UML: Diagramma delle attivita'

Roberta Gori, Laura Semini
Ingegneria del Software
Dipartimento di Informatica
Università di Pisa

Riassunto lezione precedente
Outline della lezione

- Lezioni precedente:
  - Descrizione del dominio: modello statico
- Questa lezione
  - Descrizione del dominio: modello dinamico
    - diagrammi di attività (business model)
    - diagrammi di macchina a stati

Diagrammi di attività

- Modellano il flusso di lavoro (workflow, business model)
  - di un compito o algoritmo o
  - di un processo/attività
- Un'attività descrive la coordinazione di un insieme di azioni. Centrata su:
  - sequenza e concorrenza delle azioni
  - e sulle condizioni che le abilitano
  - piuttosto che sui classificatori che eseguono queste azioni
- Antenati: flow charts e Reti di Petri

Diagrammi di attività

Modellano un'attività relativa a una qualsiasi entità o collezione di entità, ad esempio:

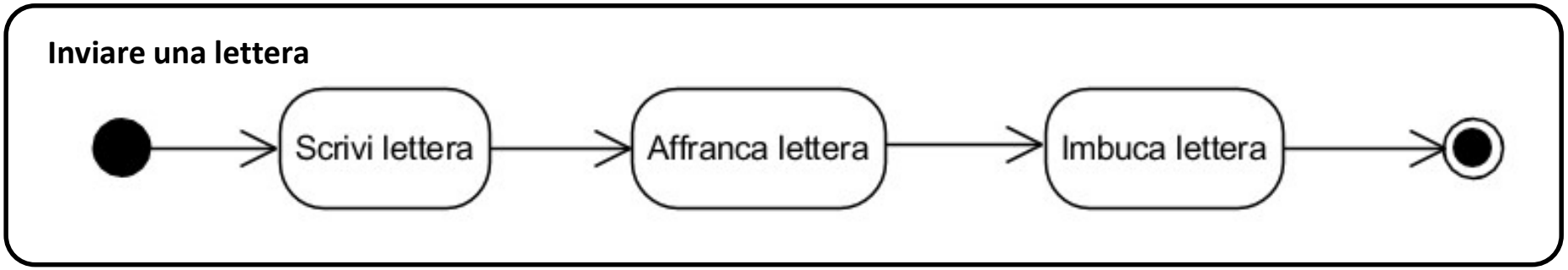
- una o più classi che collaborano in una attività comune
- uno o più attori con il sistema
- un'operazione di classe

Alcuni usi dei diagrammi di attività:

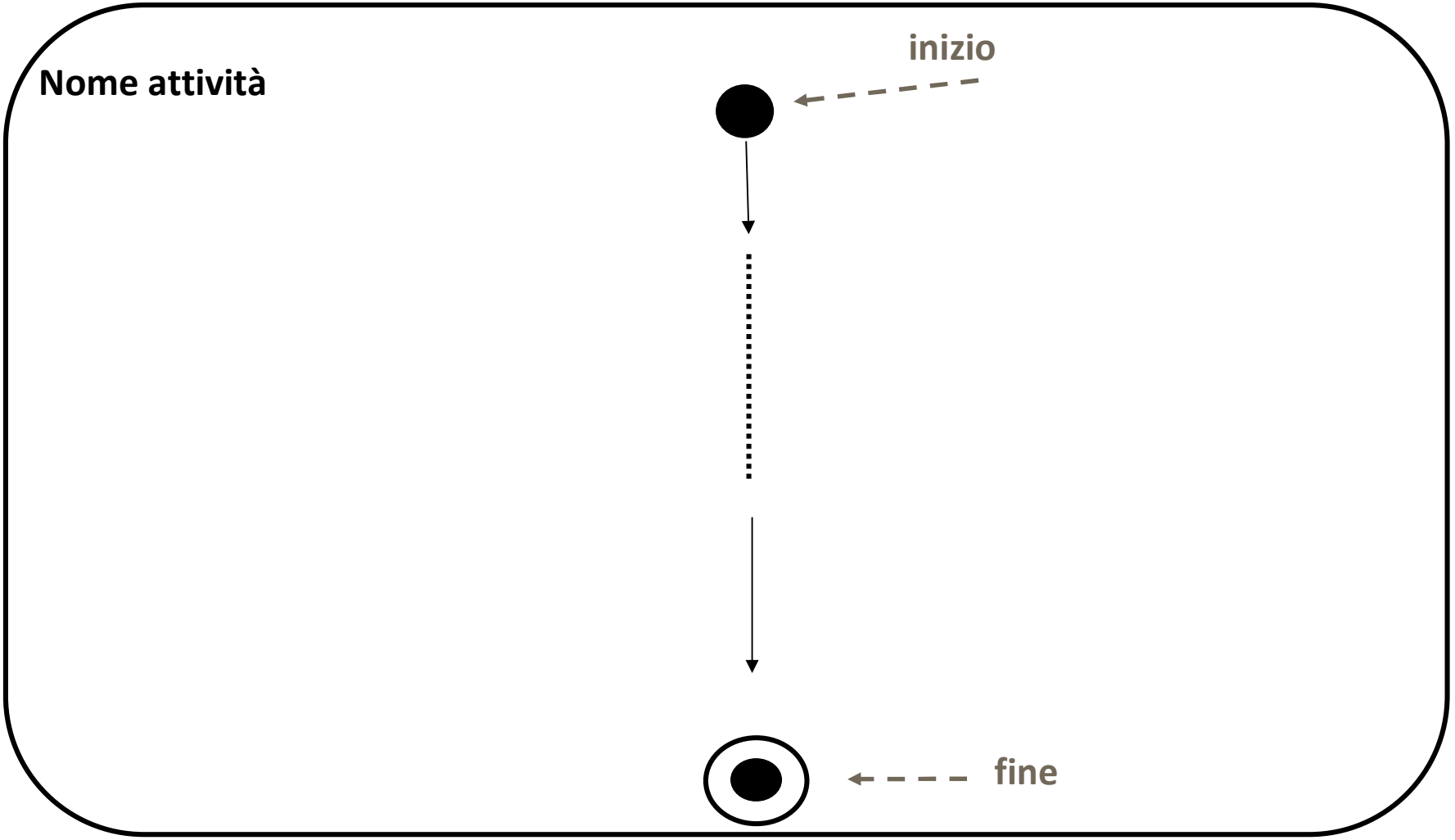
- modellare un processo aziendale (analisi)
- modellare il flusso di un caso d'uso (analisi)
- modellare il funzionamento di un'operazione di classe (progettazione)
- modellare un algoritmo (progettazione o testing)

Il concetto principe: l'attivita'

- Un'attività ha un nome ed è contenuta in un rettangolo con gli angoli smussati
- Il contenuto di un'attività è un grafo diretto i cui:
  - i nodi rappresentano le componenti dell'attività, come le azioni o i nodi di controllo (inizio, fine, etc)
  - gli archi rappresentano il control flow: i possibili path eseguibili per l'attivita'.



