5. Modellazione con UML (III)

Paola Barra
a.a. 2022/2023

UML dove eravamo rimasti

- UML fornisce un’ampia gamma di notazioni per modellare una varietà di aspetti dei sistemi software
- Un modello di Sistema è costituito
  - Modello funzionale: diagrammi dei casi d’uso
  - Modello ad oggetti: diagrammi delle classi
  - Modello dinamico: diagrammi delle sequenze, di stato

Diagrammi di interazione

- Descrivono le modalità di comunicazione tra un insieme di oggetti interagenti
- Un oggetto interagisce con un altro oggetto inviando messaggi
  - La ricezione di un messaggio da un oggetto aziona l'esecuzione di un metodo che a sua volta può inviare messaggi ad altri oggetti
  - Possono essere inviati degli argomenti insieme al messaggio compatibilmente con i parametri del metodo di cui si richiede l'esecuzione

Diagramma delle sequenze

Diagrammi di sequenza

- Si usano:
  - per descrivere le interazioni: scambio di messaggi e dati tra oggetti
    - per esempio un attore e il sistema per la realizzazione di un caso d'uso
    - oppure, in fase di progettazione, i messaggi scambiati tra sottosistemi
  - organizzati in sequenza temporale

Elementi di un diagramma di sequenza

- Oggetti partecipanti alle interazioni sono rappresentati con linee di vita formate da:
  - un rettangolo, che indica ruolo (nell'interazione) e/o tipo dell'oggetto (uno dei due obbligatorio, entrambi solo se utile)
  - una linea verticale chiamata linea di vita dell'oggetto
    - questa linea è tratteggiata quando l'oggetto è inattivo,
    - continua e doppia quando l'oggetto è attivo. Oggetti sempre attivi (es attori) hanno l'intera linea di vita continua e doppia.

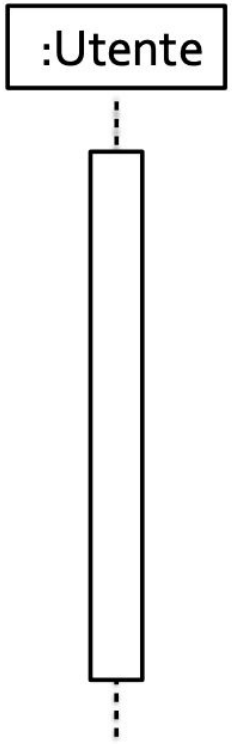
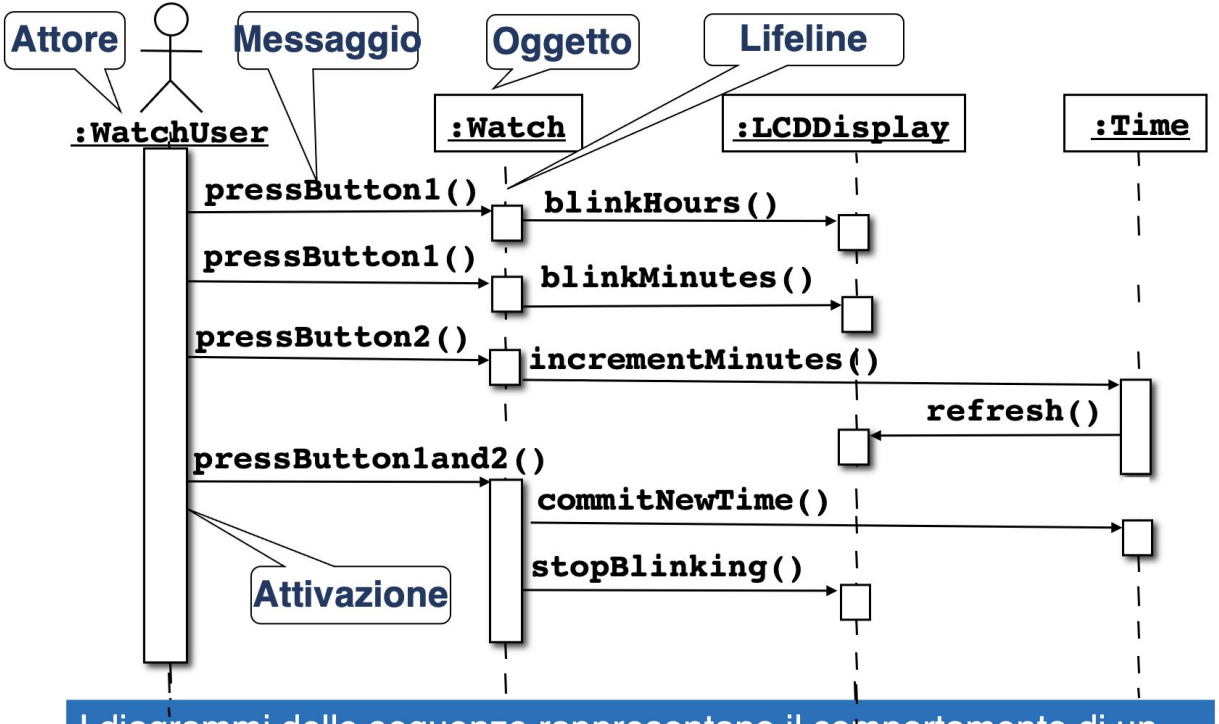


