

Progettazione del Software

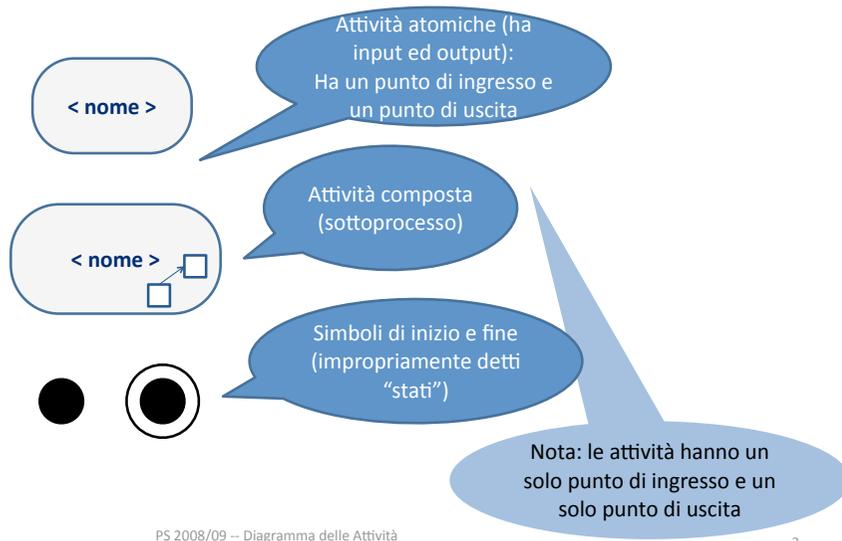
Diagramma delle Attività

Giuseppe De Giacomo, Massimo Mecella
Dipartimento di Informatica e Sistemistica
SAPIENZA Università di Roma

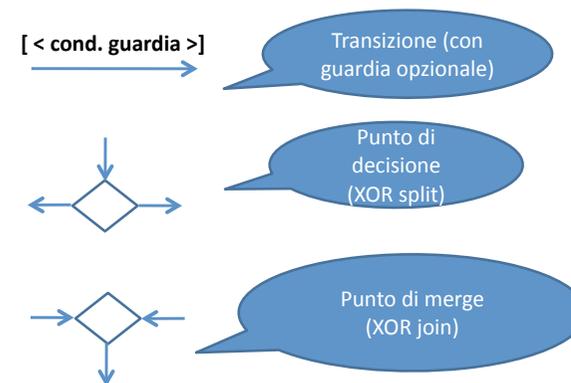
Diagramma delle Attività

- Utilizzato per
 - Modellare un comportamento dinamico di un sistema come un “flusso di attività”
 - Modellare un processo organizzativo (spesso indicato come “business process” o “workflow”)
- Nel corso, noi lo useremo per modellare i processi che coinvolgono gli oggetti identificati in precedenza
 - Comportamento “globale”
 - Sarà legato ai diagrammi degli stati dei singoli oggetti, che rappresentano il loro “comportamento locale”

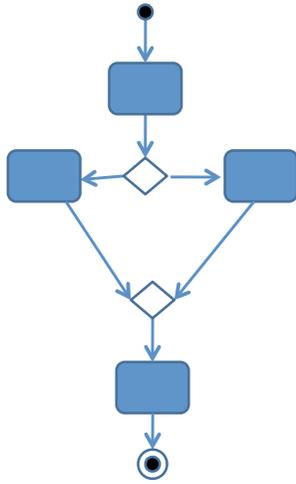
Elementi di Base: Atomici (1)



Elementi di Base: Sequenziali (2)