Diagrammi di attività: inizio e fine

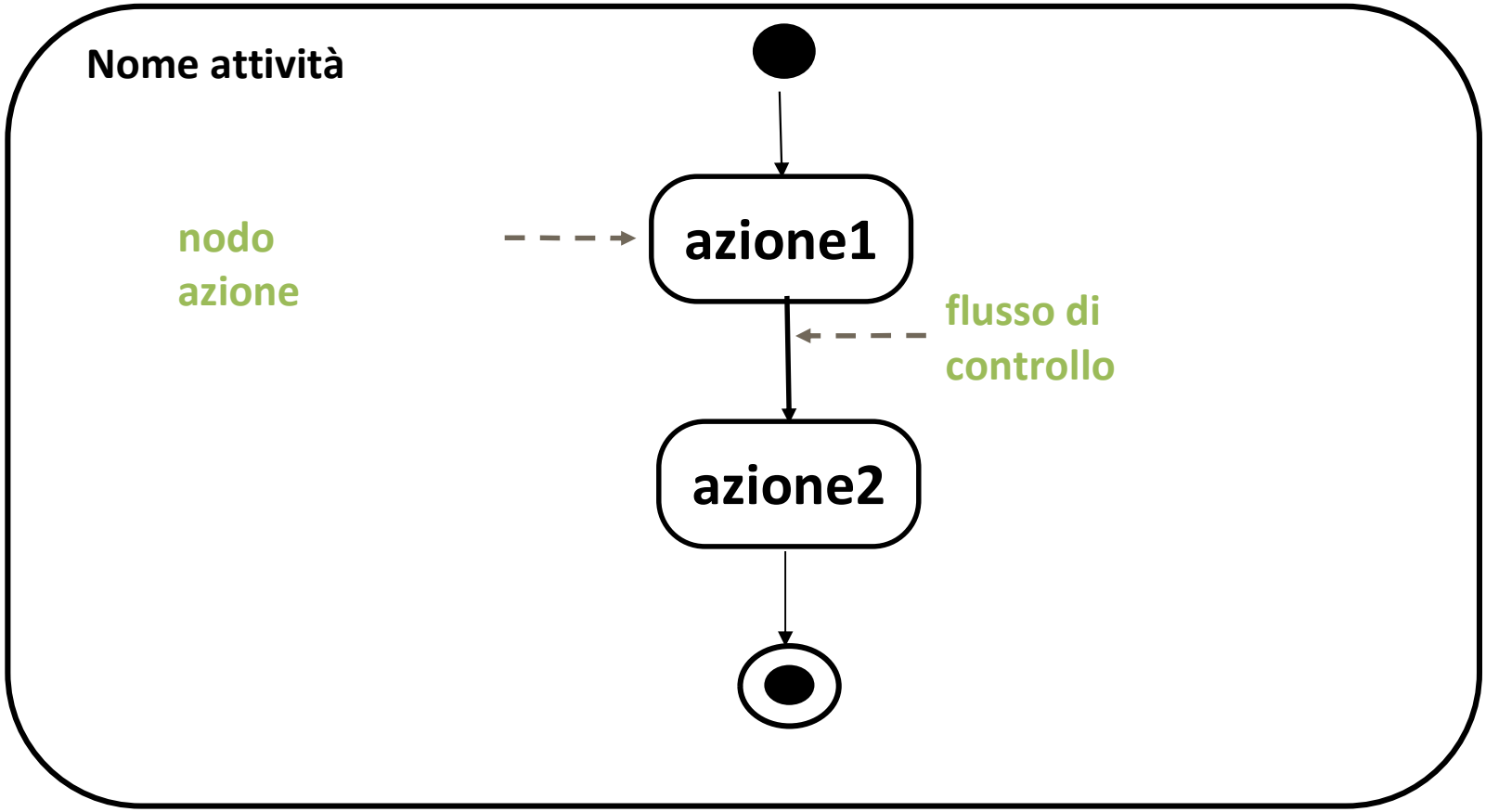


Le azioni

- Le azioni sono rappresentate anche esse da rettangoli con angoli smussati

- Possono essere specificate in linguaggio naturale
  - il nome di un'azione deve descrivere un'azione, quindi tipicamente essere un verbo
- Sono atomiche
- Oltre ai nodi azioni atomici, esistono dei nodi azione con comportamento non atomico, il cui dettaglio è specificato da diagramma di (sotto)attivita' (li vedremo tra qualche lucido)

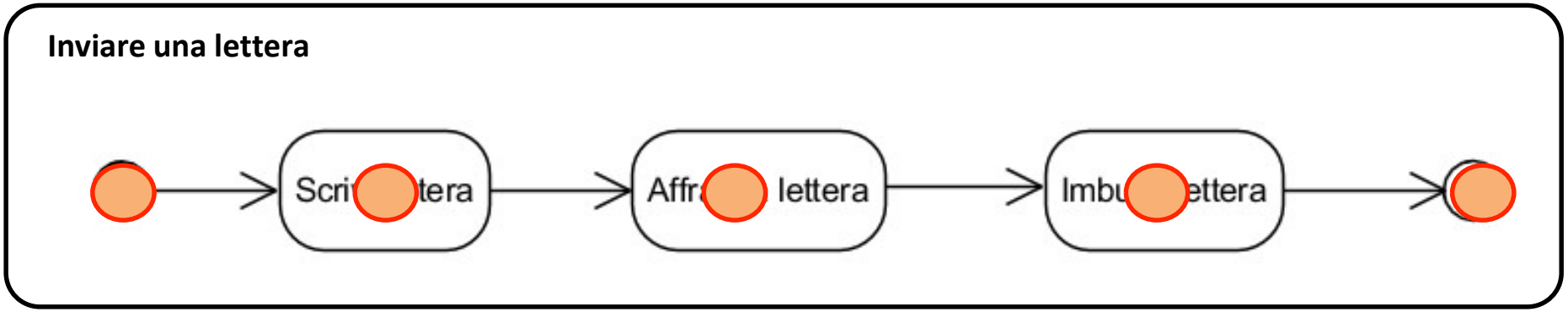
Diagrammi di attività: nodo azione



- Solo una freccia entrante e una uscente per ogni azione (vedremo perché)
- la freccia di uscita è presa appena è terminata l'azione

Transizioni

- Quando un'azione ha terminato il proprio lavoro scatta una **transizione automatica** in uscita dall'azione che porta all'azione successiva



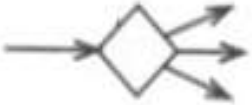
- La semantica è descritta con il token game: l'azione può essere eseguita quando riceve il token

Nodi di controllo

nodo iniziale

nodo finale

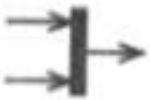
nodo di fine flusso



nodo decisione (con guardie sulle frecce uscenti)

nodo fusione

nodo di biforcazione (fork)



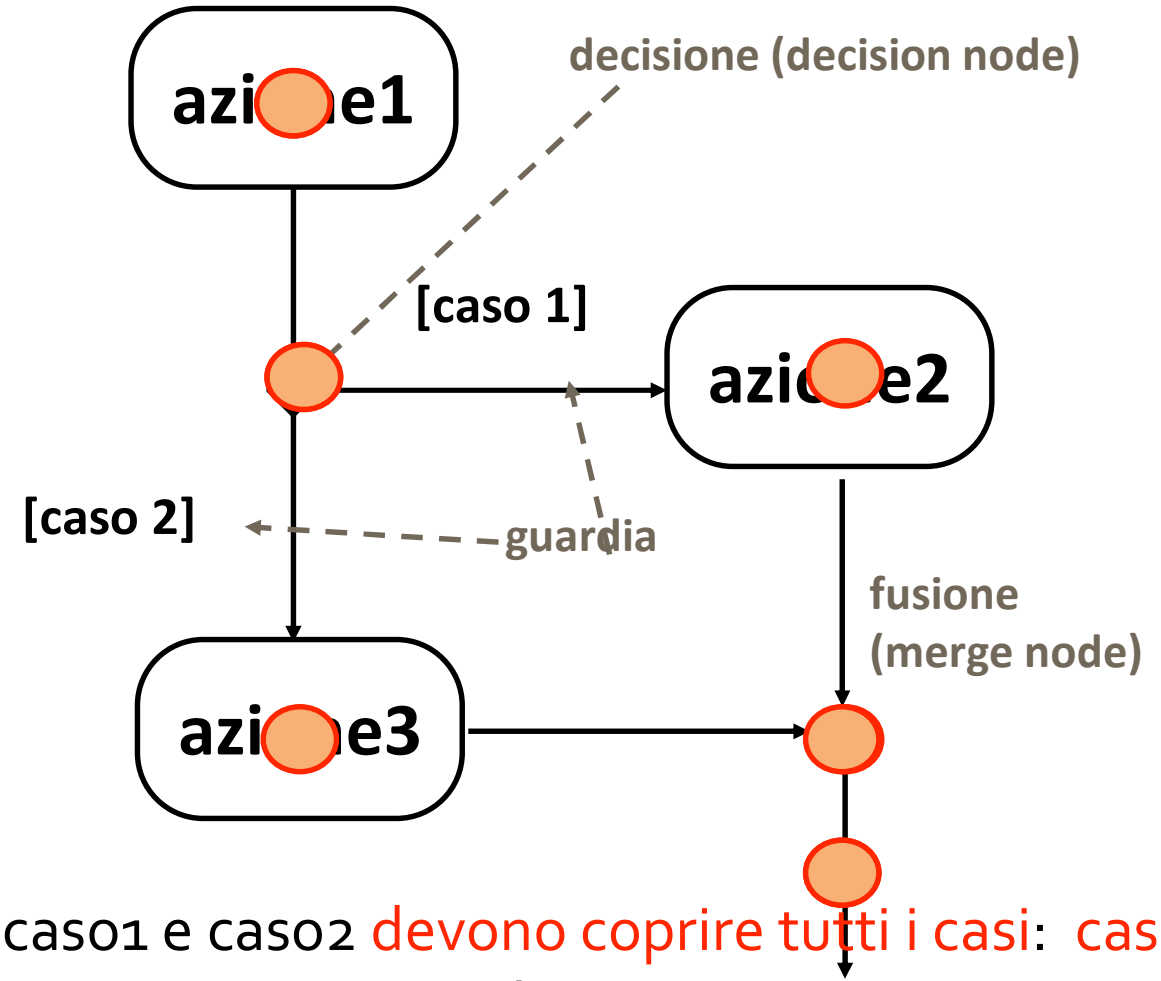
nodo di sincronizzazione (join)

Diagramma delle attivita' : scelta

- Abbiamo detto che ogni azione si attiva appena riceve un token, si esegue e poi passa il token sull'arco uscente.
- Questo meccanismo di passaggio del token viene alterato da una choice

