UML (parte seconda)

Alberto Gianoli

Univ. Ferrara - Corso di Ingegneria del Software

Diagramma di collaborazione

- è un diagramma di interazione: rappresenta un insieme di oggetti che collaborano per realizzare il comportamento di uno scenario di un caso d'uso
- a differenza del diagramma di sequenza, mostra i legami (link) tra gli oggetti che si scambiano messaggi, mentre la sequenza di tali messaggi è meno evidente
- può essere utilizzato in fasi diverse (analisi, disegno di dettaglio)

Diagramma di collaborazione (cont)

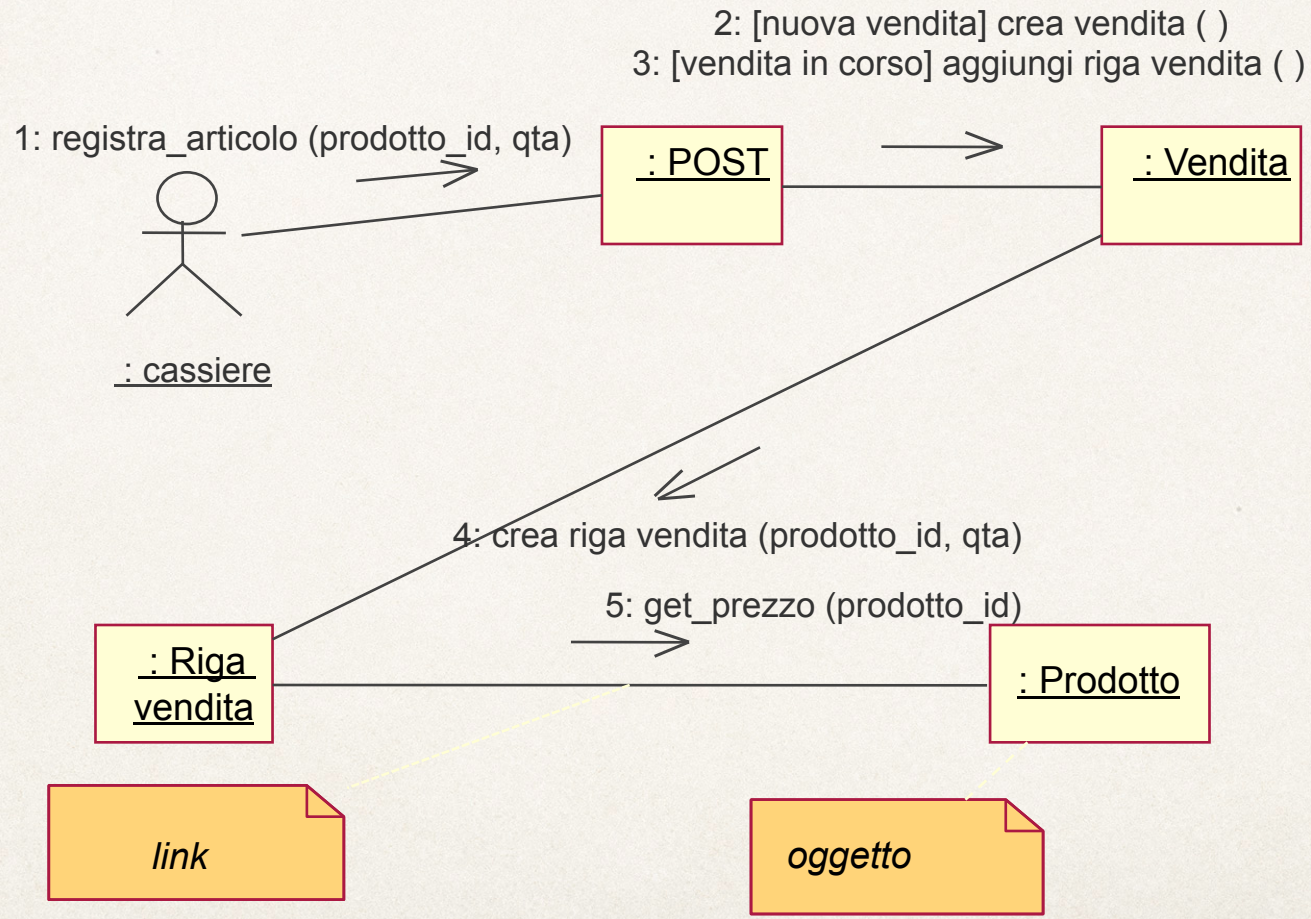


Diagramma transizioni di stato

- serve a modellare il ciclo di vita degli oggetti di una singola classe
- mostra gli eventi che causano la transizione da uno stato a un altro, le azioni eseguite in seguito a un determinato evento
- quando un oggetto si trova in un certo stato può essere interessato ad alcuni eventi e non ad altri
- va utilizzato solo per le classi che presentano un ciclo di vita complesso e segnato da una successione ben definita di eventi

Diagramma di stato

- rappresenta il ciclo di vita degli oggetti di una classe
- il ciclo di vita è descritto in termini di
  - **eventi**
  - **stati**
  - **transizioni di stato**
- gli eventi possono attivare delle transizioni di stato
- un evento in uno statechart corrisponde ad un messaggio in un sequence diagram
- uno stato è costituito da un insieme di "valori significativi" assunti dagli attributi dell'oggetto che ne influenzano il comportamento

Diagramma di stato (cont)

- Due stati "speciali", detti **pseudostati**:
  - lo **stato iniziale**
  - lo **stato finale**
- Un oggetto può non avere uno stato finale (non viene mai distrutto)

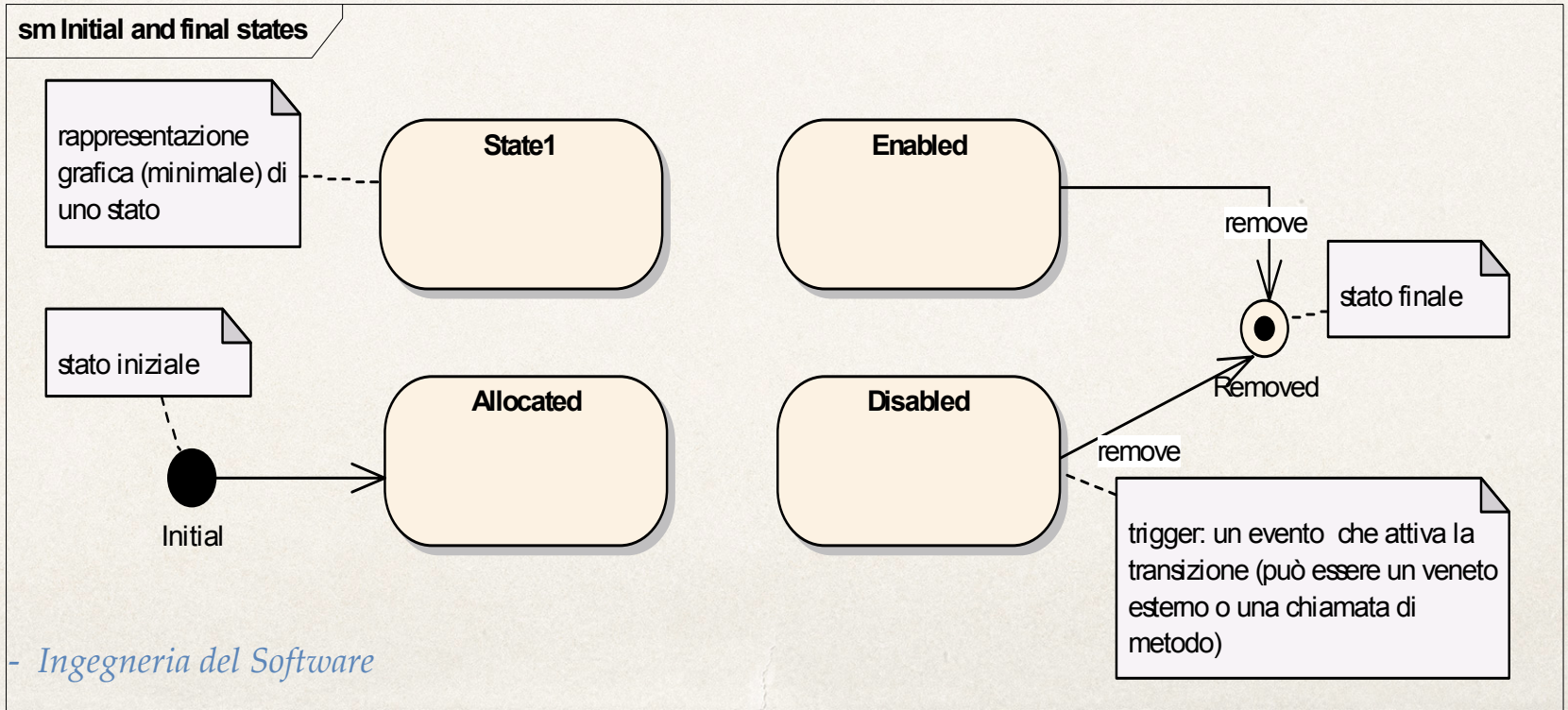


Diagramma di stato (cont)

- Un evento può essere:
- l'**invocazione sincrona** di un metodo (una "call")
- la ricezione di una **chiamata asincrona** ("signal") - es: la notifica di una eccezione lanciata
- una **condizione predefinita** che diventa vera (si parla in questa caso di "change event")
- la fine di un "**periodo di tempo**" come quello impostato da un timer ("elapsed-time event")
- Un evento si può rappresentare graficamente con una freccia (transizione) etichettata con il nome del metodo o della condizione associata all'evento stesso

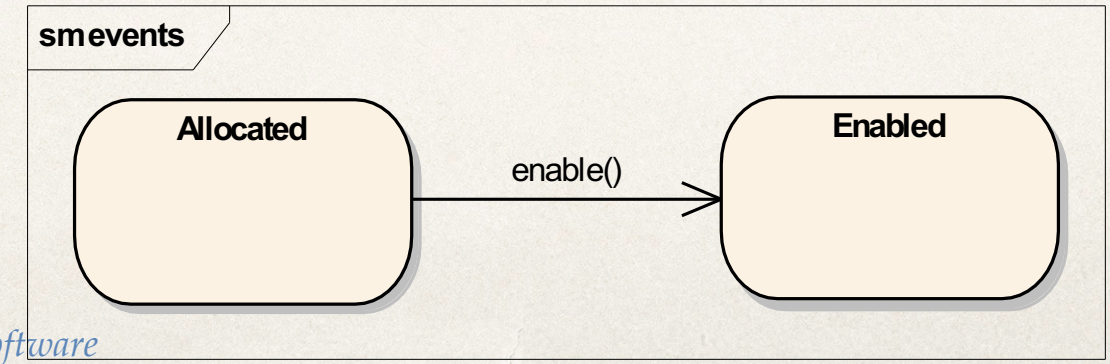


Diagramma di stato (cont)