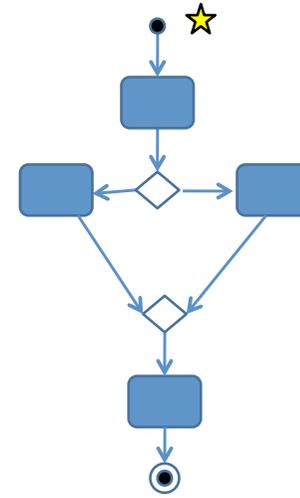
Esempio



PS 2008/09 -- Diagramma delle Attività

5

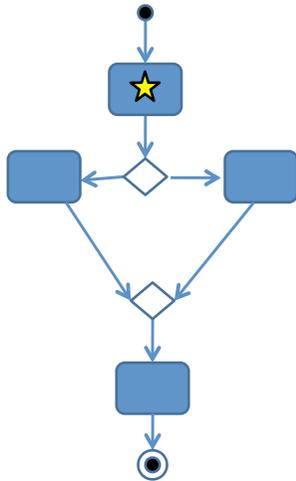
Esempio



PS 2008/09 -- Diagramma delle Attività

5

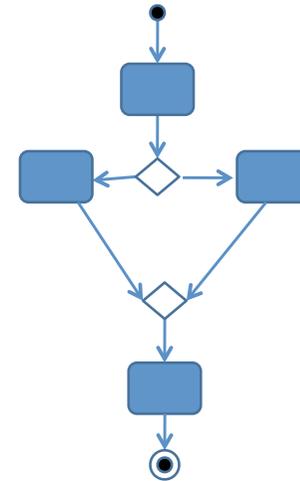
Esempio



PS 2008/09 -- Diagramma delle Attività

5

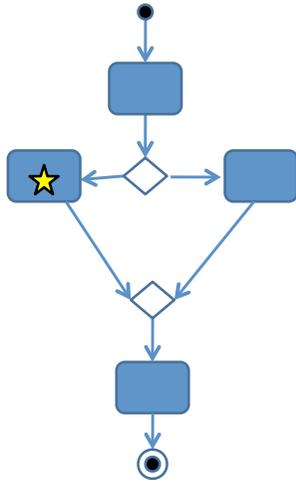
Esempio



PS 2008/09 -- Diagramma delle Attività

5

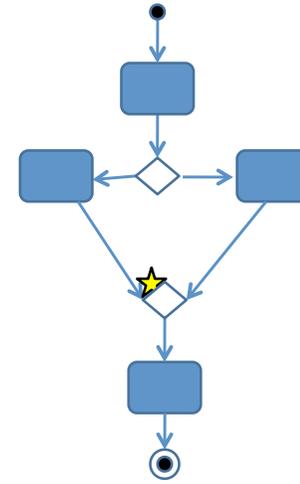
Esempio



PS 2008/09 -- Diagramma delle Attività

5

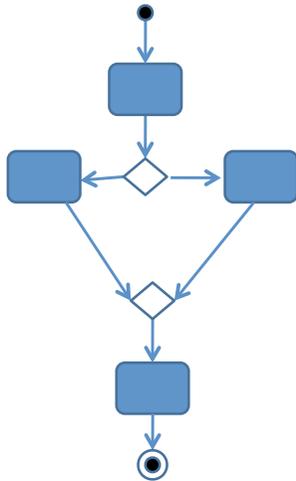