-

Diagrammi di attività: scelta



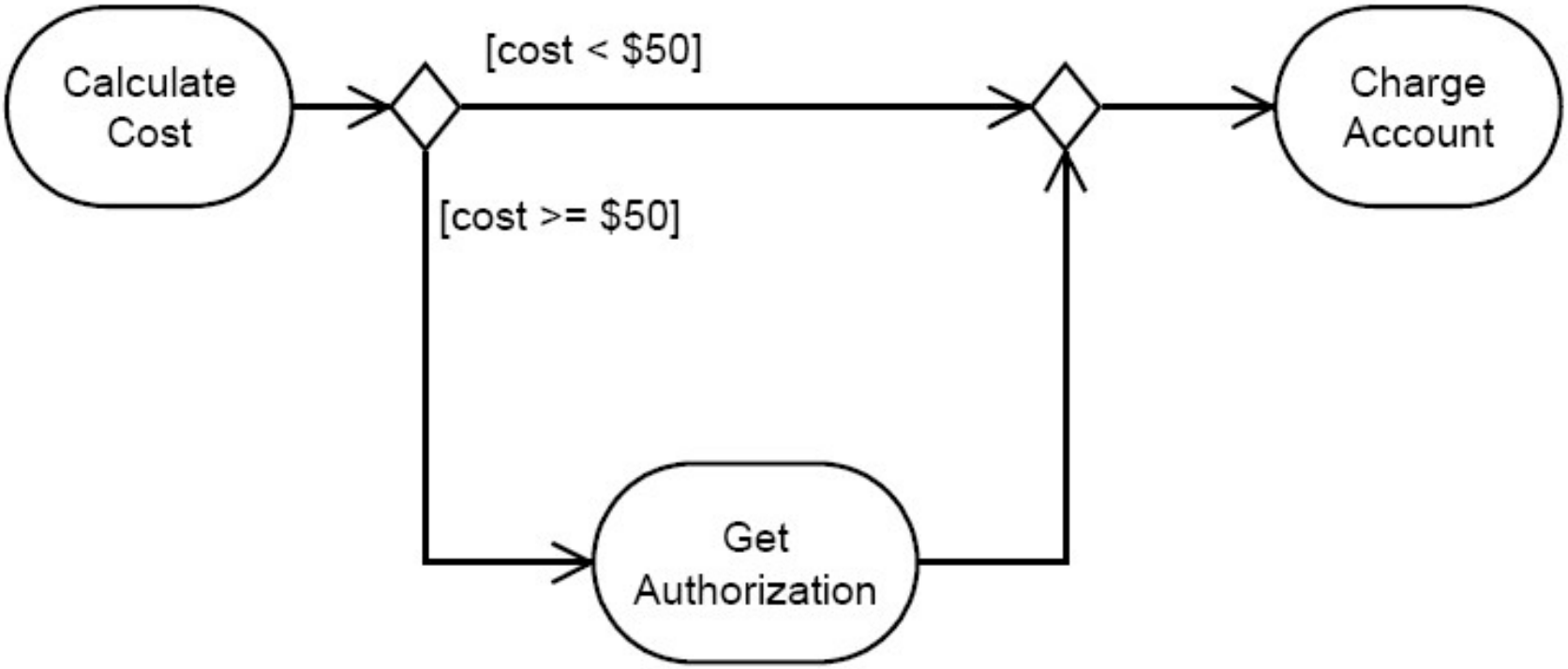
- caso1 e caso2 devono coprire tutti i casi: caso 1 OR caso2=True
- In una guardia si può scrivere "else"

Diagramma delle attivita' : scelta

A decision node has one input and two or more outputs. The input value is used to evaluate guard conditions on each of the outputs. If a guard condition evaluates true, the corresponding output is eligible for selection. Exactly one eligible output is chosen to receive a copy of the input value. If more than one guard condition evaluates true, the choice of output is nondeterministic. If no guard condition is true, the model is ill formed.

- Attenzione, sul libro il paragrafo sulla scelta e le guardie ha due imprecisioni:
  - Il libro afferma che è possibile avere g1 OR g2 ≡ False (in generale $OR_i\ g_i \equiv$ False) violando la parte sottolineata in rosso dello standard
  - Inoltre nello standard vale la parte sottolineata in verde, mentre il libro afferma che le guardie devono essere mutuamente esclusive, i.e. che deve valere g1 AND g2 ≡ False (in generale $AND_i\ g_i \equiv$ False)

Esempio:
Decisione e fusione / decision and merge

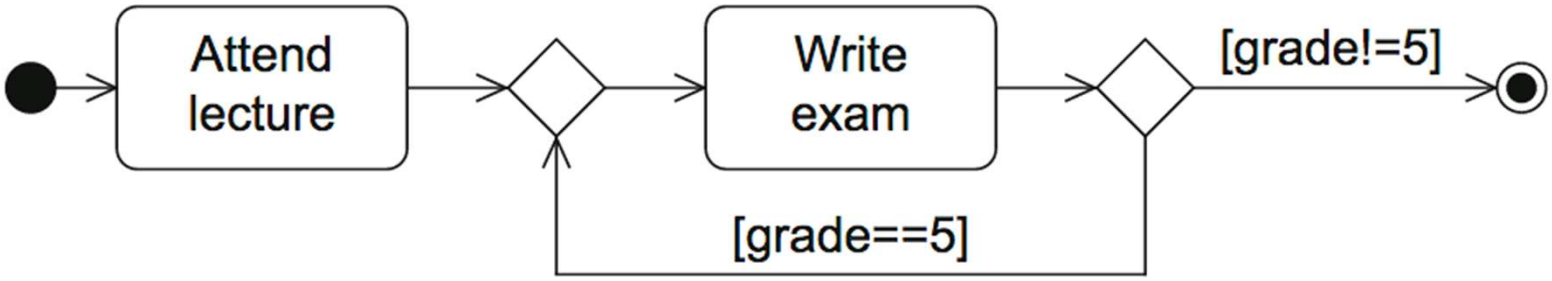


- Il token prende uno dei cammini
- Deve prendere sempre uno dei cammini

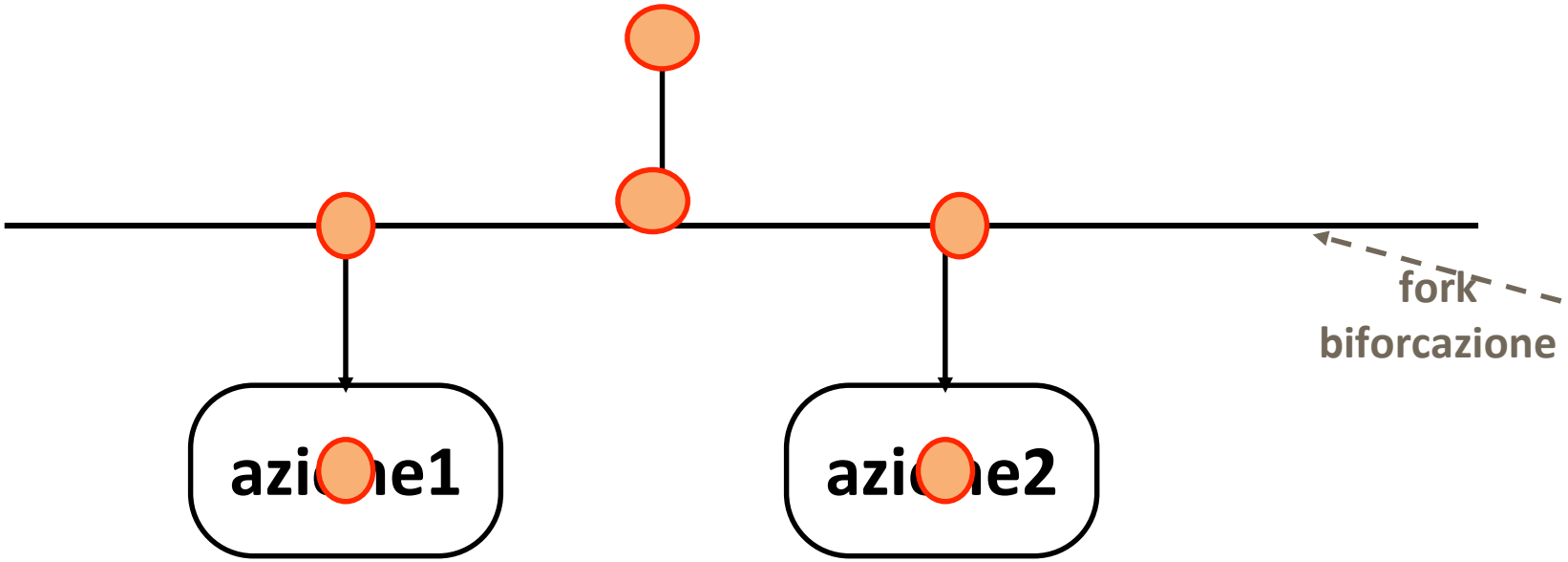
Semanica:
Decisione e fusione / decision and merge

- Le guardie devono coprire tutte le possibilità
  - In caso si usa [else]
- E' bene (ma non necessario) che siano mutualmente esclusive altrimenti comportamento non definito (non deterministico).
- Le condizioni di guardia sempre tra [ ]
  - (in generale in UML)
- Dato un nodo decisione non è obbligatorio un nodo fusione corrispondente.
  - Potrebbe per esempio esserci un nodo di fine flusso