Diagramma delle sequenze



I diagrammi delle sequenze rappresentano il comportamento di un Sistema come messaggi (*interazioni*) tra oggetti differenti

Diagramma delle sequenze

- Usati
  - Durante l'analisi dei requisiti per rifinire le descrizioni dei casi d'uso e per trovare oggetti aggiuntivi (oggetti partecipanti)
  - Durante la progettazione del sistema per rifinire le interfacce dei sottosistemi

- I diagrammi delle sequenze rappresentano orizzontalmente gli oggetti partecipanti nell'interazione e verticalmente il tempo
- Esempio
  - Un orologio con due pulsanti (2Bwatch)

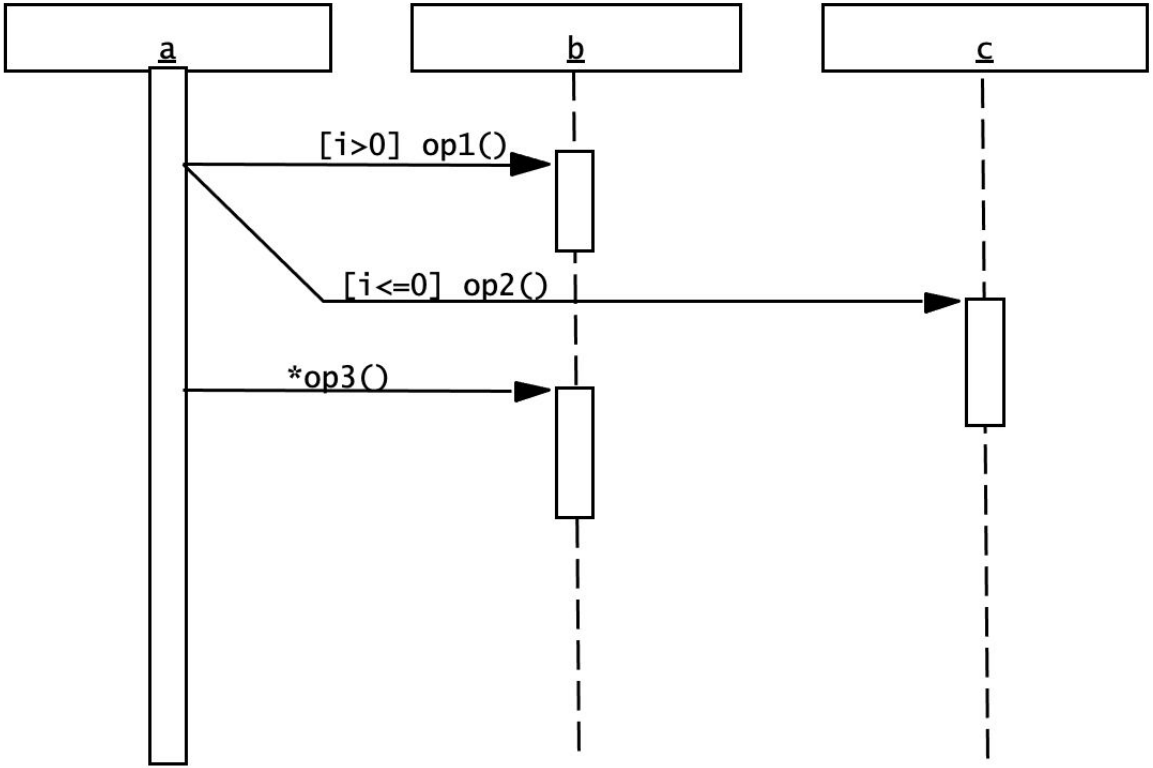
Notazioni

- Le colonne rappresentano gli oggetti che partecipano nell'interazione
- Le frecce rappresentano i messaggi
  - Le etichette rappresentano i nomi dei metodi che possono contenere argomenti
- I rettangoli verticali rappresentano le attivazioni (esecuzione dei metodi)
- L'attore che inizia l'interazione è rappresentato nella prima colonna a sinistra
- I messaggi provenienti dall'attore rappresentano le interazioni descritte nei diagrammi dei casi d'uso
  - Se altri attori comunicano con il sistema durante il caso d'uso, questi attori sono rappresentati sul lato destro e possono ricevere messaggi