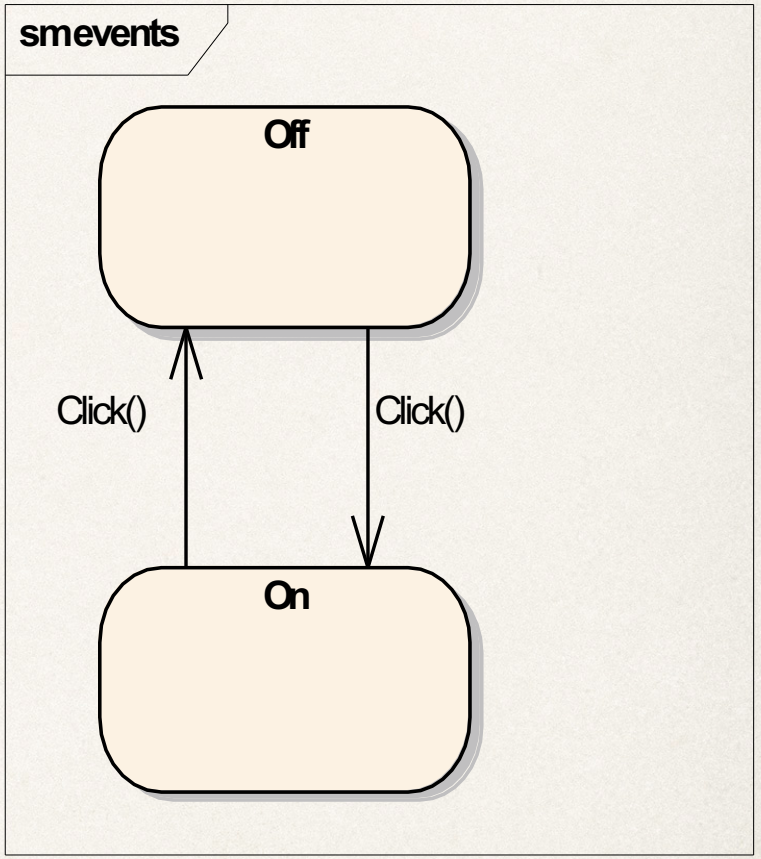
- Un evento può essere rappresentato anche mediante una espressione testuale con la seguente sintassi:
  - event-name ‘(‘ [comma-separated-parameter-list] ‘)’ [‘[‘guard-condition’]’] / [action-expression]

    dove:
    - **event-name** identifica l’evento
    - **parameter-list** definisce i valori dei dati che possono essere passati come parametro con l’evento
    - **guard-condition** determina se l’oggetto che riceve l’evento deve rispondere ad esso (cioè eseguire il metodo associato)
    - **action-expression** definisce come l’oggetto ricevente deve rispondere all’evento

Diagramma di stato (cont)

- Event + state = response

  Lo stesso evento causa diversi comportamenti in base allo stato in cui l'oggetto che riceve l'evento si trova



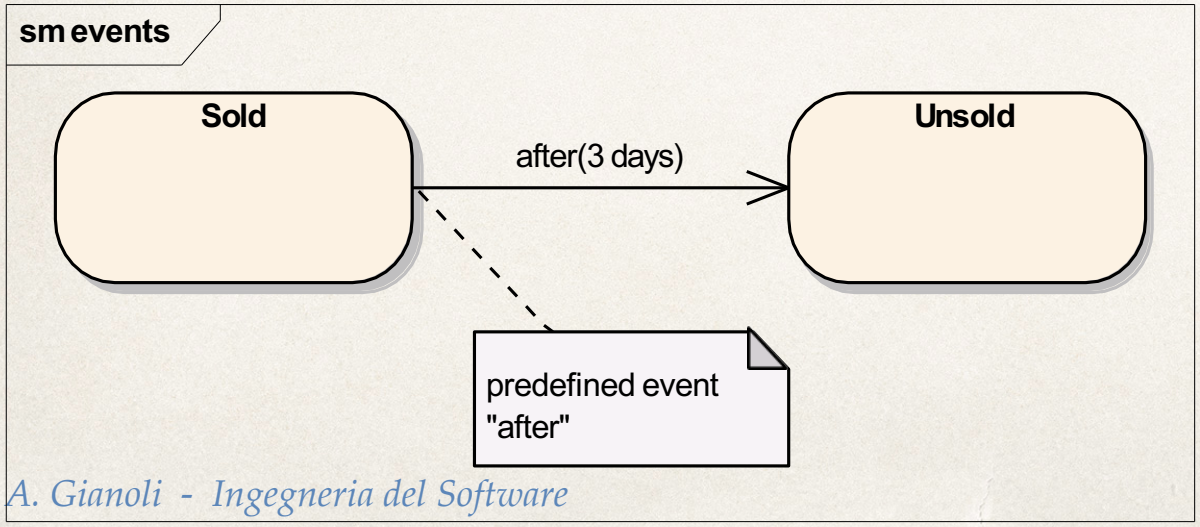
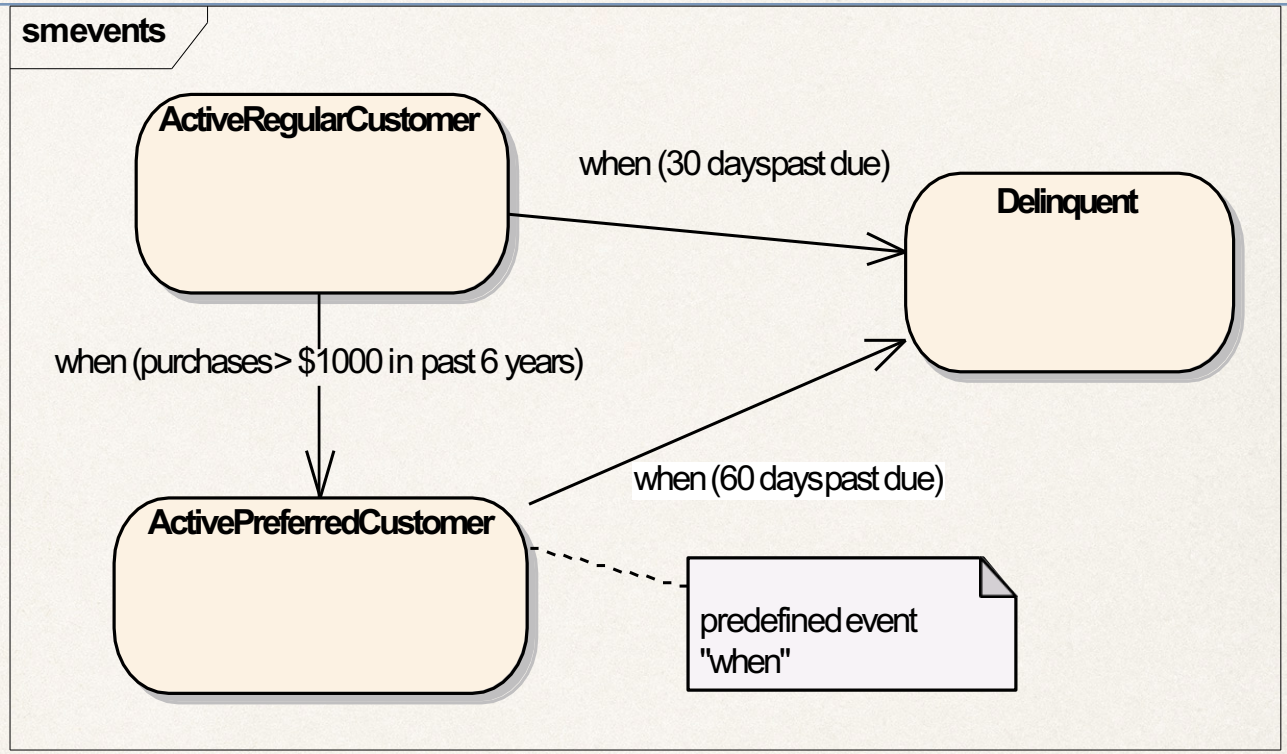


Diagramma di stato (cont)

- Change event



- Guarded event

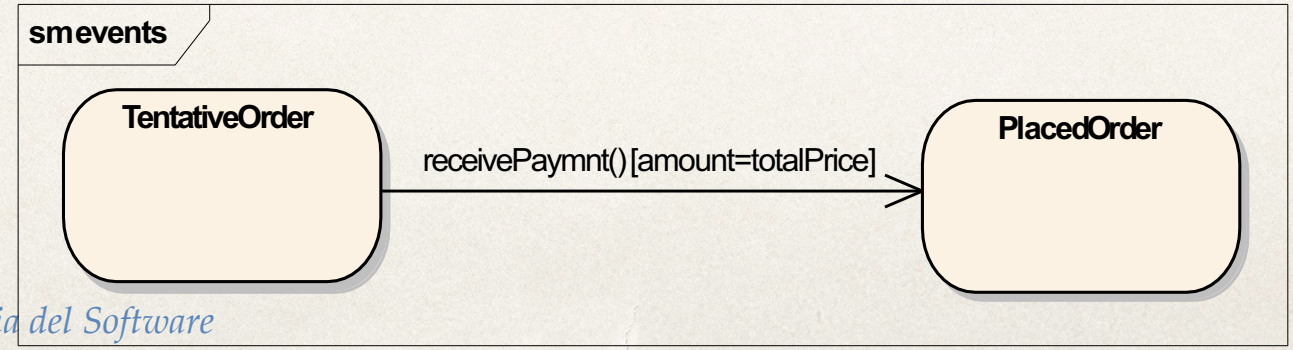
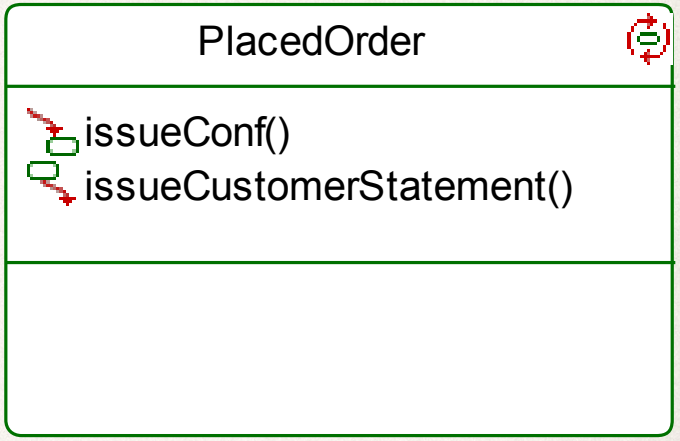
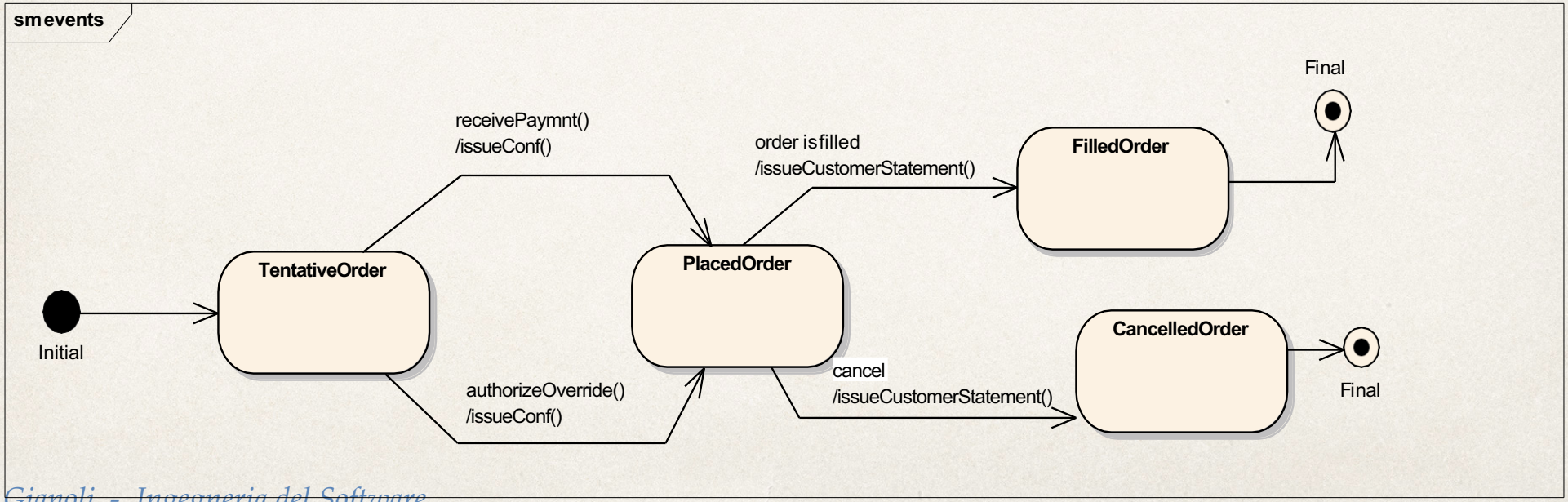