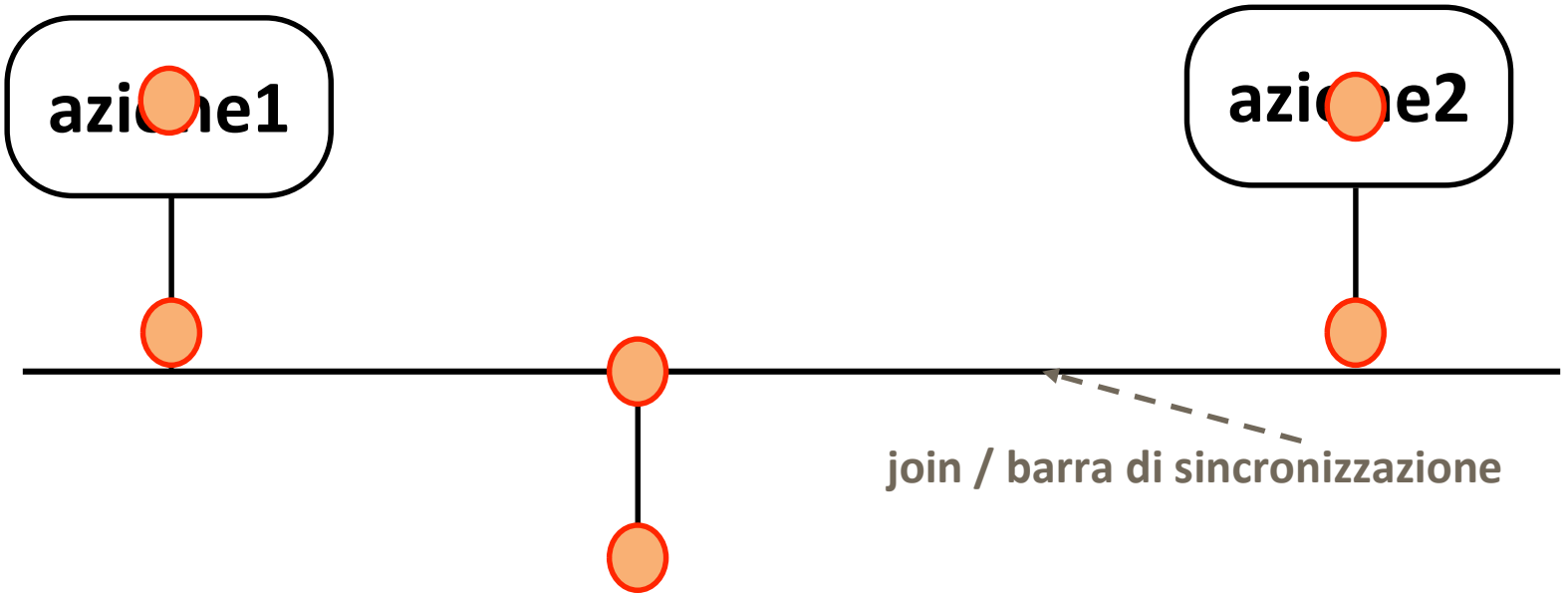
Loops



Diagrammi di attività: fork e join



Diagrammi di attività: fork e join



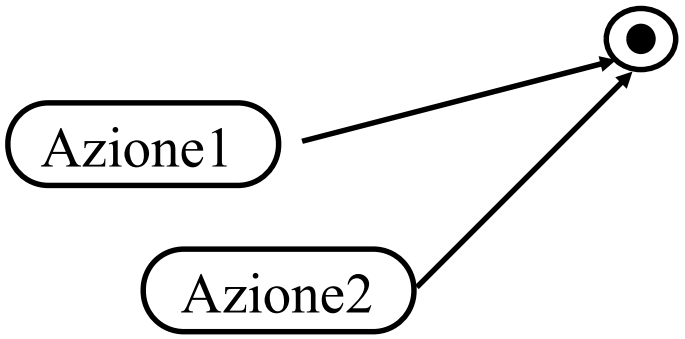
Biforcazione e ricongiunzione / fork and join

- Token game:
  - La fork moltiplica i token:
    - Dato un token in ingresso, ne "produce" uno per ogni freccia uscente
  - La join li consuma:
    - Si attende un token per ogni freccia entrante
    - Si consumano tutti e ne esce solo uno
- Non è necessaria una join per ogni fork

Nodo di fine attività

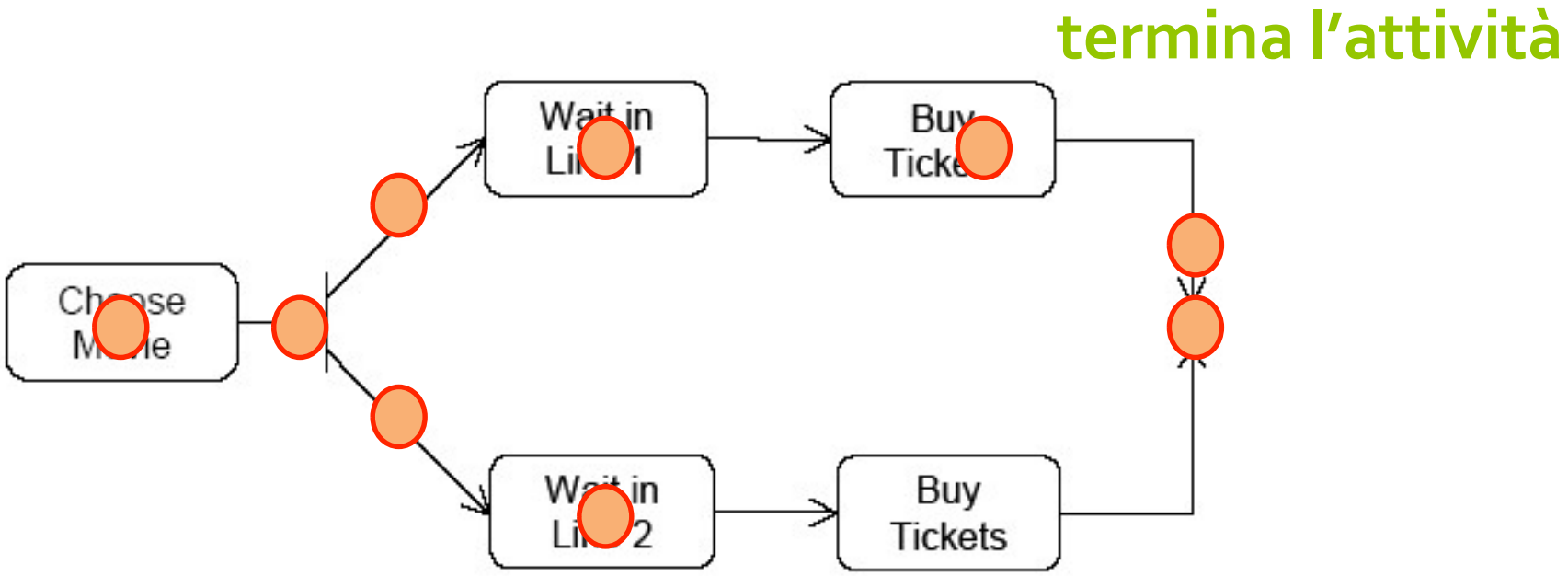
- Se un token raggiunge un nodo di fine attività , l'intera attività è terminata

- Permettiamo più archi entranti su un nodo di fine attività o di fine flusso (e solo su questi)
  - La semantica è: il primo token che arriva termina l'attività



Nodo di fine attività

- il primo che compra i biglietti termina l'attività

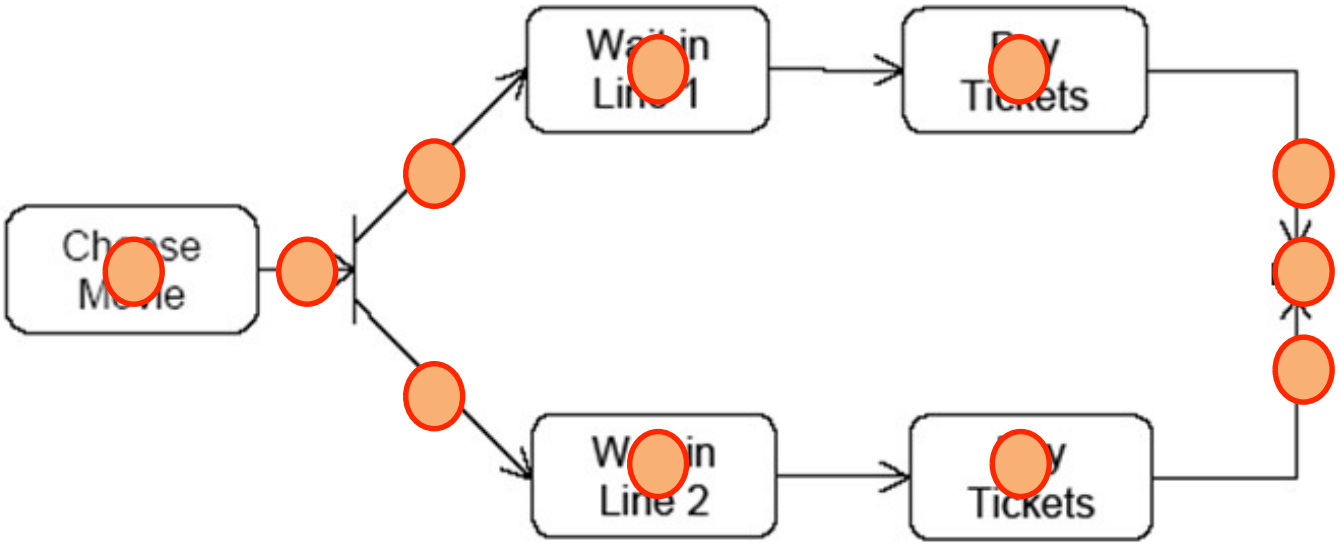


Nodo di fine flusso

- Serve per terminare un execution path non tutta l' attività .