Esempio



PS 2008/09 -- Diagramma delle Attività

5

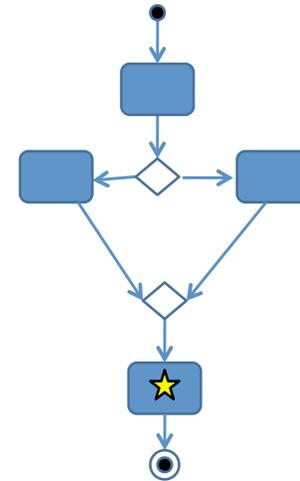
Esempio



PS 2008/09 -- Diagramma delle Attività

5

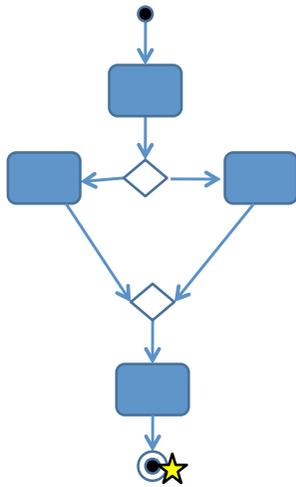
Esempio



PS 2008/09 -- Diagramma delle Attività

5

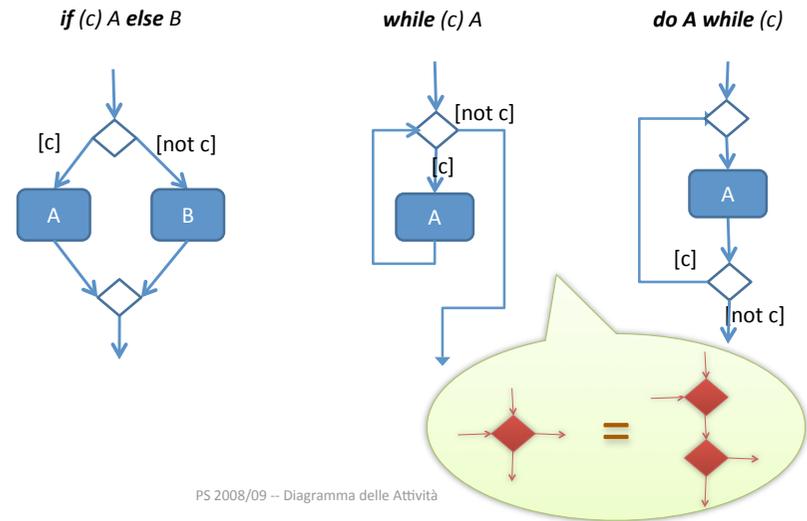
Esempio



PS 2008/09 -- Diagramma delle Attività

5

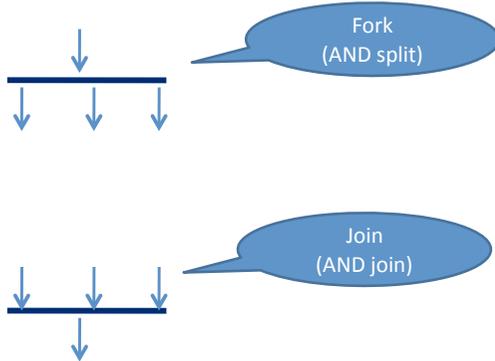
Altri esempi



PS 2008/09 -- Diagramma delle Attività

6

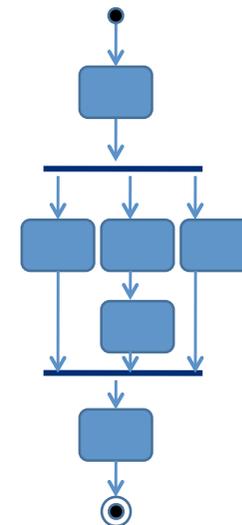
Elementi di Base: per Concorrenza (3)