Diagramma di stato (cont)

- **entry action**: azione che viene eseguita in una transizione entrante nello stato

- **exit action**: azione che viene eseguita in una transizione uscente dallo stato





Modellare le attività

- All'interno degli stati posso essere eseguite delle azioni
- Negli statechart distinguiamo tra
  - **Azioni**: operazioni atomiche
  - **Attività**: operazioni generalmente non atomiche
  - Le azioni provocano un cambiamento di stato e quindi non possono essere interrotte
  - Le attività non alternano lo stato dell'oggetto

Modellare le attività (cont)

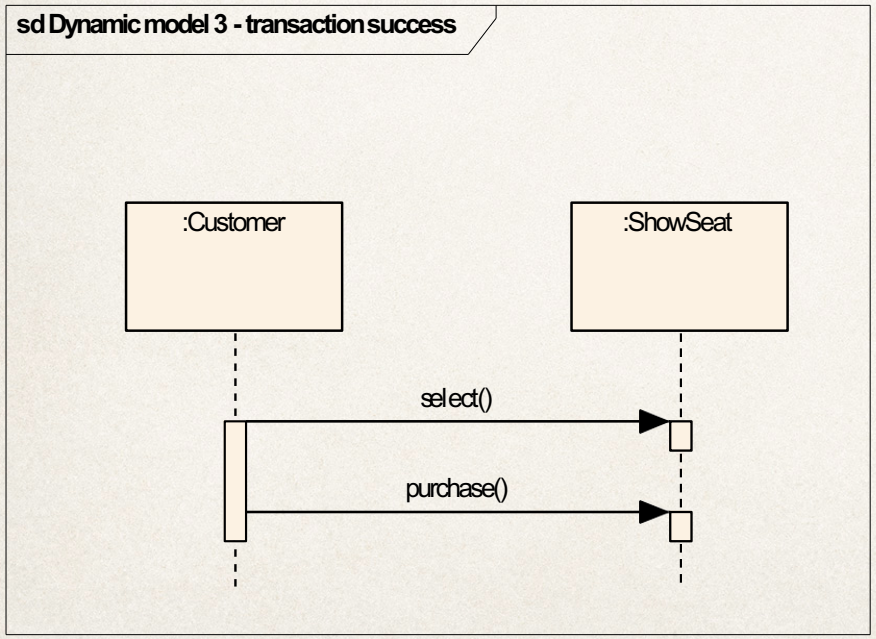
- Quando si verifica un evento associato ad una transizione, l'ordine di esecuzione è il seguente:

1. Se è in esecuzione un'attività, questa viene interrotta ("gracefully" se è possibile)
2. Si esegue l'exit action
3. Si esegue l'azione associata all'evento
4. Si esegue l'entry action del nuovo stato
5. Si inizia l'esecuzione delle eventuali attività del nuovo stato

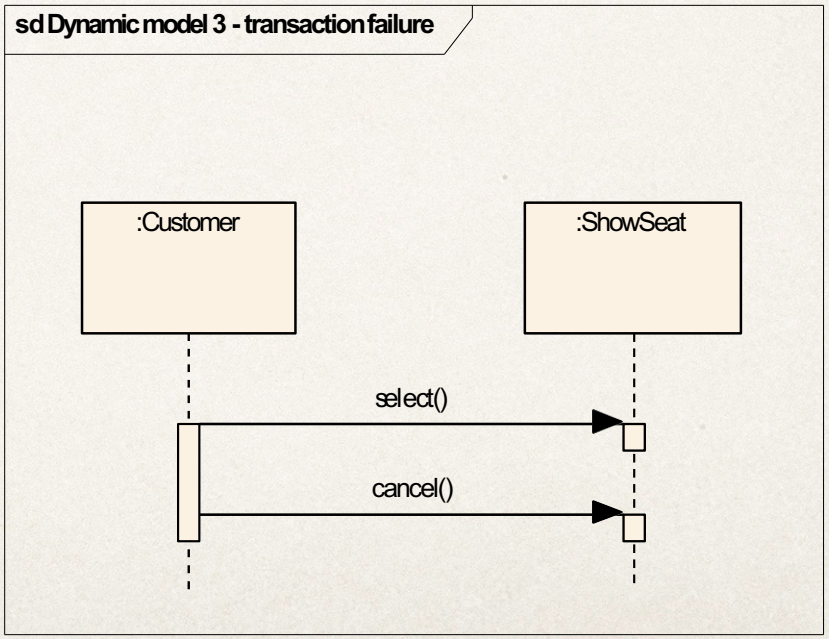
Diagrammi di stato e di sequenza

- Due scenari (sequence diagram): successo e fallimento di una transizione

(successo)



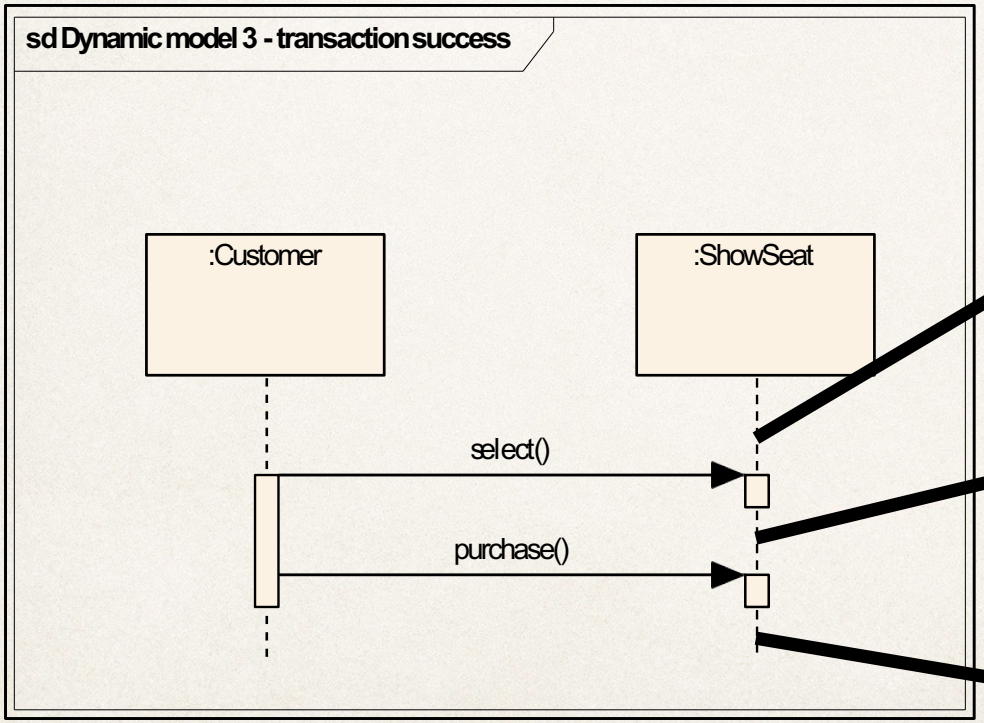
(fallimento)



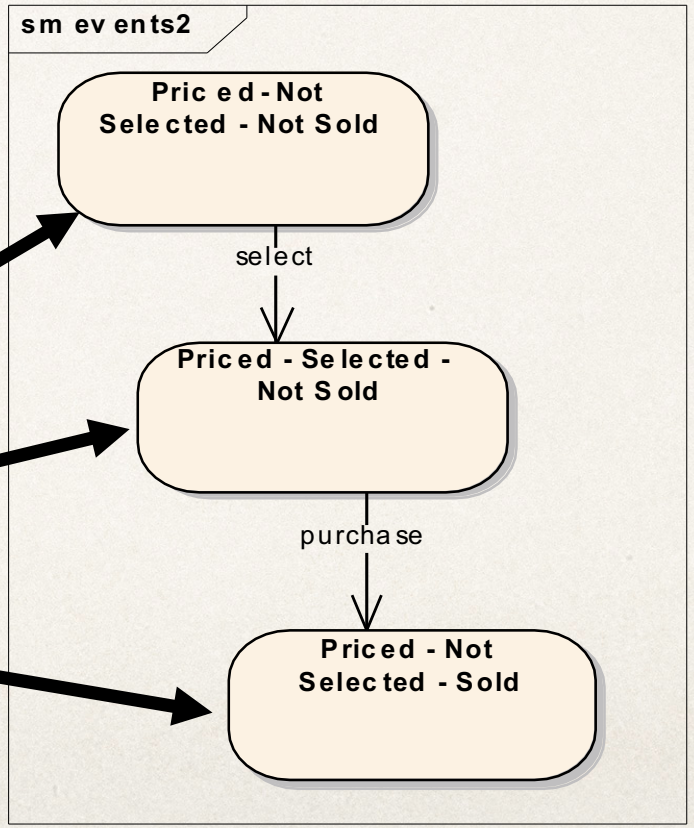
Diagrammi di stato e di sequenza (cont)

- Scenario di successo e relativo (parziale) diagramma a stati

(successo)



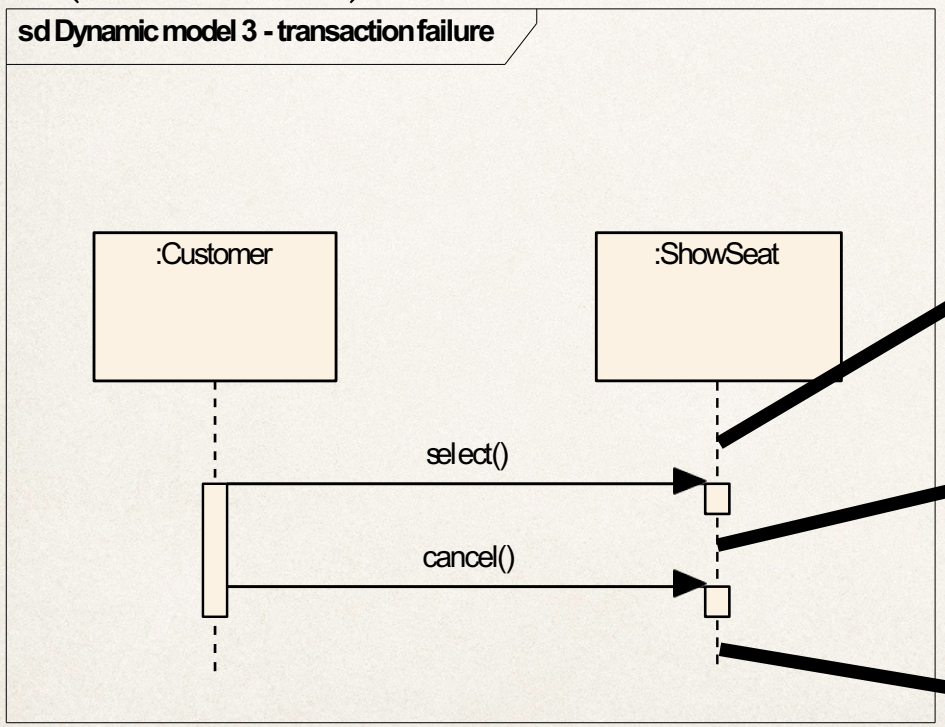
(diagramma a stati parziali)