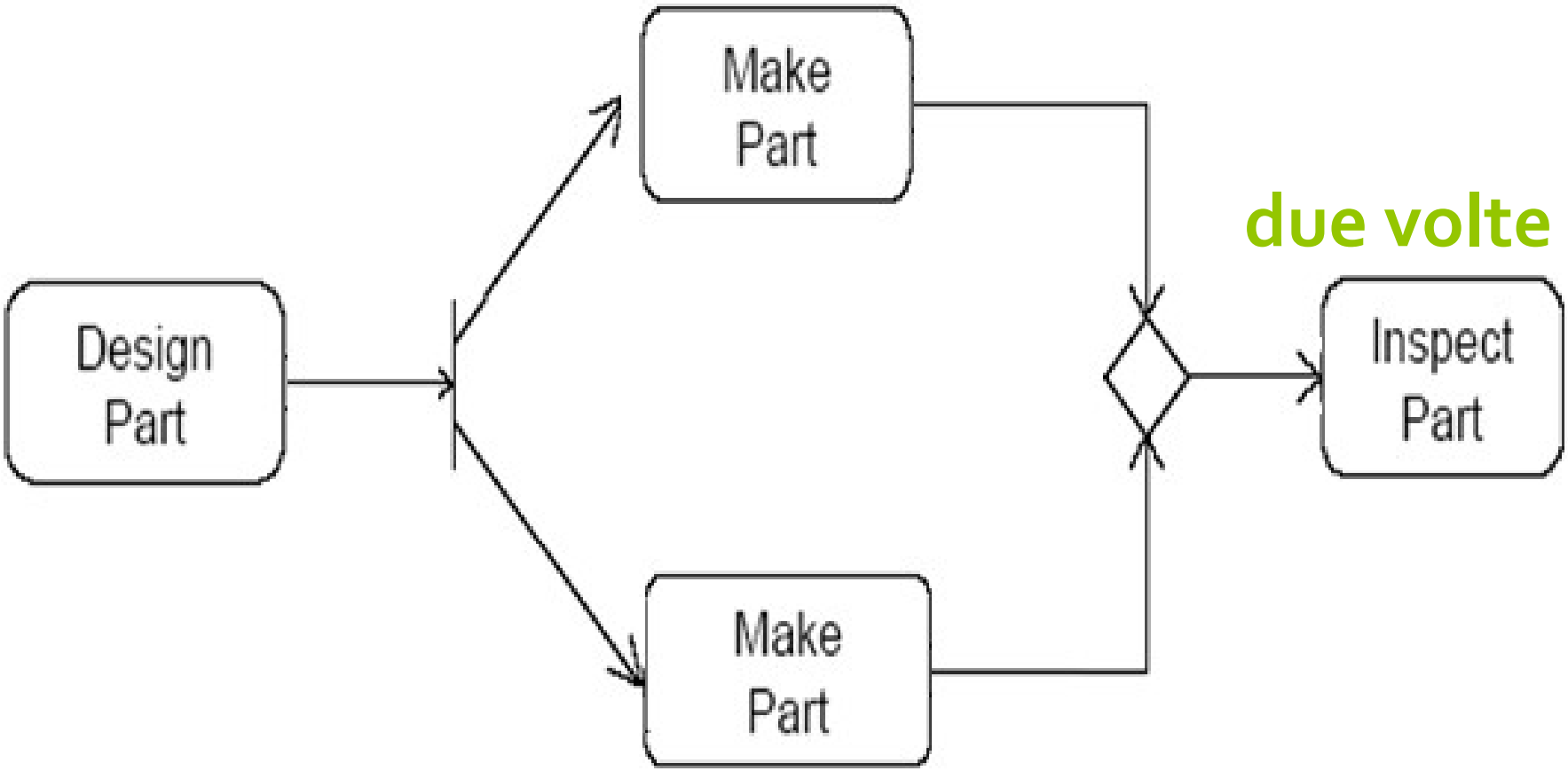
Nodo di fine flusso

- il primo che compra i biglietti non termina l'attività
- Vengono presi i biglietti in entrambe le code

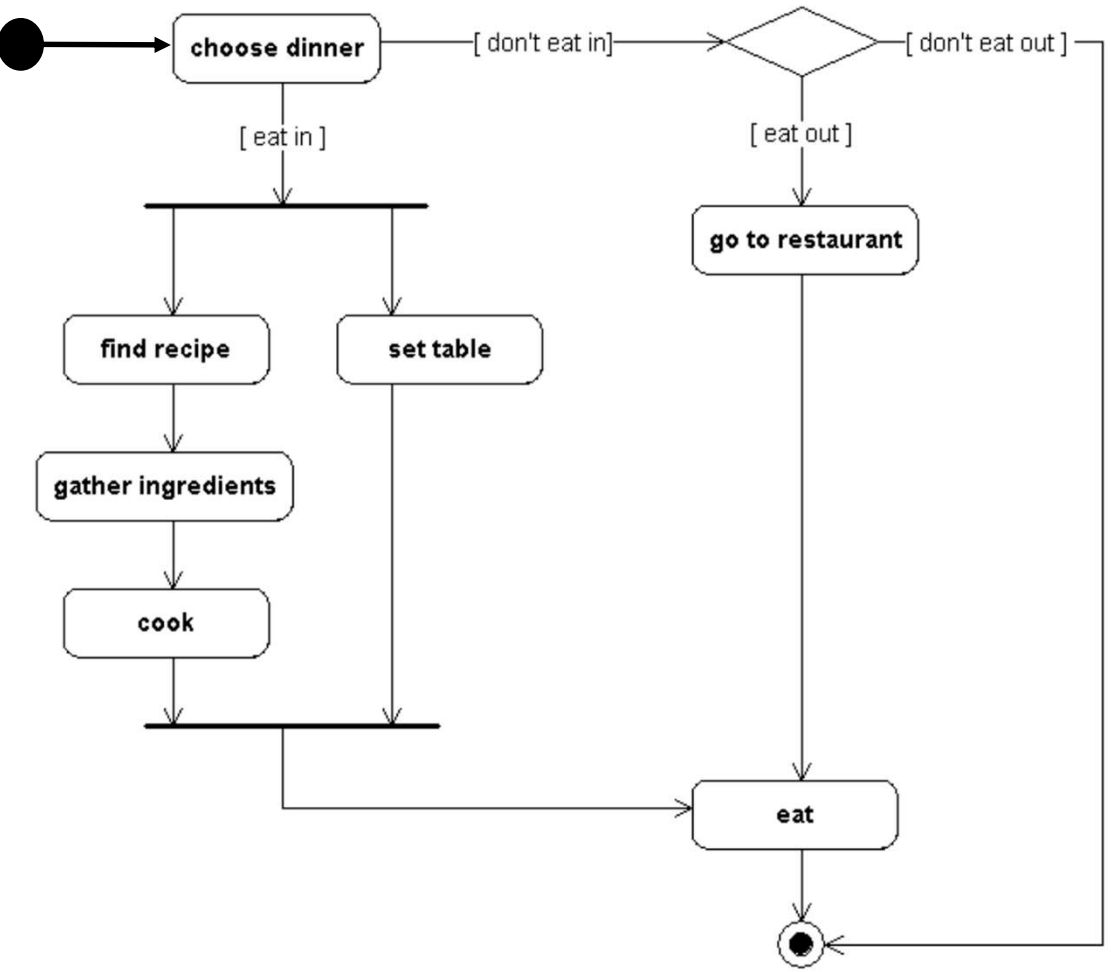


non termina l'attività, solamente si consuma il token

Fork e merge: possibile ma azioni eseguite due volte

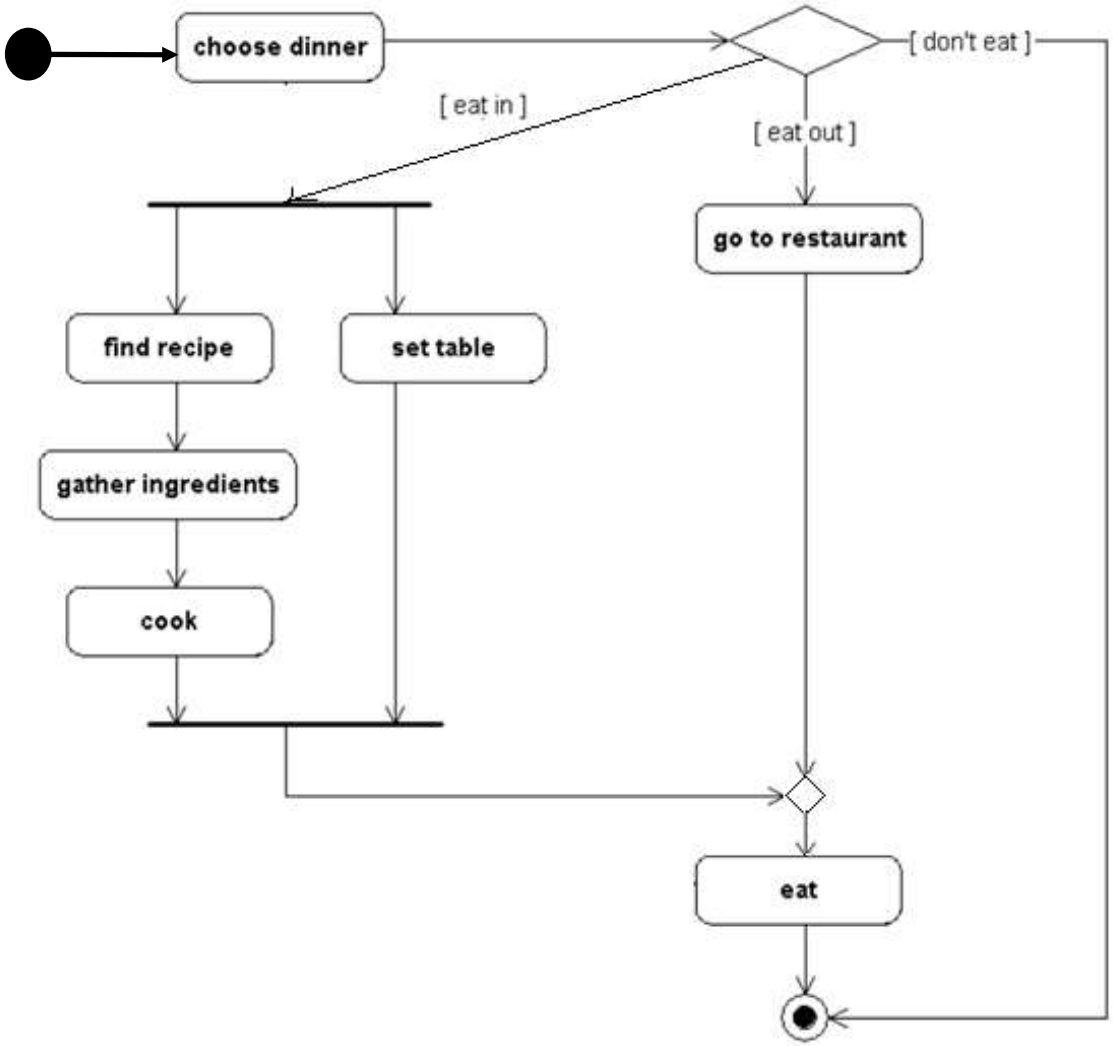


Esempio preso da web, interessante perché sbagliato



Anche se UML permette frecce multiple entranti/uscenti in/da un nodo, se ne sconsiglia (vieta in questo corso) assolutamente l'uso: la semantica UML in questo caso è quella della fork/join, ma poi è facile sbagliarsi e disegnare diagrammi come questo che vanno in deadlock. Infatti eat attende due token che non possono mai arrivare.