PS 2008/09 -- Diagramma delle Attività

7

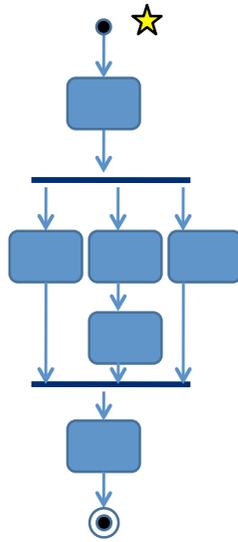
Esempio



PS 2008/09 -- Diagramma delle Attività

8

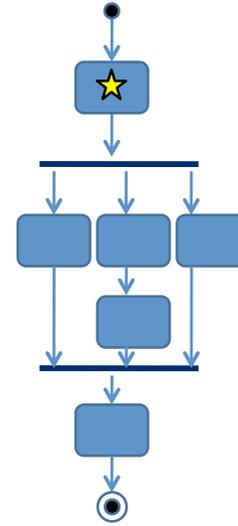
Esempio



PS 2008/09 -- Diagramma delle Attività

8

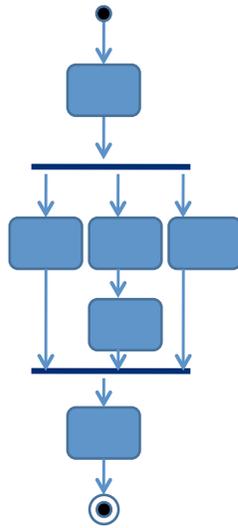
Esempio



PS 2008/09 -- Diagramma delle Attività

8

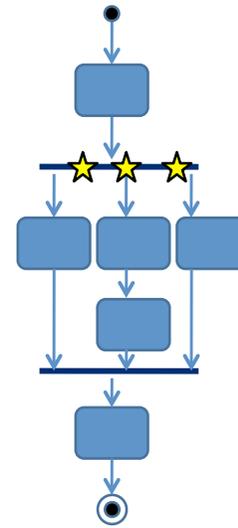
Esempio



PS 2008/09 -- Diagramma delle Attività

8

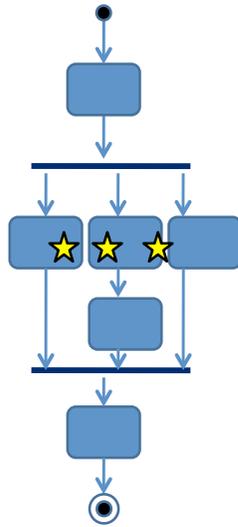
Esempio



PS 2008/09 -- Diagramma delle Attività

8

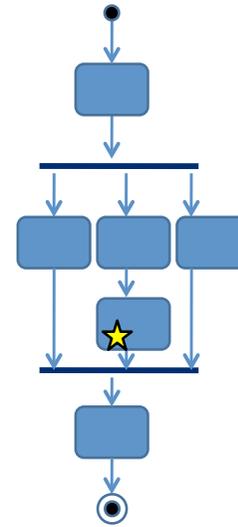
Esempio



PS 2008/09 -- Diagramma delle Attività

8

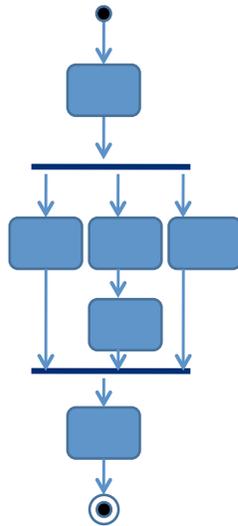
Esempio



PS 2008/09 -- Diagramma delle Attività

8

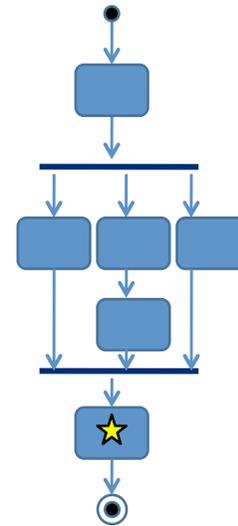
Esempio



PS 2008/09 -- Diagramma delle Attività

8

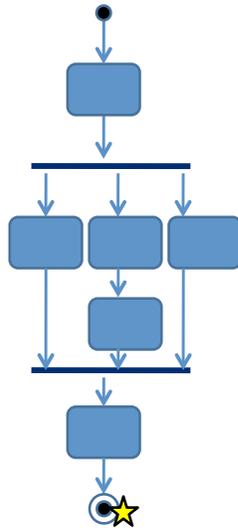
Esempio



PS 2008/09 -- Diagramma delle Attività

8

Esempio



PS 2008/09 -- Diagramma delle Attività

8

Nota sul significato del Fork/Join

- Flusso (*thread*) di esecuzione
 - “Esecutore” che è in carico di gestire l’avanzamento del processo
 - Nel fork, il numero di tali flussi/thread/esecutori viene aumentato, in quanto prima era unico e dopo ce ne è uno per ogni transizione in uscita
 - Nel join il numero viene diminuito, in quanto prima ce ne erano tanti (uno per ogni transizione in entrata) e dopo ce ne è di nuovo uno solo
 - Questo può partire solo dopo che sono terminati tutti quelli che devono fare join
 - Sincronizzazione dei esecutori

PS 2008/09 -- Diagramma delle Attività

9

Significato del Fork/Join (2)

- Attenzione che nel Punto di decisione/Merge non è così, in quanto in questo caso solo una delle transizioni in uscita viene eseguita dall'esecutore, non tutte contemporaneamente (come nel fork)