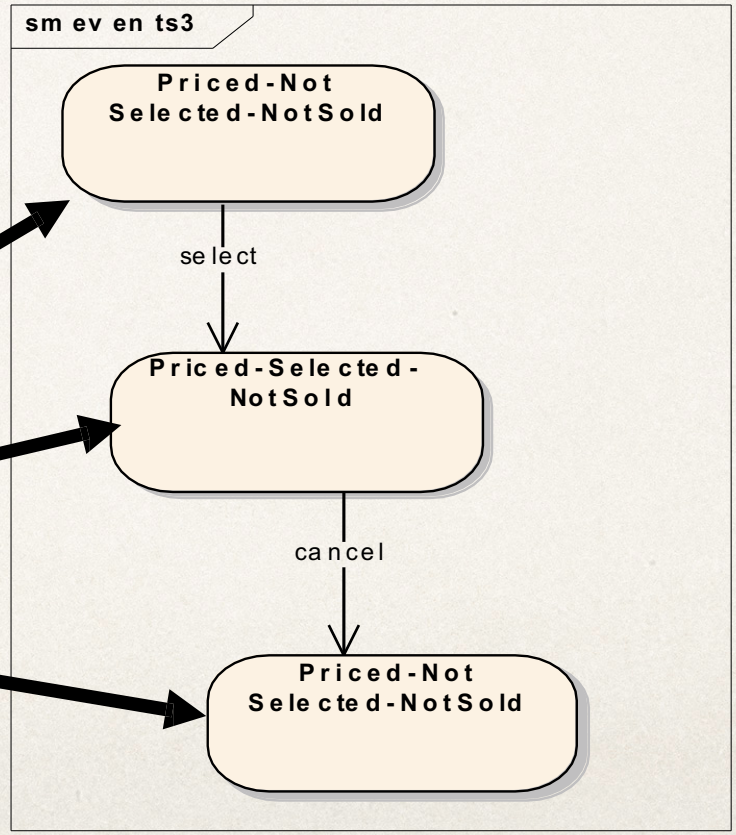
Diagrammi di stato e di sequenza (cont)

- Scenario di fallimento e relativo (parziale) diagramma a stati

(fallimento)



(diagramma a stati parziali)



Diagrammi di stato e di sequenza (cont)

- Diagramma a stati completo, relativo ai due scenari precedenti

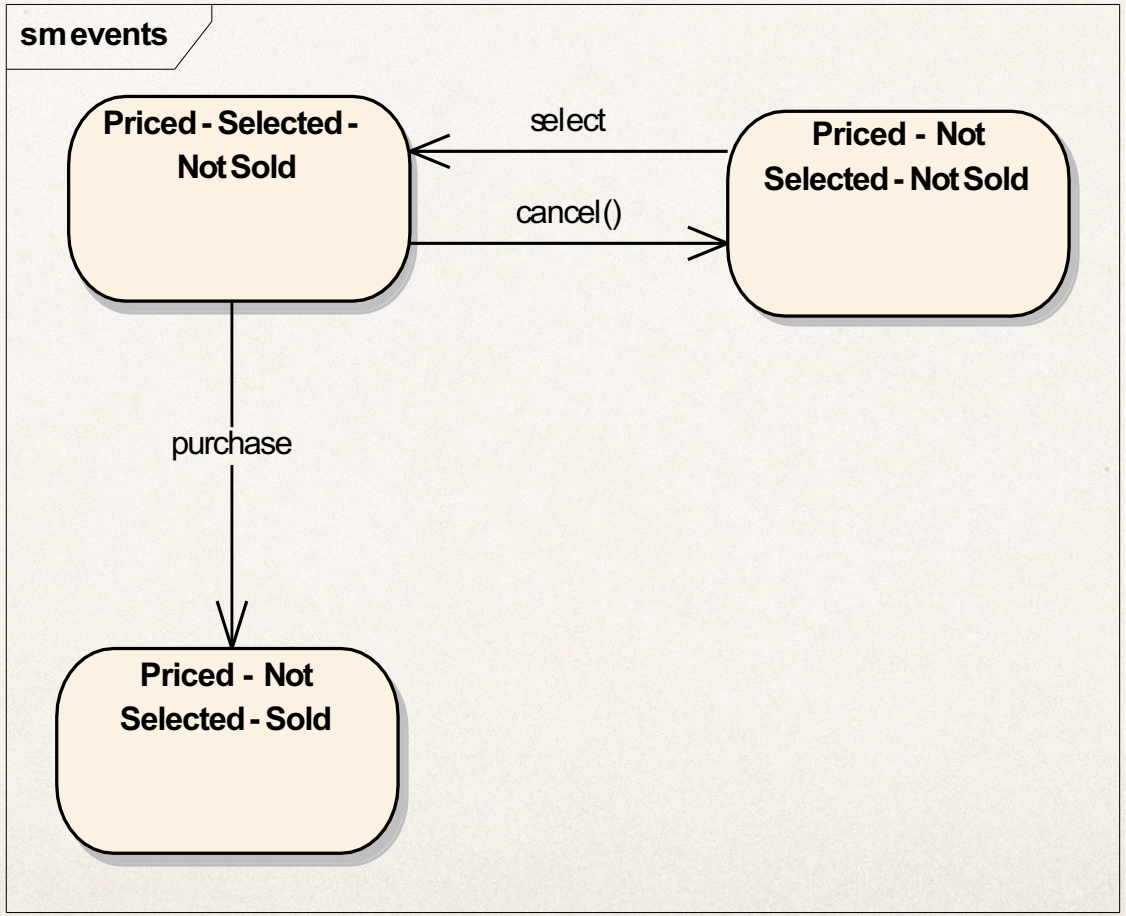
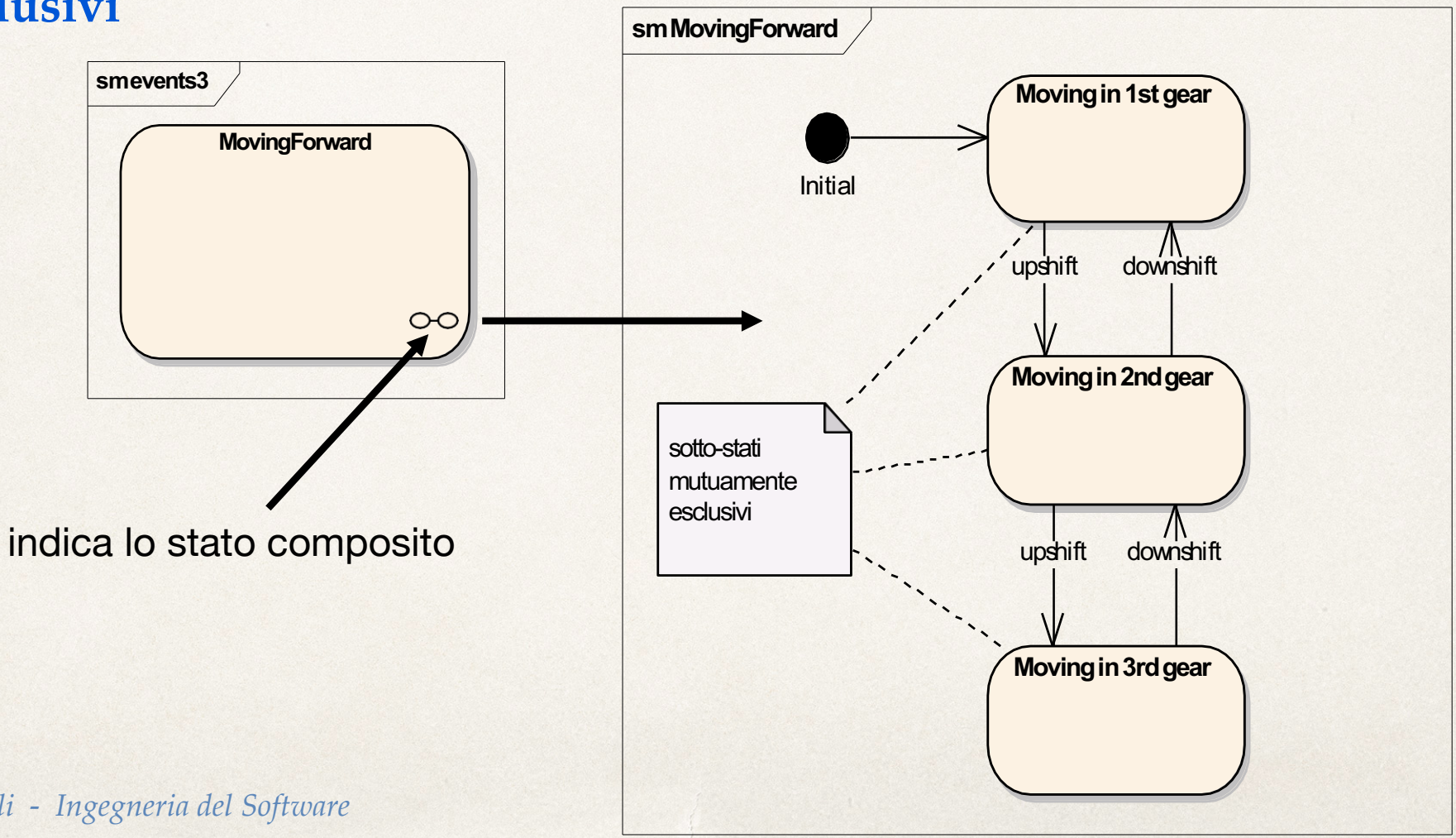