Diagramma corretto



Prima di eat serve un modo fusione e dopo choose dinner un nodo decisione.

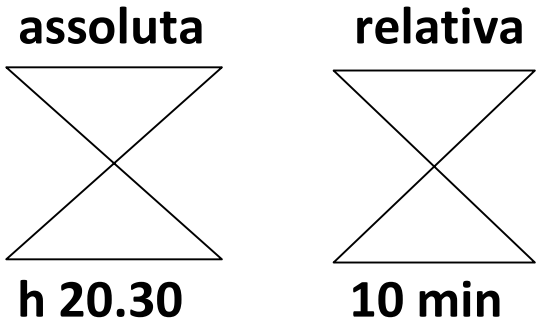
Sono tollerate due frecce entranti nello stato finale.

Segnali ed Eventi

- Accettazione di evento esterno

- Invio di un segnale

- Accettazione di evento temporale

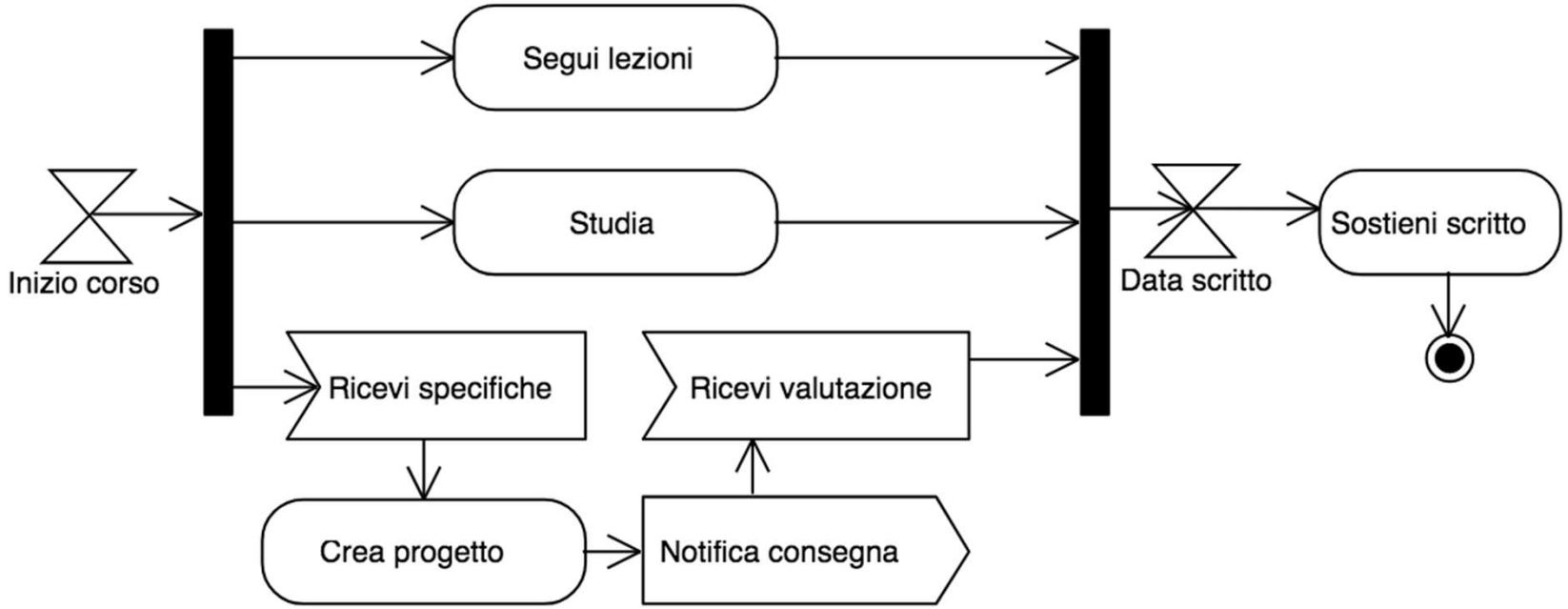


Nodi specializzati che gestiscono l'invio e la ricezione di segnali.

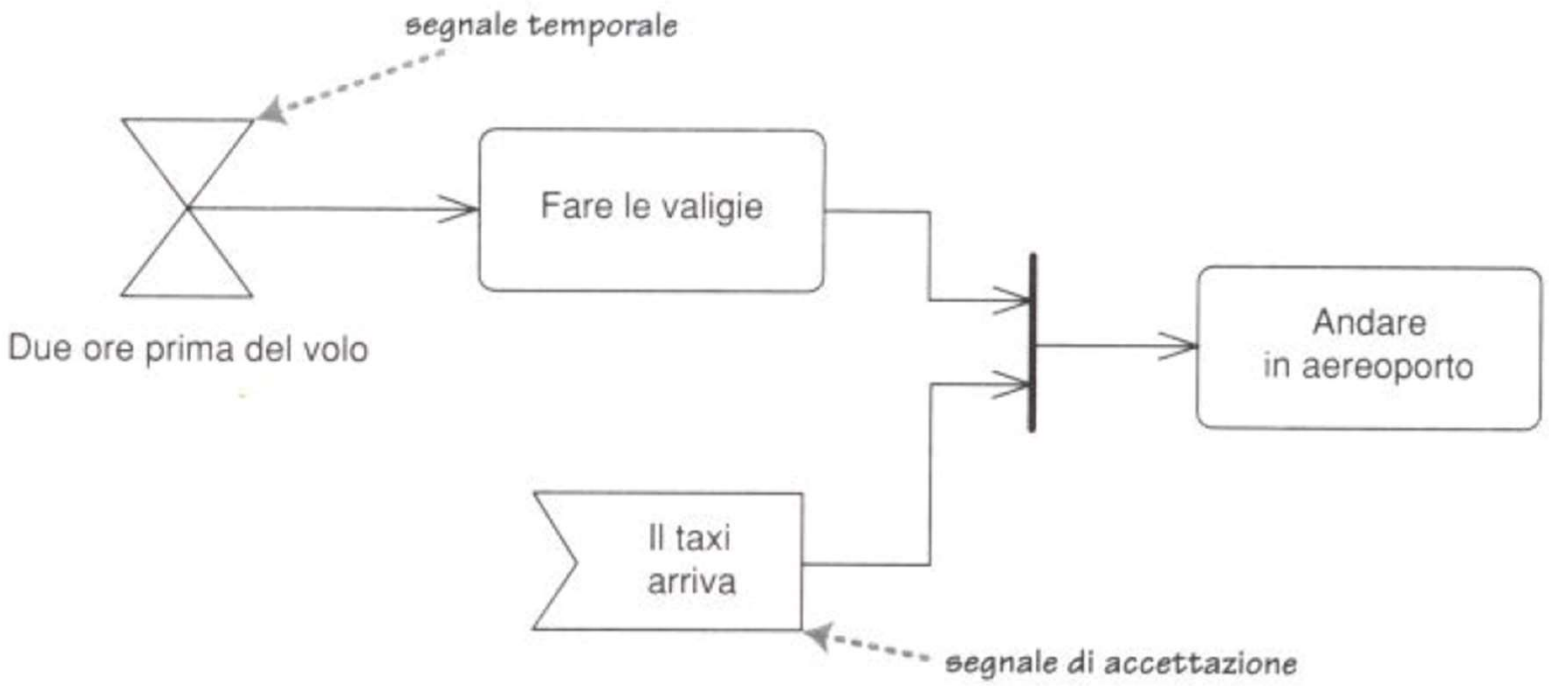
L'invio di segnali è **asincrono** e **non blocca l'attività**.