PS 2008/09 -- Diagramma delle Attività

10

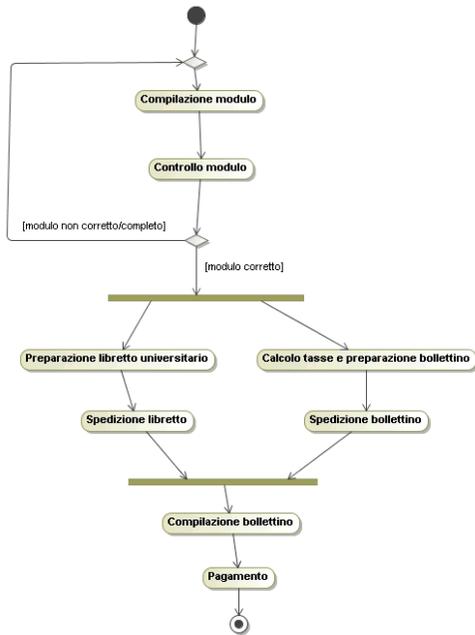
Esempio

- Per iscriversi all'Università viene compilato un modulo, che viene ispezionato al fine di verificarne la correttezza, ed eventualmente fatto ricompilare fino a quando non è corretto e completo. Successivamente, viene preparato il libretto universitario e spedito a casa dello studente, e contemporaneamente vengono calcolate le tasse di iscrizione e spedito l'apposito bollettino. Alla ricezione di entrambi, si può procedere alla compilazione del bollettino (infatti bisogna inserire il numero del libretto) ed al pagamento del bollettino

PS 2008/09 -- Diagramma delle Attività

11

Esempio



PS 2

12

Swimlane

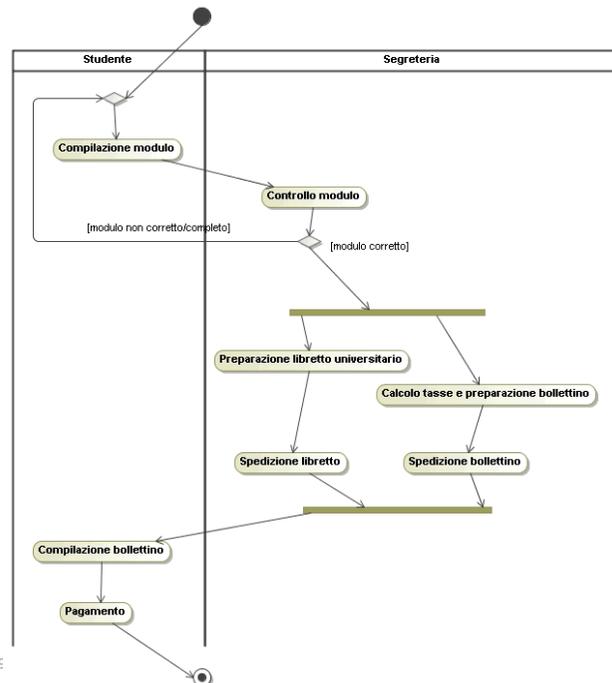
- Problema: il diagramma non evidenzia chi deve fare cosa (ovvero gli attori che eseguono le attività).
- Se si vuole mettere in evidenza chi fa cosa si introducono le swimlane.
- Una swimlane (“corsia di nuoto”) permette di evidenziare quali azioni sono eseguite da un dato attore
 - Graficamente è una linea verticale che identifica la “corsia”

PS 2008/09 -- Diagramma delle Attività

13

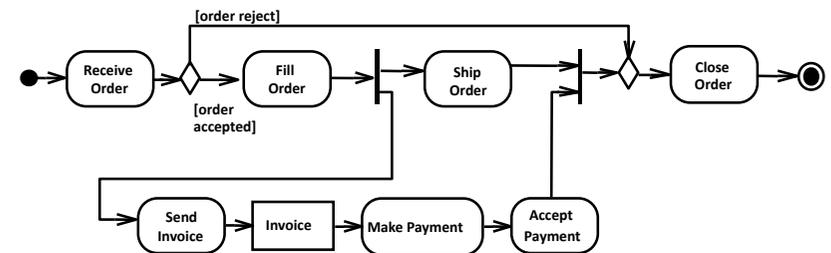
Esempio tratto dalle slide “Activity diagrams in UML 2.0” di Emanuele DeBenedetti -Univ. Genova