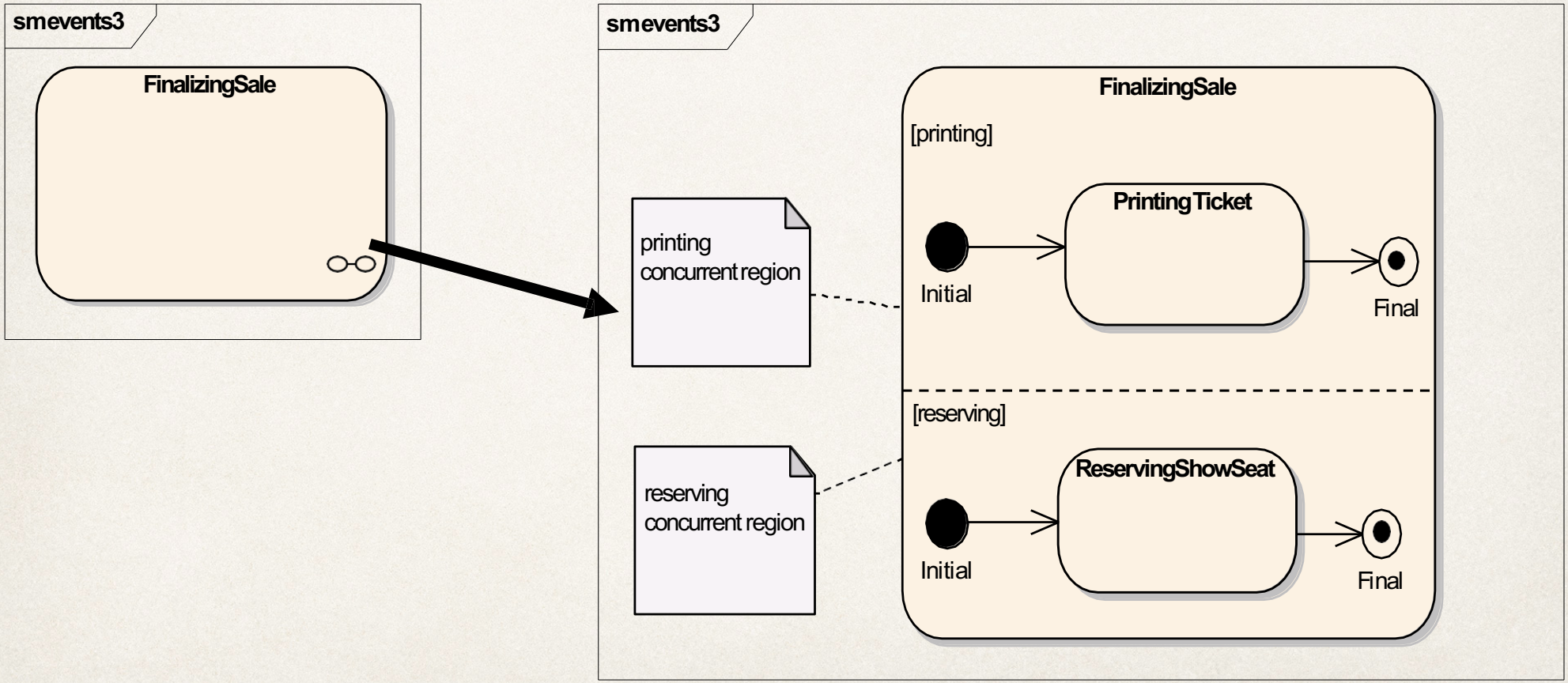
Diagramma a stati composti

- Uno stato può contenere al suo interno più **sottostati mutuamente esclusivi**



Diagrammi di stato composti (cont)

- Uno stato può contenere al suo interno **sottostati concorrenti**



Diagrammi di attività

- rappresenta sistemi di workflow, oppure la logica interna di un processo (processo di business o processo di dettagli), di un caso d'uso o di una specifica operazione di una classe
- permette di modellare processi paralleli e la loro sincronizzazione
- è un caso particolare di diagrammi di stato, in cui ogni stato è uno stato di attività

Diagramma di attività (cont)

- Un diagramma di attività mostra il **flusso di azioni** relativo ad un'attività
- Un'attività è una **esecuzione non atomica** di operazioni all'interno di una macchina a stati
- L'esecuzione di un'attività viene decomposta in azioni atomiche
- Ogni azione può o meno cambiare lo stato del sistema
- I diagrammi di attività sono spesso usati per descrivere la **logica di un algoritmo** (sono l'equivalente UML dei diagrammi di flusso)
- Graficamente un diagramma di attività è un insieme di archi e nodi

Diagrammi di attività (cont)

- Azioni (atomiche)
  - Valutazione di espressioni
  - Assegnamenti/Ritorno di un valore
  - Invocazione di un'operazione su un oggetto
  - Creazione/distruzione di un oggetto
- Nodi Attività
- Raggruppamento di azioni atomiche o di altri nodi attività
- Un'azione può essere vista come un'attività che non può essere ulteriormente decomposta
- A parte questa differenza, i due concetti sono rappresentati mediante lo stesso simbolo grafico

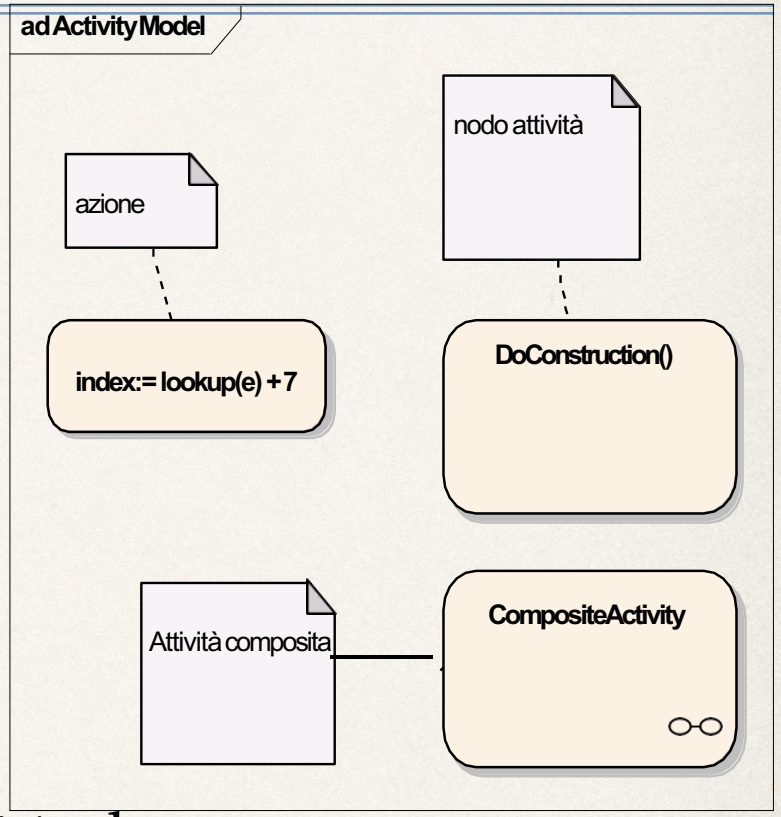


Diagramma di attività (cont)

- Quando un'azione o un'attività viene completata, il flusso di controllo passa al nodo azione (attività) immediatamente successivo
- Il flusso di controllo viene specificato mediante frecce che collegano due nodi (attività o azione)
- Il flusso mostrato in figura è quello più semplice: il **flusso sequenziale**

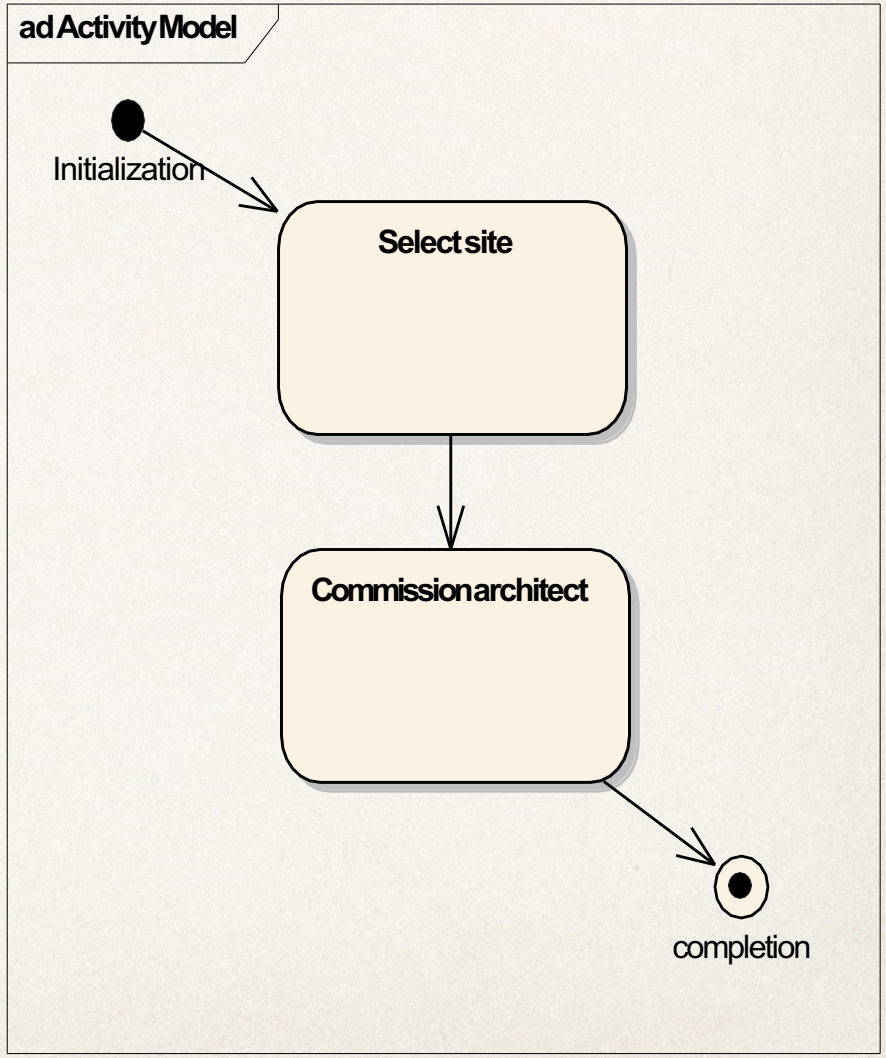


Diagramma di attività (cont)

- Un altro tipo di flusso possibile è il **branch**
- Un branch è rappresentato da un diamante
- Ogni branch ha:
  - Un flusso entrante
  - Due o più flussi uscenti
  - Una condizione logica (talvolta implicita) che determina quale dei flussi uscenti verrà eseguito da una particolare esecuzione
- Quando due flussi si riuniscono, è possibile usare ancora il simbolo di diamante: in questo caso viene detto **merge**
- Ogni merge ha almeno due flussi entranti e un flusso uscente

Diagramma di attività (cont)