Accettazione evento esterno o accettazione di evento temporale

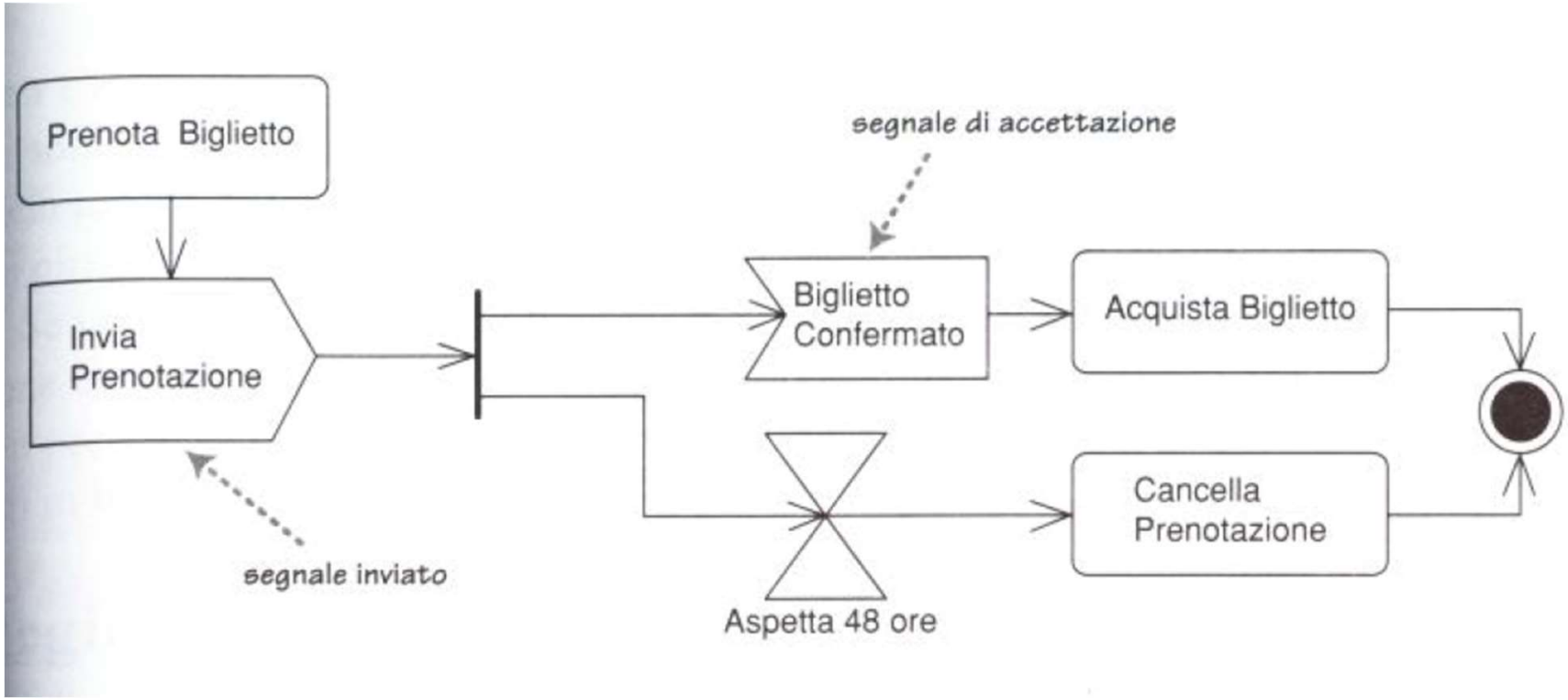
- Per accettazione evento esterno (analogo per accettazione eventi temporali) : arco entrante non necessario
- Se assente, quando arriva l'evento, si genera un token
- Se presente, l'azione è abilitata quando arriva il token e si attende l'evento esterno per farlo transitare



Esempio


segnale temporale
Fare le valigie
Due ore prima del volo
Andare
in aereoporto
Il taxi
arriva
segnale di accettazione

Esempio di time-out

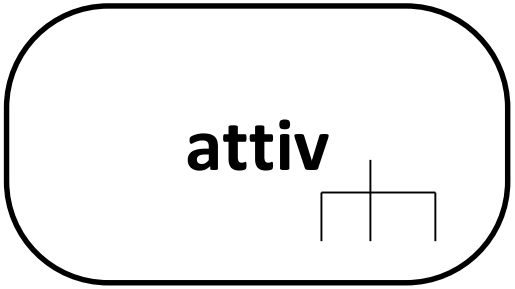


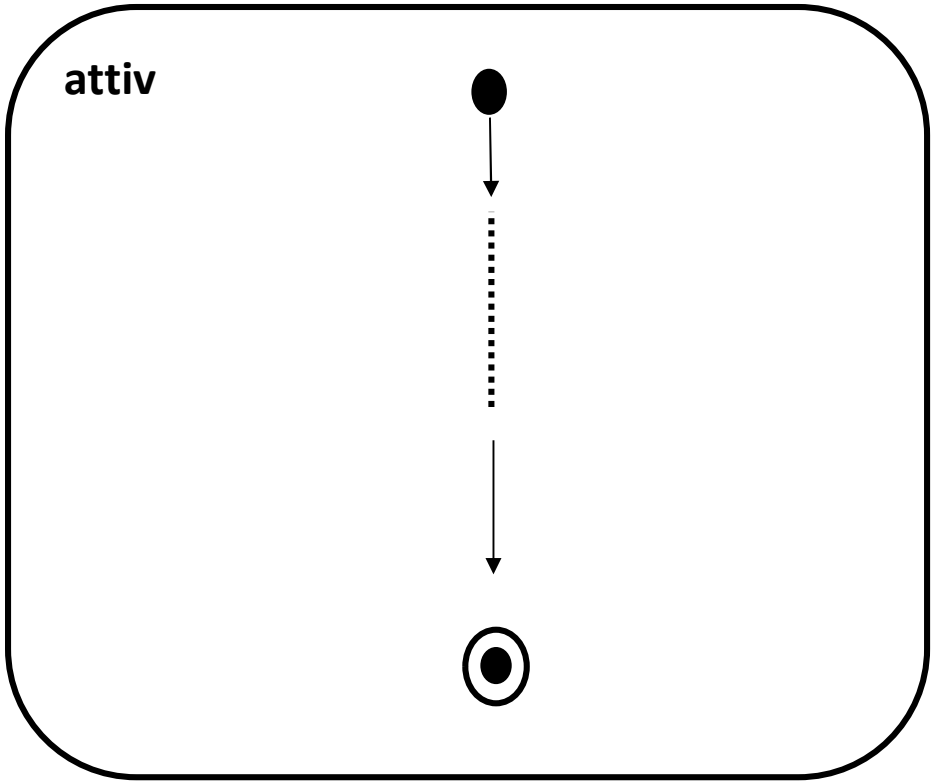
Accettazione di eventi esterni e invio segnali vs azioni

- Quando usare un' azione e quando usare accettazione di eventi esterni o invio segnali:
  - Si usa un' azione quando è effettuata dal classificatore/insieme di classificatori di cui si sta descrivendo il comportamento
  - I secondi si usano quando si comunica con una entità esterna

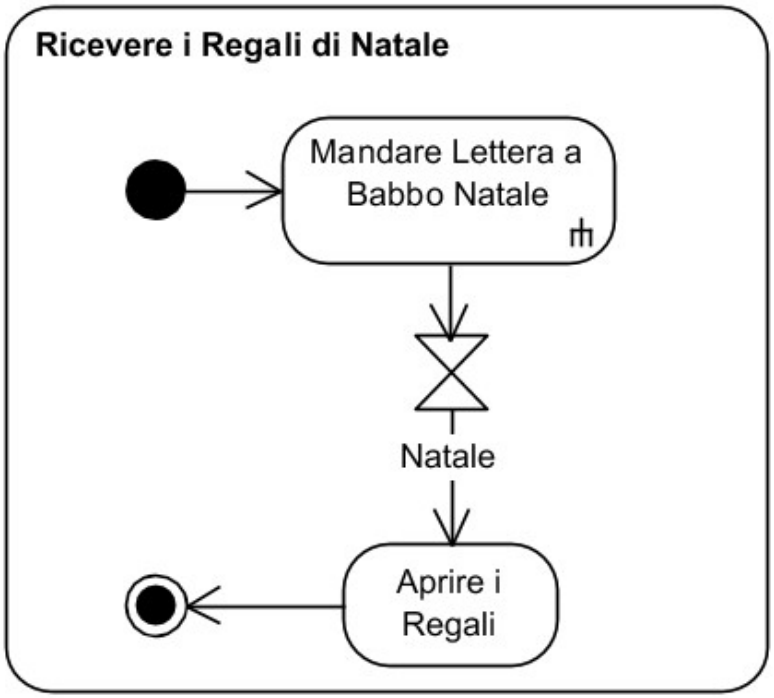
SottoAttività

- Un'azione può includere (chiamare) un'altra attività (secondaria) :
  - si usa il "rastrello" (rake) per dire che l'azione include una sotto-attività
  - Si descrive la sotto-attività in un diagramma a parte
  - Migliora il riuso e la leggibilità