Esempio con Swimlane



PS 2008/09

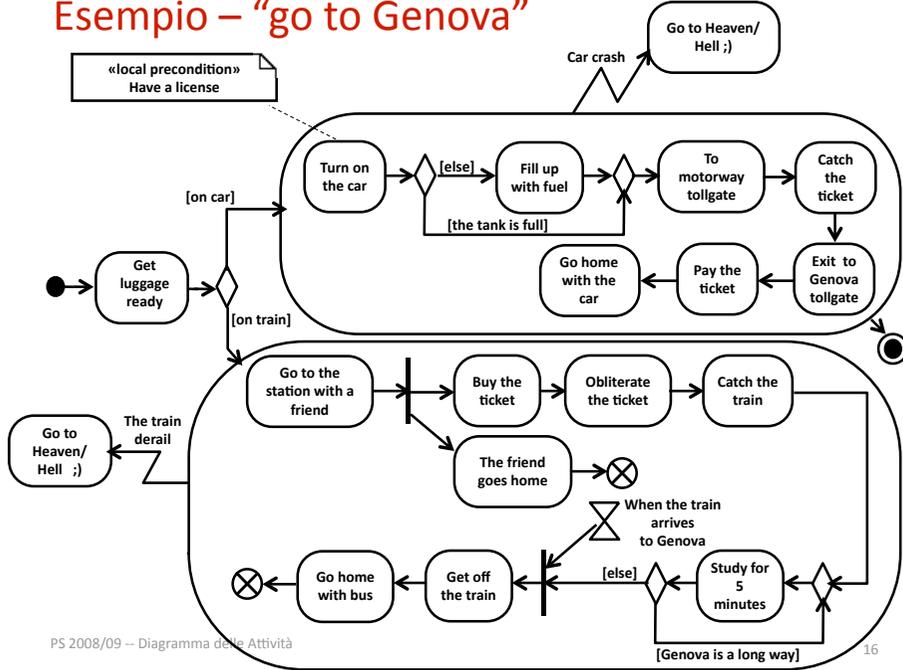
Esempio – “orders”



PS 2008/09 -- Diagramma delle Attività

15

Esempio – "go to Genova"



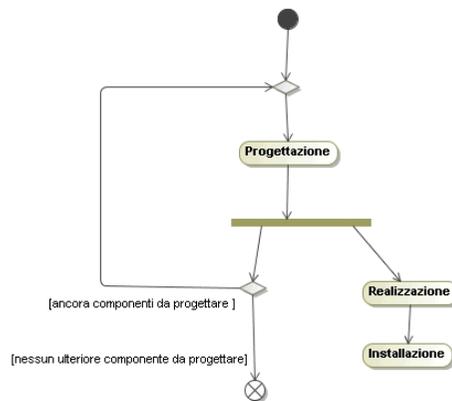
FlowFinal

- A volte serve segnalare che un ramo concorrente della computazione va a chiudersi.
- Per fare ciò si usa il simbolo di terminazione di un ramo concorrente: FlowFinal



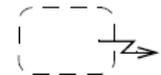
Esempio

- Pattern "multiple instances without synchronization": necessità di creare più istanze di una determinata attività, ognuna indipendente
 - ... fino a che ci sono componenti da sviluppare, progettare, realizzare ed installare un componente ...



Altri costrutti

- Regione interrompibile
 - Raggruppamento di attività e transizioni che supporta la terminazione del flusso di esecuzione
 - Se la terminazione avviene, tutto il flusso ed il comportamento dovuto alle attività interne alla regione è terminato
- Evento Temporale
 - genera un evento per sincronizzarsi con una condizione temporale



at() indica la condizione che triggera l'evento (e la relativa transizione):



