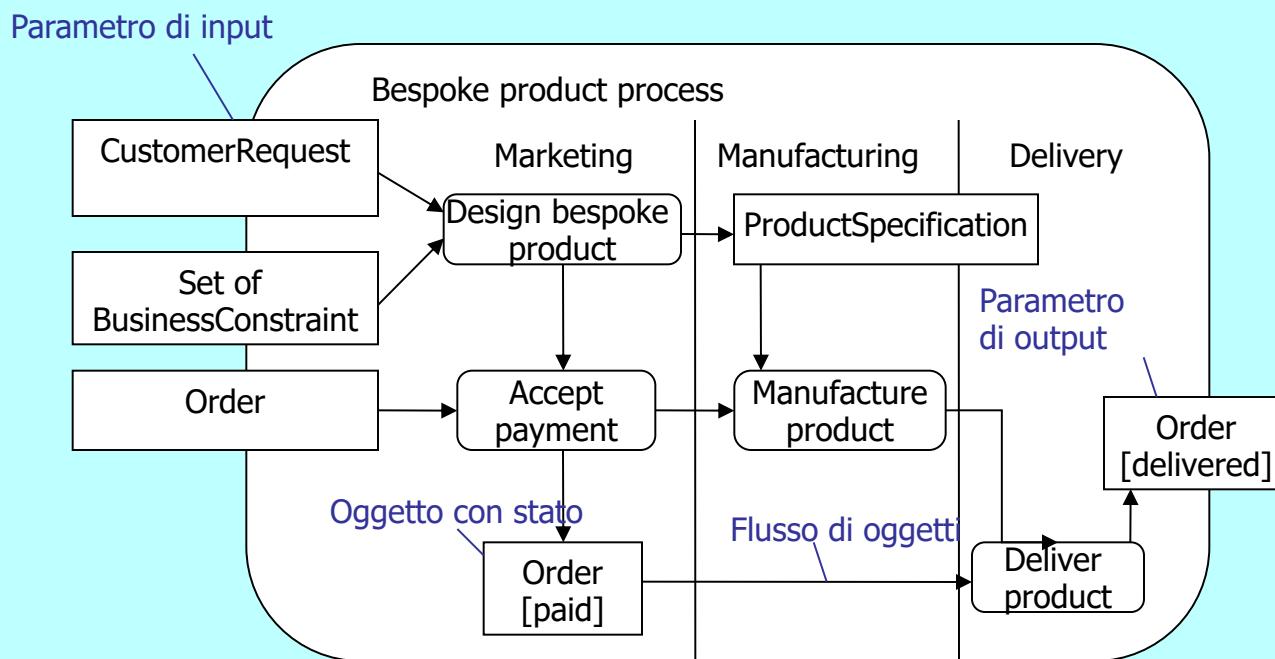
Nodi oggetto (3/3)

- ♦ I nodi oggetto possono rappresentare gli **oggetti in un particolare stato** e consentono di comunicare come i nodi azione modificano lo stato del flusso di oggetti.



Parametri di attività

- È possibile usare i nodi oggetto per fornire input alle attività e output dalle attività.
- I nodi oggetto in ingresso e uscita sono disegnati sovrapposti con il riquadro dell'attività
- I parametri di input hanno uno o più flussi di uscita nell'attività
- I parametri di output hanno uno o più flussi di ingresso dall'attività



Pin

- Un pin è un nodo oggetto che rappresenta un input in un'azione o un output da un'azione.
 - I Pin hanno esattamente un arco in entrata ed un arco in uscita
 - Sono usati per semplificare la rappresentazione grafica dei diagrammi di attività con molti flussi di oggetti