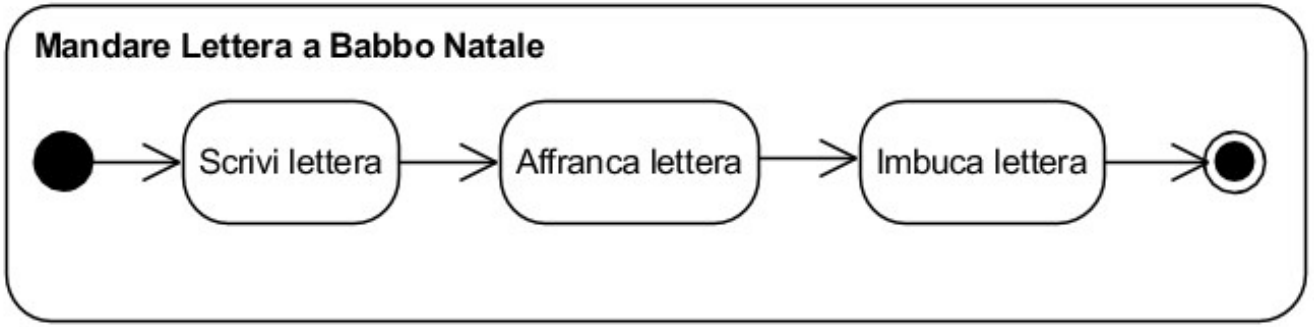


SottoAttività

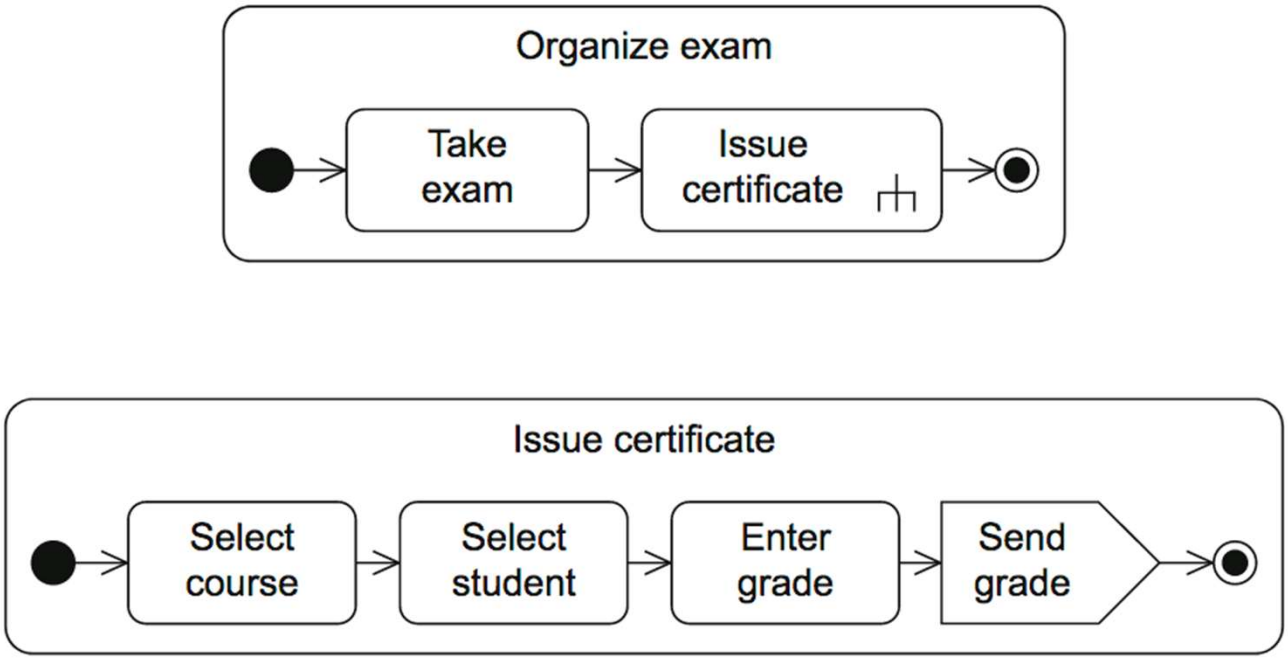


- Mandare Lettera a Babbo Natale è lasciata astratta in un diagramma (Ricevere i Regali di Natale), rappresentata come un solo nodo, il rastrello dice che è descritta in un altro diagramma

- Visione bottom-up: si definisce una attività (Mandare Lettera a Babbo Natale) e poi la si riferisce in un altro diagramma (Ricevere i Regali di Natale) usando il rastrello

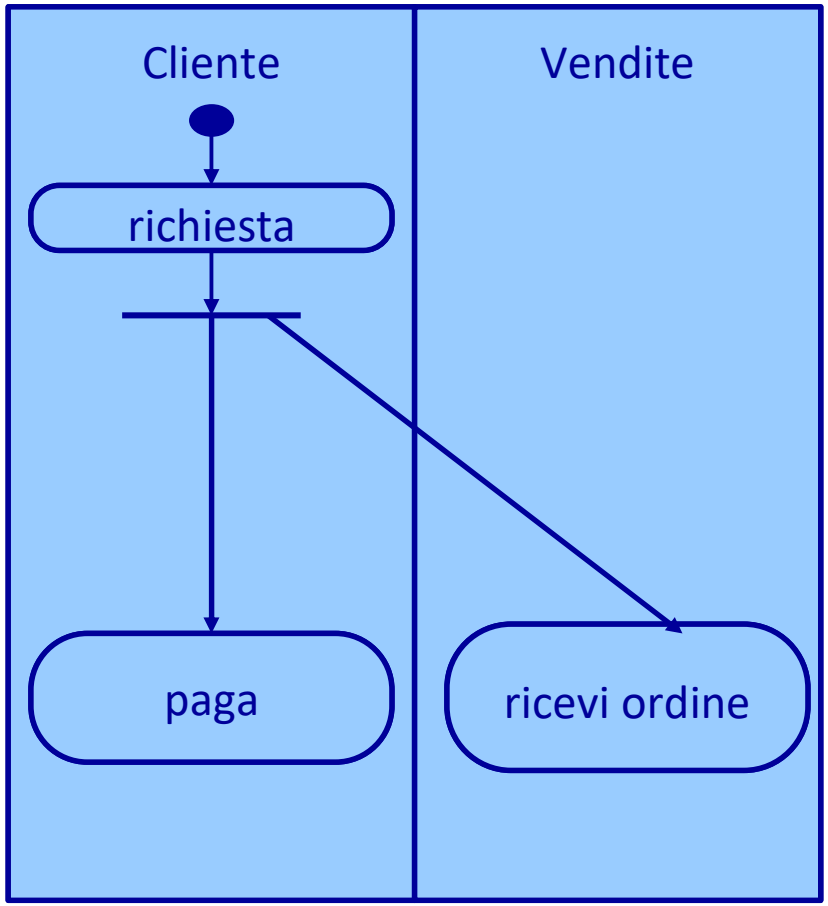


Esempio

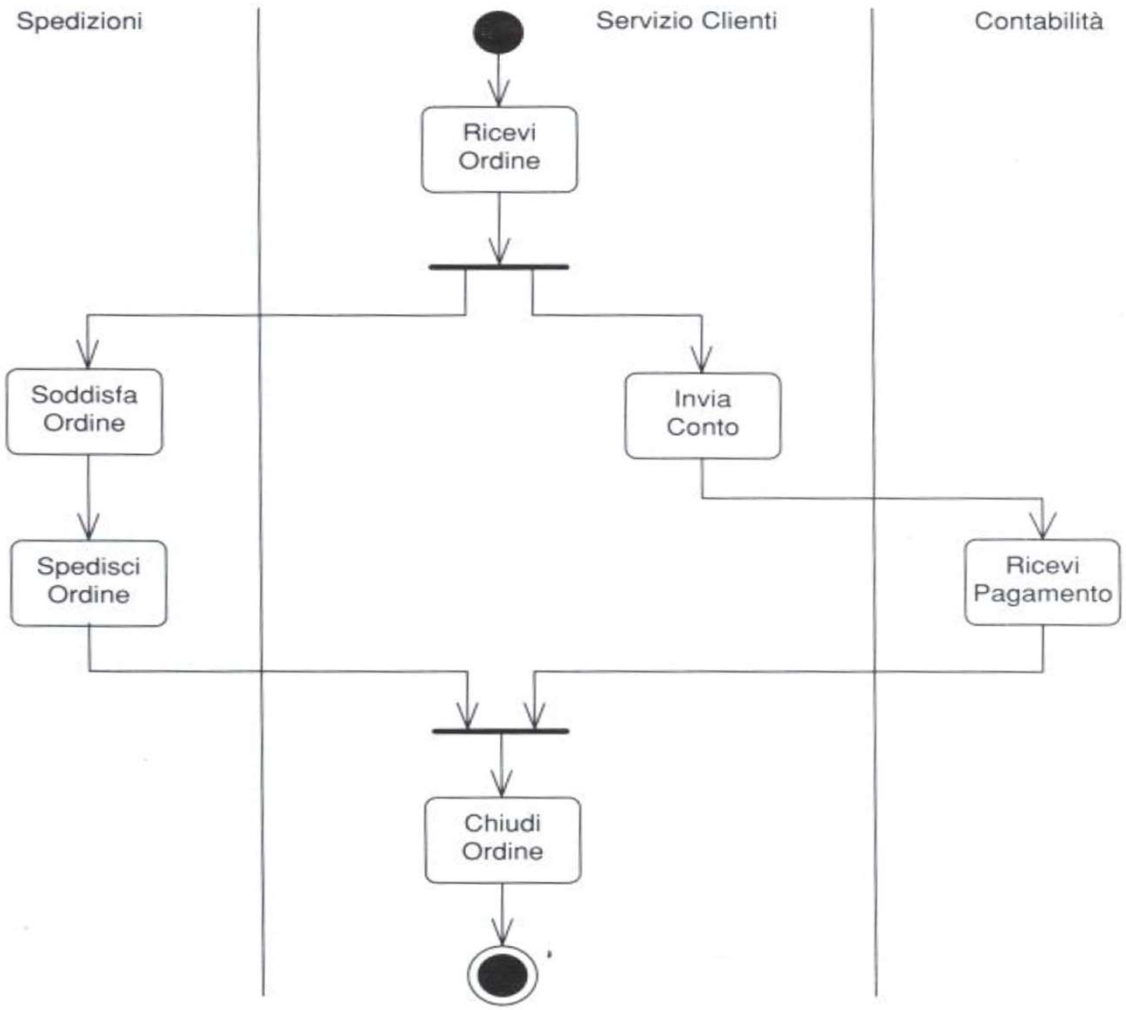

Organize exam
Take exam
Issue certificate
Issue certificate
Select course
Select student
Enter grade
Send grade

Partizioni

- Una *partizione*
  - per dividere le azioni in gruppi
  - Spesso corrisponde alla divisione in unità operative in un modello di business.
- Permettono di
  - assegnare la responsabilità delle azioni



Esempio


Spedizioni
Servizio Clienti
Contabilità
Ricevi Ordine
Soddisfa Ordine
Invia Conto
Spedisci Ordine
Ricevi Pagamento
Chiudi Ordine

Syllabus

UML@Classrom:

- 7.1 (senza parametri, precondizioni e post condizioni)
- -7.2(senza object flow wedge)
- -7.3 (senza guardie,weight edge,connettori,decision behaviour e con diversa semantica delle scelte)
- 7.5
- 7.7