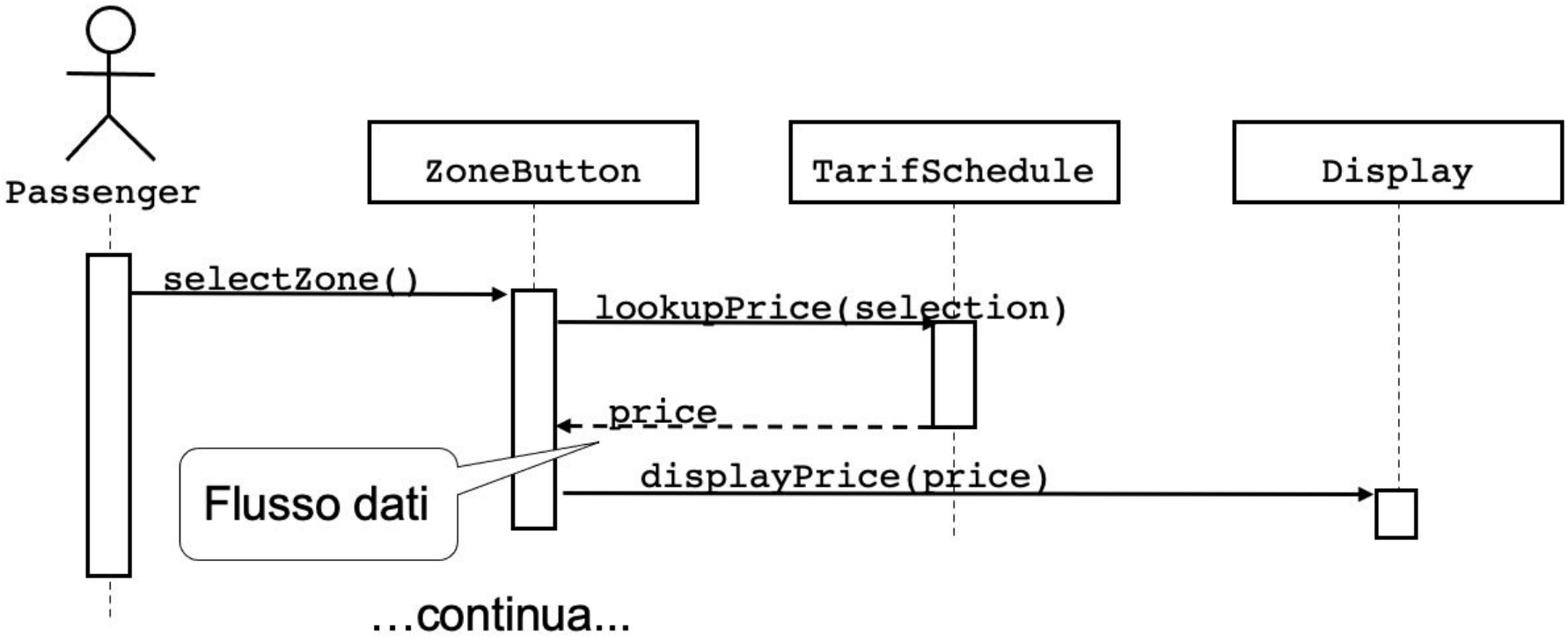
Condizioni ed iterazioni

- I diagrammi delle sequenze possono essere usati per descrivere una sequenza astratta (tutte le possibili interazioni) o sequenze concrete (una possibile interazione)
- Sono disponibili notazioni per esprimere condizioni o iterazioni (quando si descrivono tutte le possibili interazioni)
  - Una **condizione** su un messaggio è rappresentata da una espressione tra parentesi quadre prima del nome del messaggio. Se la condizione è vera il messaggio è inviato
  - Una invocazione **ripetitiva** di un messaggio è denotata da un '*' prima del nome del messaggio