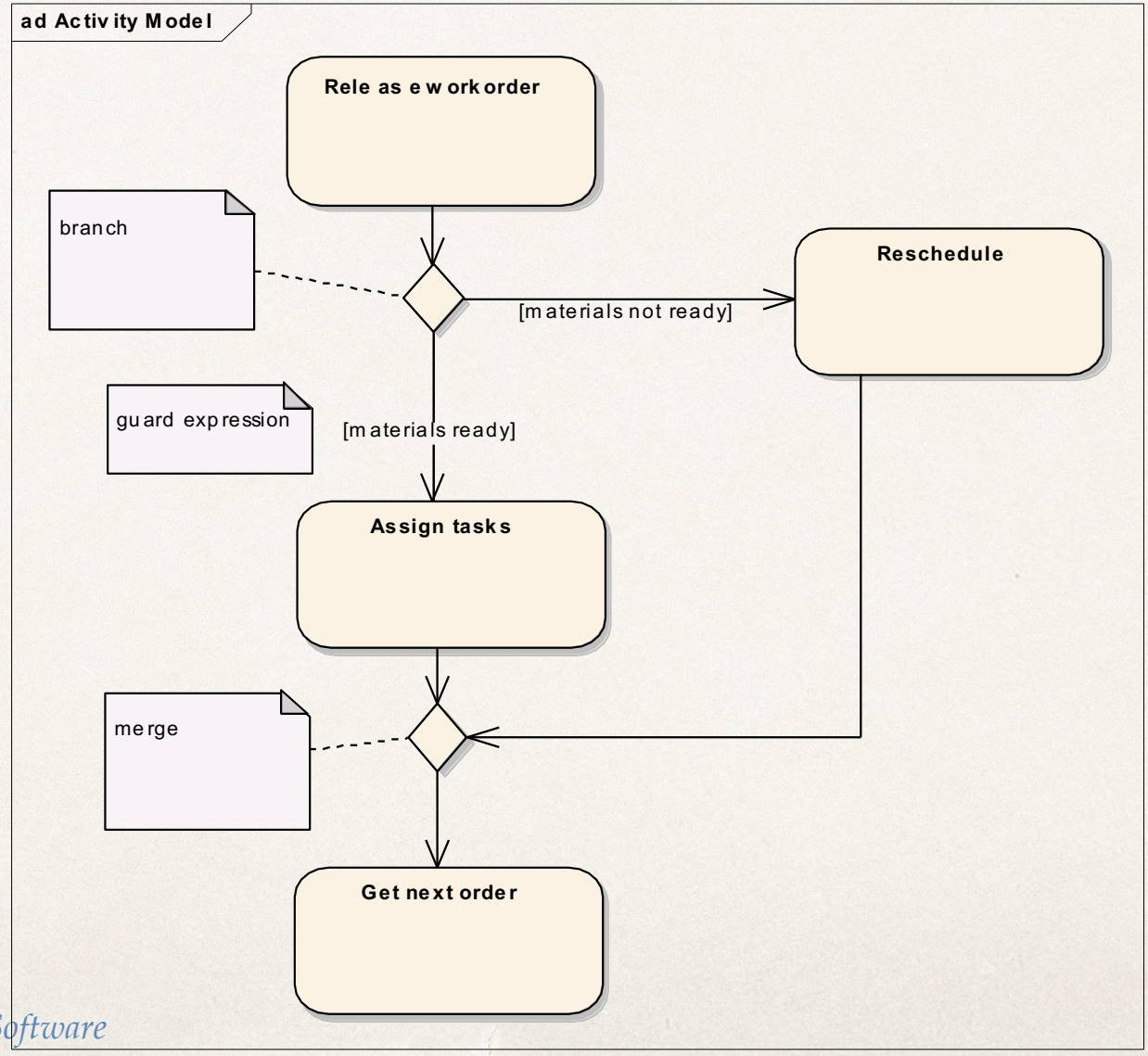
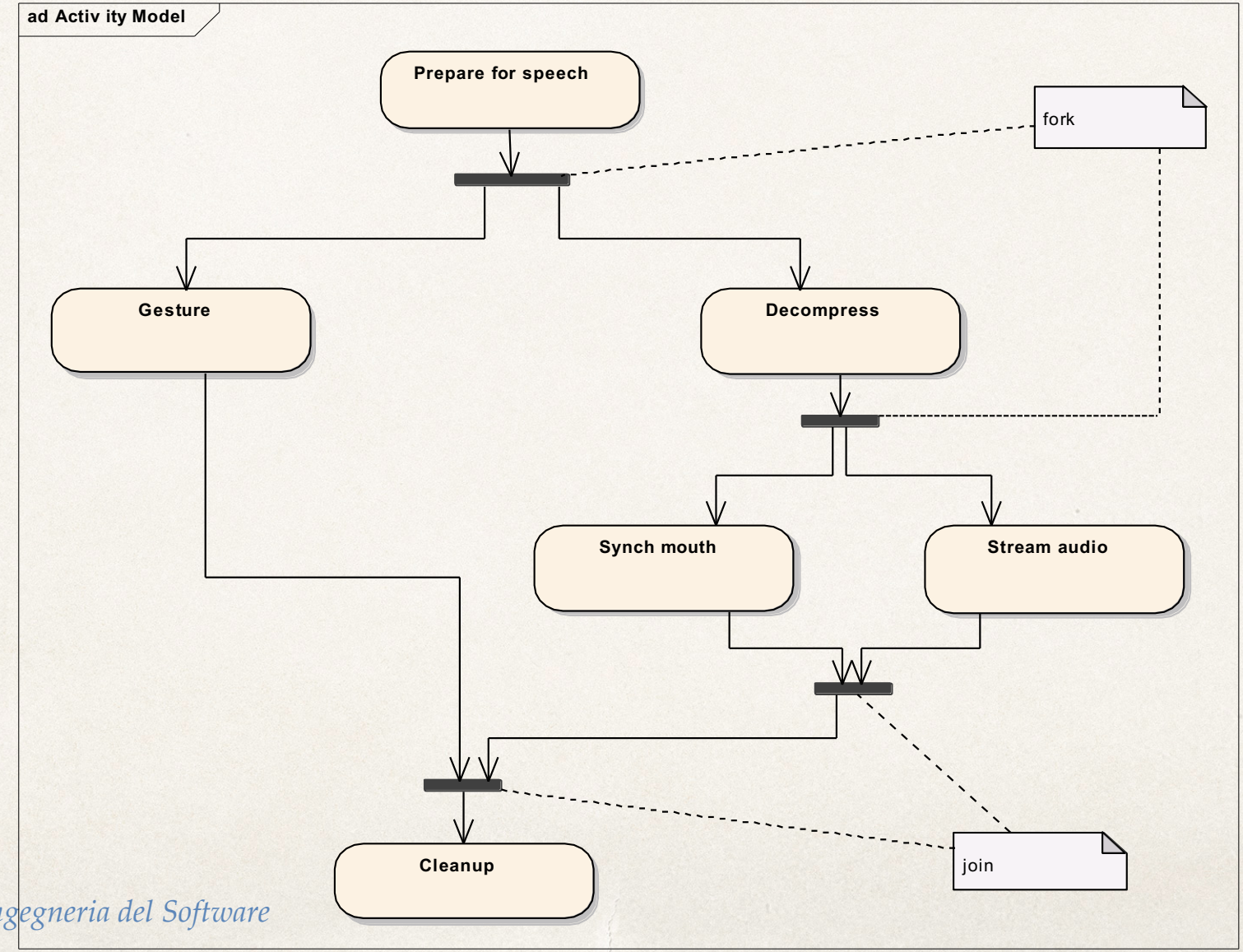


Diagramma di attività (cont)

- Alcuni flussi possono essere concorrenti
- In UML vengono usate delle barre di sincronizzazione per specificare fork e join di flussi di controllo paralleli
- Un join rappresenta la sincronizzazione di due o più flussi di controllo concorrenti
- Un join ha due o più flussi entranti e un flusso uscente
- La sincronizzazione sul join attende che tutte le attività nei flussi entranti abbiano terminato la loro esecuzione prima di procedere
- Join e fork si devono bilanciare
- Le attività in un flusso di controllo parallelo comunicano tra loro spedendosi segnali (stile di comunicazione detto co-routine)

Diagramma di attività (cont)



Diagrammi di attività (cont)

- A volte conviene separare le attività in base alle entità che le devono svolgere
- In UML si usano le cosiddette swimlane
- E' un raggruppamento (verticale o orizzontale) di attività eseguite da una stessa entità (per esempio una classe)
- Ogni swimlane deve avere un nome univoco nel diagramma
- Rappresentano responsabilità specifiche nel contesto di un'attività generale
- Le attività sono associate univocamente ad una unica swimlane
- Solo le transizioni (flussi) possono attraversare due o più swimlane

Diagrammi di attività (cont)

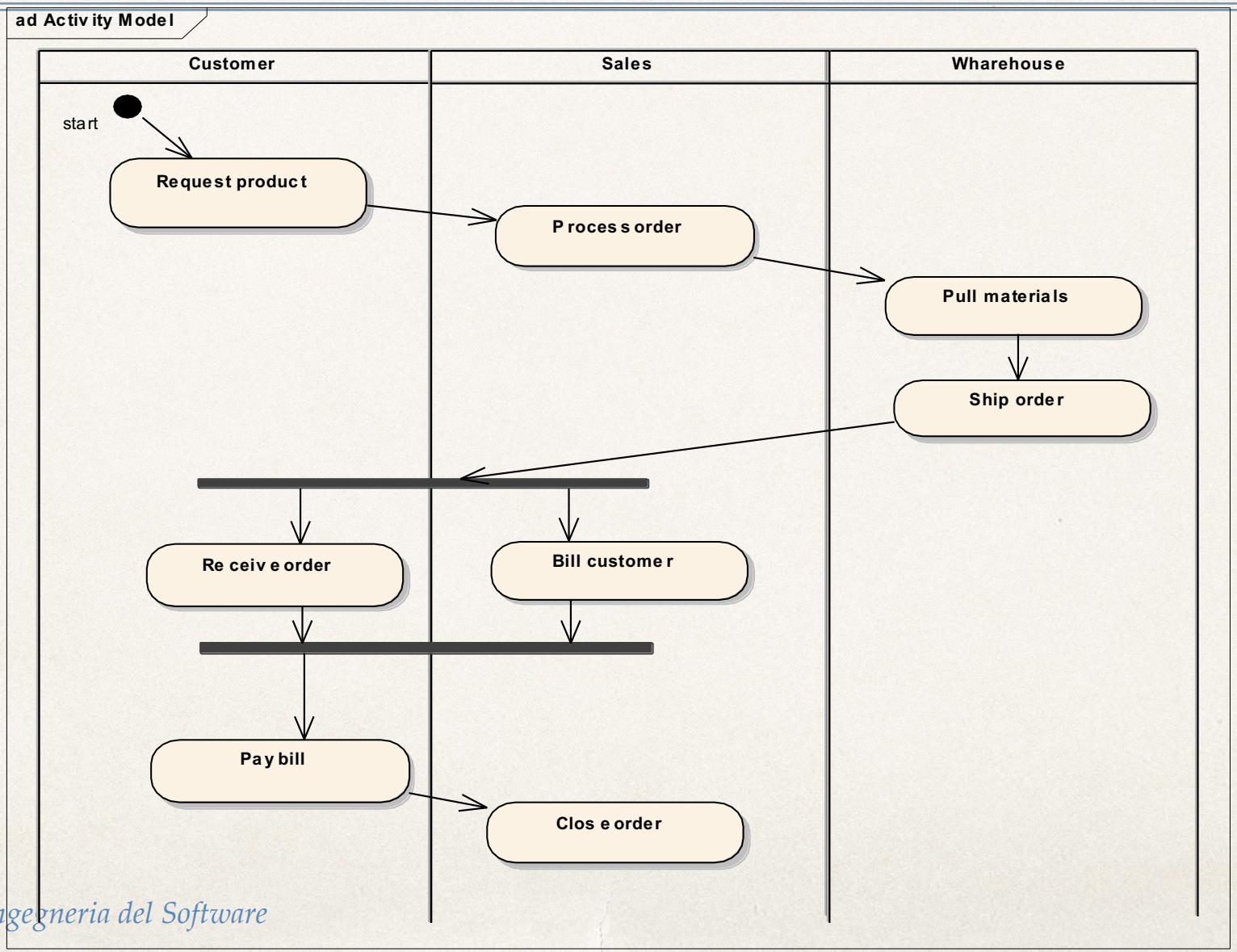


Diagramma di attività (cont)

- A volte è utile evidenziare non solo il flusso di controllo, ma anche gli oggetti coinvolti
- Un'attività può creare un oggetto
- Un'altra attività può contenere azioni che modificano lo stato interno di un oggetto
- Il flusso del valore (stato) di un oggetto tra due azioni è detto flusso dell'oggetto
- Lo stato viene rappresentato tra parentesi quadre all'interno dell'oggetto, oppure come constraint in una nota associata all'oggetto stesso

Diagramma di attività (cont)