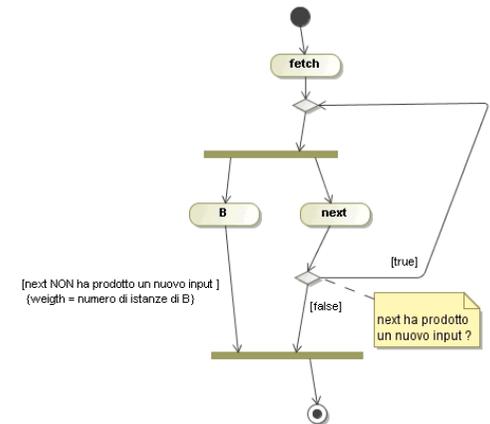
Fork Iterato

- A volte è necessario iterare un fork per un numero di volte definibile solo a runtime
- Poi tutti i fork devono essere sincronizzati, e per fare ciò si mettono dei pesi sulle transizioni, intendendo che la transizione, quando avviene, consuma quel determinato numero di flussi di esecuzione
 - minimo numero di attivazioni che devono avvenire contemporaneamente

{weight = < valore >
→

Esempio (1)

- Pattern “iterated activity”: si vogliono avviare più flussi della stessa attività (ognuna con input differenti), il cui numero è noto solo a run-time dopo l’esecuzione di un’attività ulteriore; alla fine si deve attendere il completamento di tutte le attività avviate



Esempio (2)

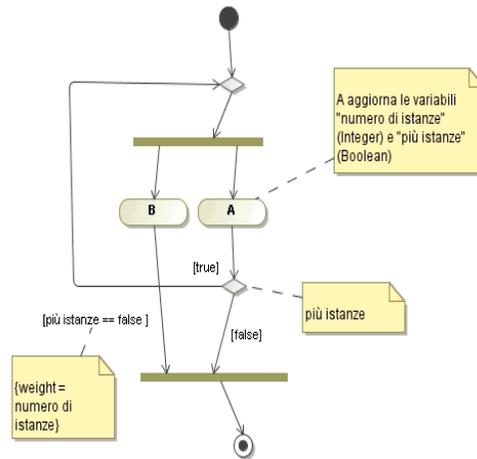
- Serve una variabile che controlla il processo, numero di istanze, che indica quante istanze di attività B sono state attivate
- L’attività `fetch` prepara l’insieme di dati di input su cui le varie istanze di B lavorano; la `next` estrae da tale insieme il dato di input che verrà passato all’istanza di B che si sta per forkare
- L’attività B rappresenta quello che si vuole attivare contemporaneamente

Esempio (3)

- La transizione in uscita da B è controllata da:
 - Una guardia, che è vera solo quando non ci sono più iterazioni
 - Un peso, il cui valore, nel momento in cui la transizione scatta, è uguale a `numero istanze`, è quindi esattamente pari al numero di B che sono state “forkate”
 - La sincronizzazione quindi avviene tra le `numero istanze thread` di B ed il thread “iteratore”
- Notare come all’iterazione `i`, esistono `i+1` flussi di controllo contemporaneamente attivi: 1 d’iterazione e `i` per B

Esempio avanzato (1)

- Pattern “multiple instances with synchronization”: come il precedente, ma ora l’iterazione è controllata da un’altra attività



Esempio avanzato (2)

- Servono due variabili che controllano il processo
 - numero di istanze, che indica quante istanze di attività B sono state attivate
 - più istanze (booleano), che indica se è necessario, ad ogni iterazione, attivare una nuova istanza
- L’attività A, che potremmo definire “controllore” dell’iterazione, modifica ad ogni esecuzione le variabili (in base alla propria logica interna)
- L’attività B rappresenta quello che si vuole attivare contemporaneamente in un numero di istanze controllato a tempo di esecuzione

Esempio avanzato (3)

- La transizione in uscita da B è controllata da:
 - Una guardia, che è vera solo quando non ci sono più iterazioni (notare che la guardia è `più istanze == false`, e che la condizione di uscita dall’iterazione – nodo di decisione, è anch’essa `più istanze == false`)
 - Un peso, il cui valore, nel momento in cui la transizione scatta, è uguale a `numero istanze`, è quindi esattamente pari al numero di B che sono state “forkate”
 - La sincronizzazione quindi avviene tra le `numero istanze` thread di B ed il thread “decisore” (quello di A)
- Notare come all’iterazione i , esistono $i+1$ flussi di controllo contemporaneamente attivi: 1 per A e i per B