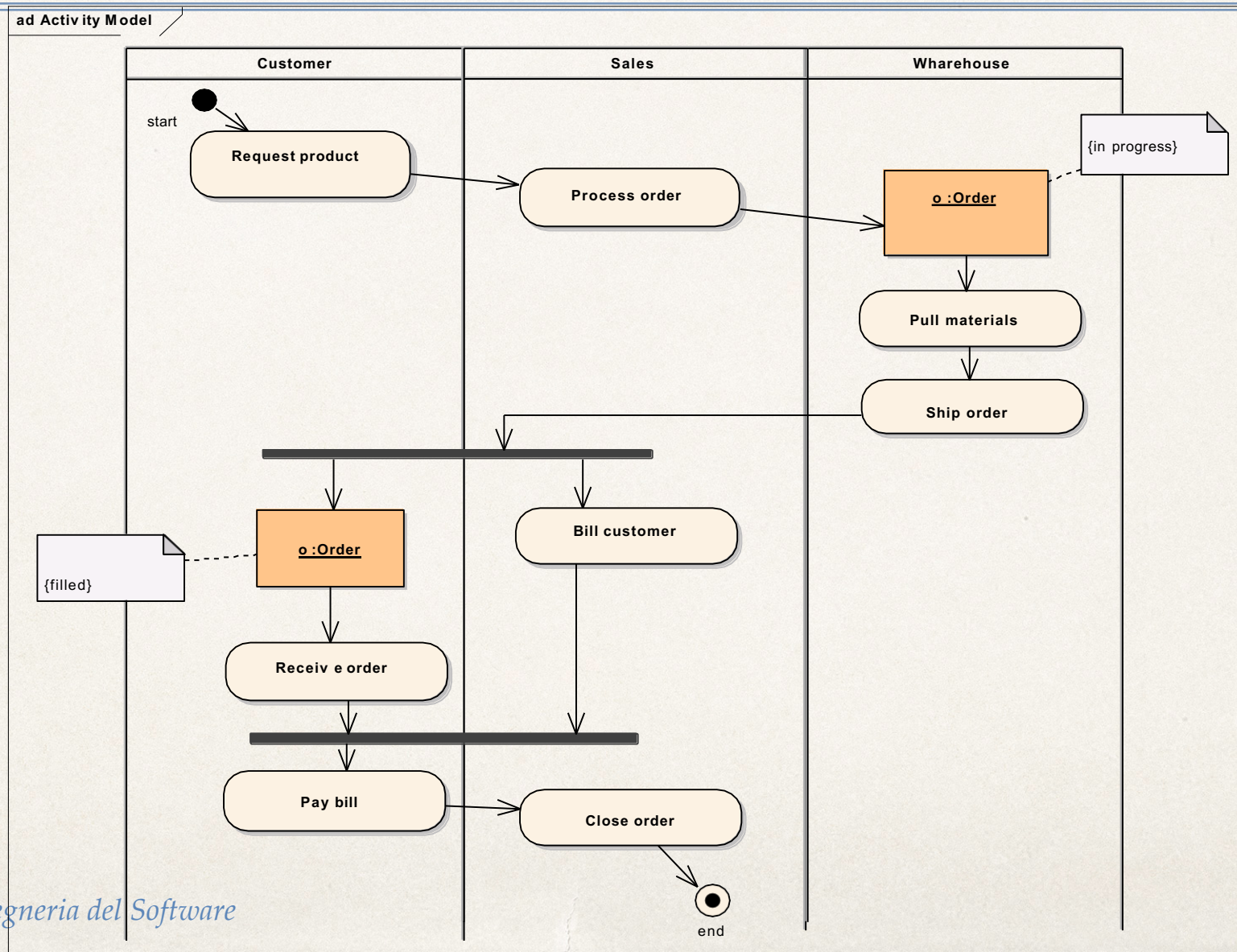


Diagramma di attività e casi d'uso

- Un caso d'uso può essere il punto di partenza per la costruzione di un diagramma di attività
- Entrambi sono rappresentazioni tipiche dell'analisi di un problema (o di un dominio)
- Il diagramma di attività fornisce una prospettiva algoritmica, mentre i casi d'uso forniscono una prospettiva funzionale
- Le due viste sono correlate, ma non totalmente equivalenti
- Il punto di partenza per costruire un diagramma di attività da un caso d'uso sono le descrizioni testuali, i flussi alternativi, le eccezioni, i singoli passi, le post-condizioni, le condizioni di terminazione
- Esempio: la spedizione di un ordine

Esempio: spedizione ordine

Descrizione testuale del caso d'uso Spedire Prodotti (Ship Product)	
Nome caso d'uso	Spedire Prodotti (Ship Product)
Scope	Delivery Shipment Subsystem
Goal (summary)	Il magazziniere predispone l'invio dei prodotti in un ordine a mezzo corriere
Attori	Magazziniere
Precondizioni	La spedizione viene effettuata utilizzando un corriere già registrato negli archivi del negozio
Descrizione (main success scenario o scenario principale)	1.Il magazziniere inserisce il codice del corriere da usare per la spedizione 2.il sistema visualizza il riepilogo relativo al corriere selezionato 3.Il magazziniere inserisce le informazioni della spedizione (data spediz., mittente, destinatario, elenco prodotti, quantità) 4.per ogni prodotto da spedire: (1)il magazziniere inserisce il numero d'ordine a cui si riferisce il prodotto (2)il sistema cerca corrispondenza tra ordine inserito e prodotto da spedire (3)il sistema controlla che il prodotto e la quantità corrispondano con dati spedizione

Esempio: spedizione ordine (cont)

	5.Il magazziniere notifica la bolletta di pagamento relativa all'ordine aggiornato del cliente 6.il sistema visualizza la notifica relativa alla bolletta di pagamento 7.in parallelo a 5 e 6, il magazziniere aggiorna il magazzino per registrare l'uscita dei prodotti dell'ordine che sta evadendo 8.il sistema visualizza lo stato del magazzino aggiornato 9.il magazziniere stampa la conferma di spedizione
Alternative (estensioni)	1a. Se il codice non corrisponde a un corriere viene visualizzato messaggio errore "Spedizioniere non trovato" 1b. Il sistema chiede di riprovare o di terminare la procedura 1c. il magazziniere decide di riprovare 1d. il caso d'uso torna al punto 1 del caso principale 4.1a Se l'ordine non viene trovato viene visualizzato errore "Numero d'ordine invalido" 4.1b il prodotto viene tolto dalla spedizione e viene messo da parte 4.2a se non viene trovata corrispondenza tra prodotto e ordine inserito, messaggio di errore "oggetto non valido" 4.2b il prodotto viene tolto dalla spedizione e viene messo da parte
Postcondizioni	Se la transazione viene completata con successo, il magazzino è aggiornato, viene emessa bolletta di pagamento relativa all'ordine evaso e viene stampata la conferma di spedizione. Se la transazione viene annullata, viene ripristinato lo stato iniziale
Condizioni di terminazione	Il caso d'uso termina se: •il sistema visualizza errore spedizioniere e il magazziniere annulla la transazione •viene emessa la bolletta e il magazzino e' aggiornato •il magazziniere annulla la transazione

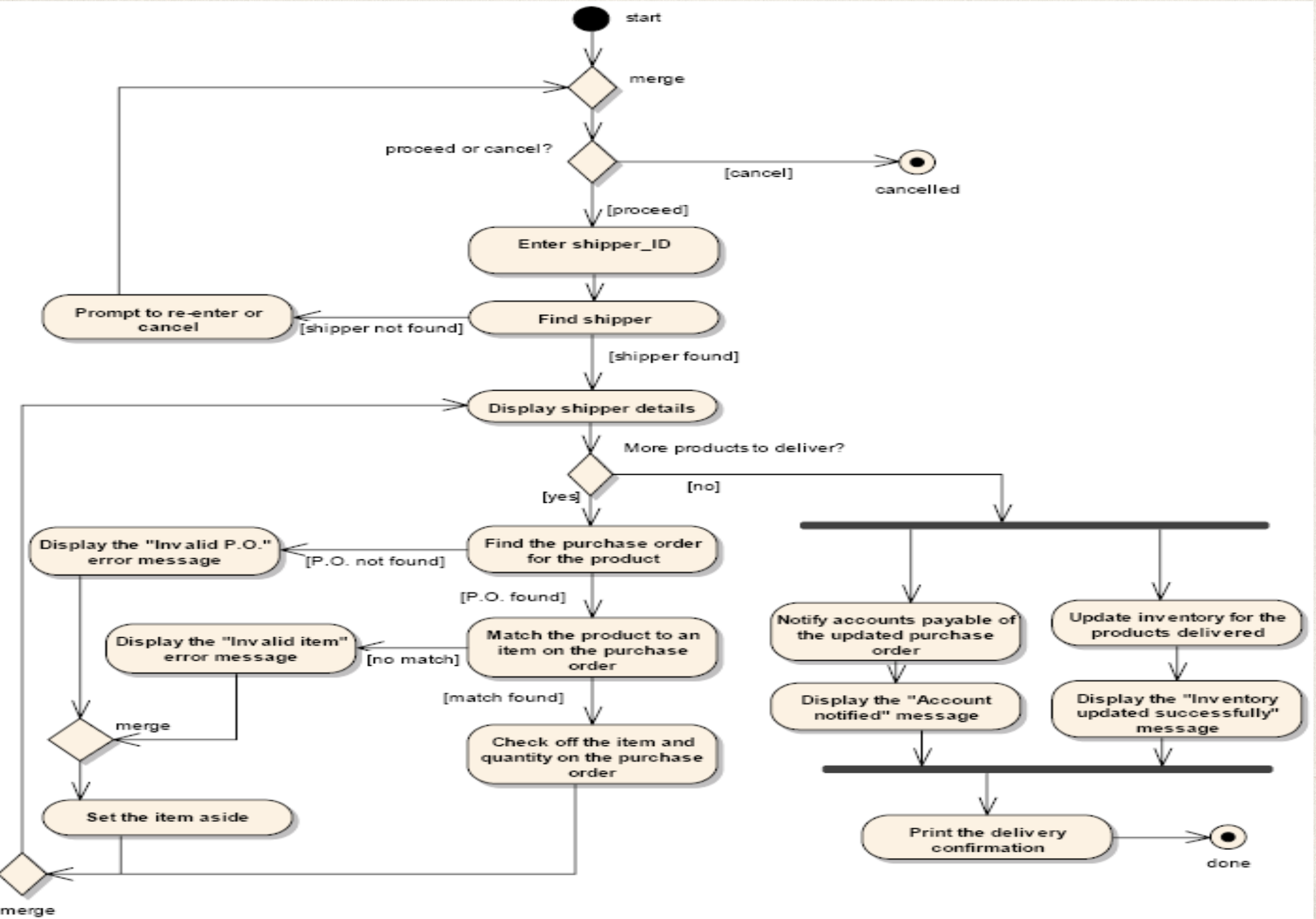

start
merge
proceed or cancel?
[cancel]
cancelled
[proceed]
Enter shipper_ID
Prompt to re-enter or cancel
[shipper not found]
Find shipper
[shipper found]
Display shipper details
More products to deliver?
[yes]
[no]
Display the "Invalid P.O." error message
[P.O. not found]
Find the purchase order for the product
[P.O. found]
Notify accounts payable of the updated purchase order
Update inventory for the products delivered
Display the "Invalid item" error message
[no match]
Match the product to an item on the purchase order
Display the "Account notified" message
Display the "Inventory updated successfully" message
[match found]
merge
Check off the item and quantity on the purchase order
Set the item aside
Print the delivery confirmation
done
merge

Diagramma dei componenti

- evidenzia l'organizzazione e le dipendenze tra i componenti software
- i componenti (come i casi d'uso o le classi) possono essere raggruppati in package
  - un componente è una qualunque porzione fisica riutilizzabile con un'identità e un'interfaccia (dichiarazione dei servizi offerti) ben definite
  - un componente può essere costituito dall'aggregazione di altri componenti

Diagramma dei componenti (cont)

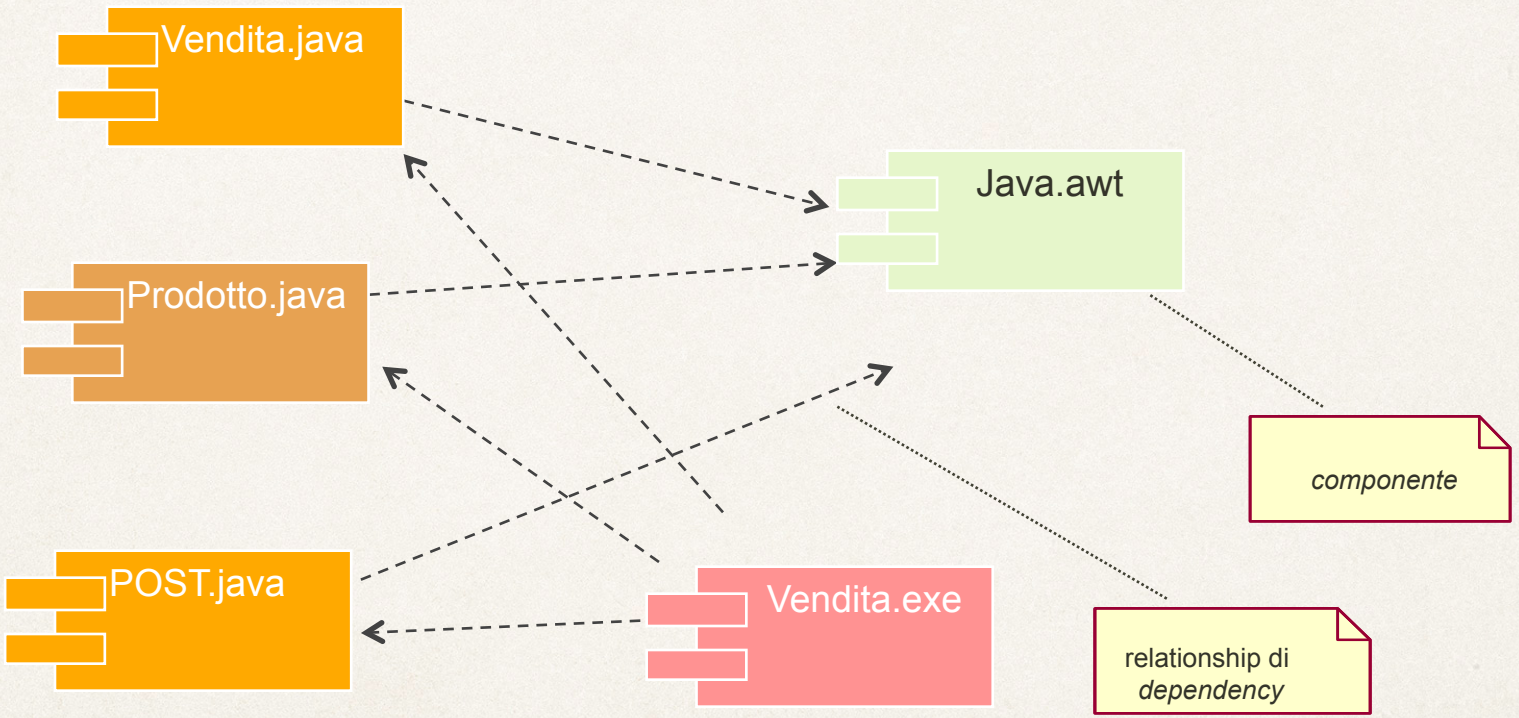
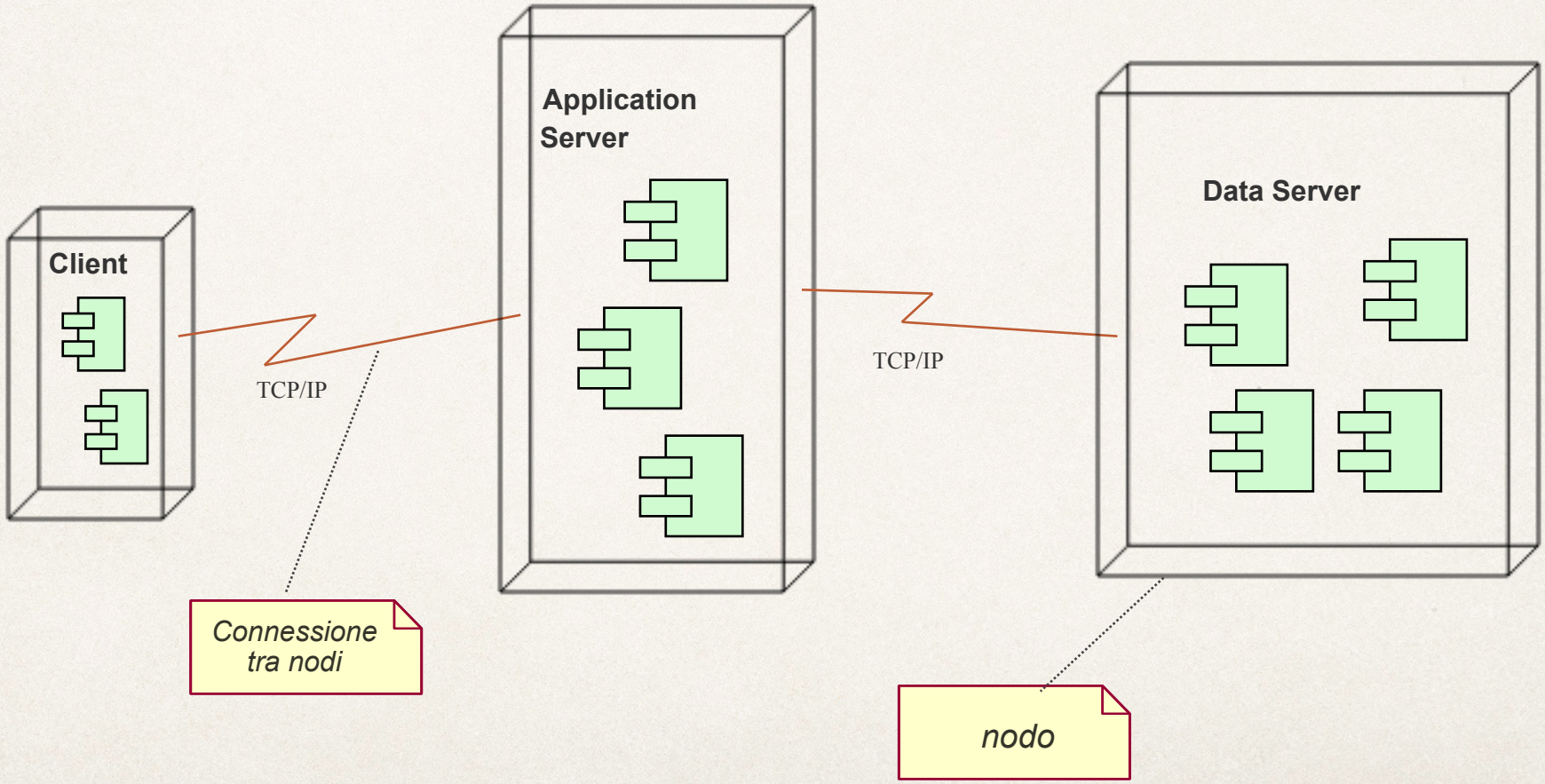


Diagramma di distribuzione

- Serve per mostrare come sono configurate e allocate le unità hardware e software per un'applicazione
- evidenzia la configurazione dei nodi elaborativi in ambiente di esecuzione (run-time) e dei componenti, processi ed oggetti allocati su questi nodi

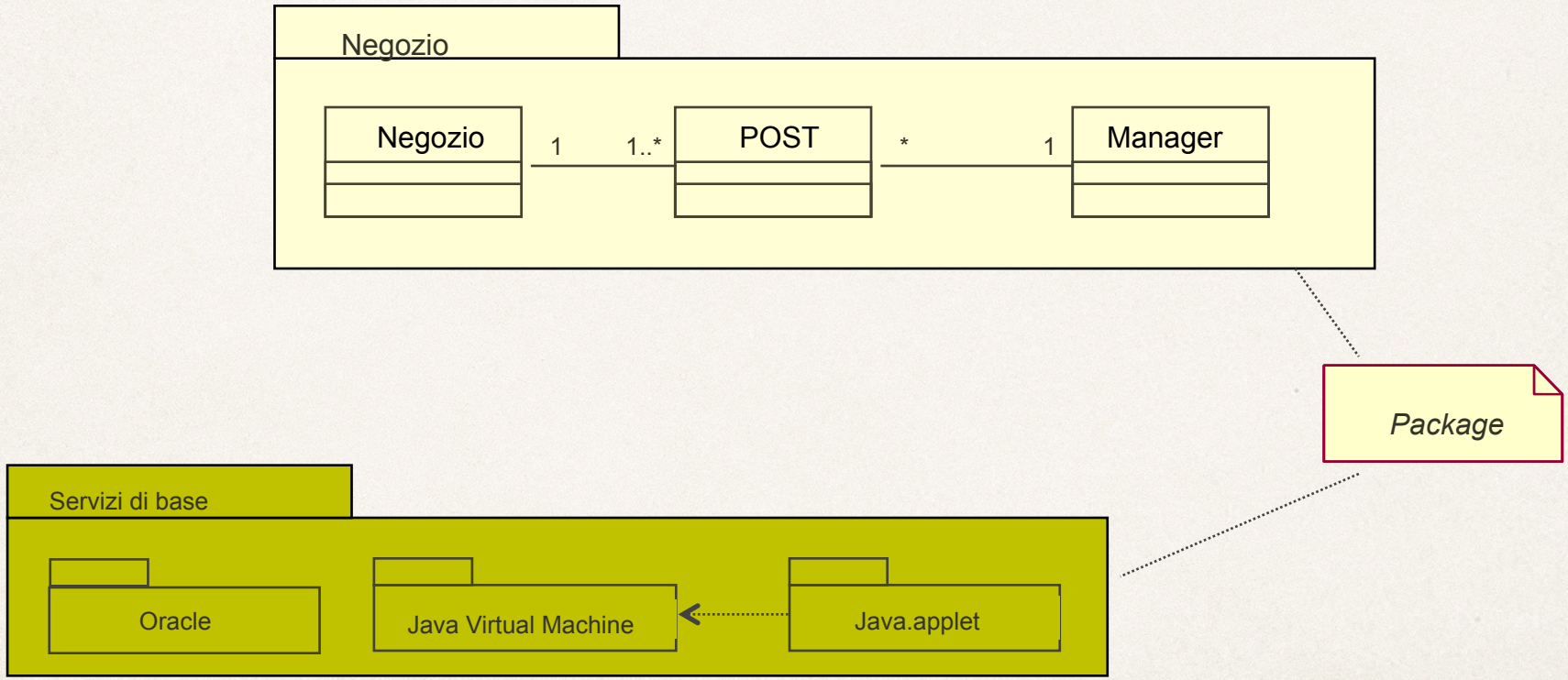
Diagramma di distribuzione (cont)



Package

- consente di partizionare il sistema in sottosistemi costituiti da elementi omogenei di
  - natura logica (classi, casi d'uso, …)
  - natura fisica (moduli, tabelle, …)
  - altra natura (processori, risorse di rete, …)
- ogni elemento appartiene ad un solo package
- un package può referenziare elementi appartenenti ad altri package

Package (cont)



Riassumendo

- UML è una evoluzione di modelli preesistenti
- è adatto a esprimere modelli di vario tipo, creati per obiettivi diversi
- può descrivere un sistema software a diversi livelli di astrazione, dal piano più svincolato dalle caratteristiche tecnologiche fino alla collocazione dei componenti software nei diversi processori di un'architettura distribuita
- può rispondere a tutte le necessità di modellazione, ma è opportuno "adattarlo" alle specifiche esigenze dei progettisti e dei progetti, utilizzando solo ciò che serve nello specifico contesto