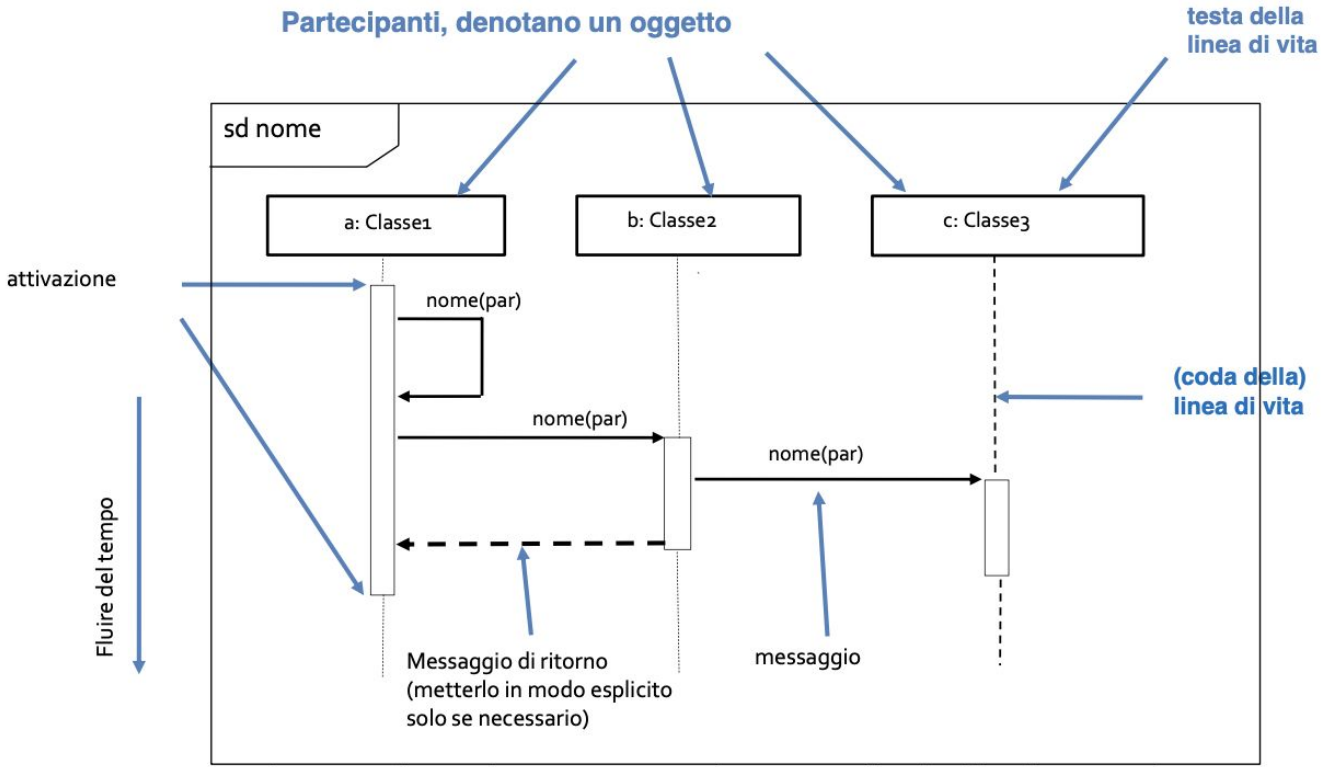
Esempi di condizioni ed iterazioni



Diagrammi delle sequenze: flusso di dati



Diagrammi di sequenza



Messaggi scambiati, l'ordine cronologico è dall'alto in basso

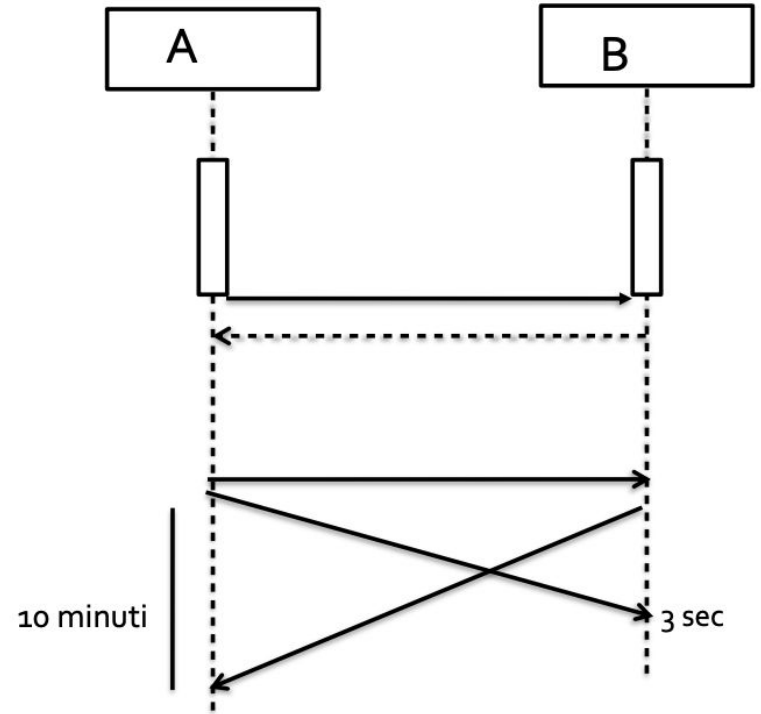
I messaggi: rappresentano invocazione di operazione o segnali

- Possono essere
  - sincroni

(es comunicazione diretta)

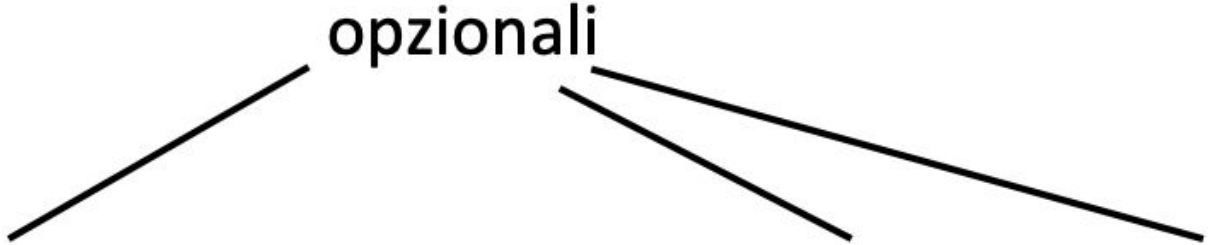
  - di return (opzionali)
  - asincroni

(es invio email)



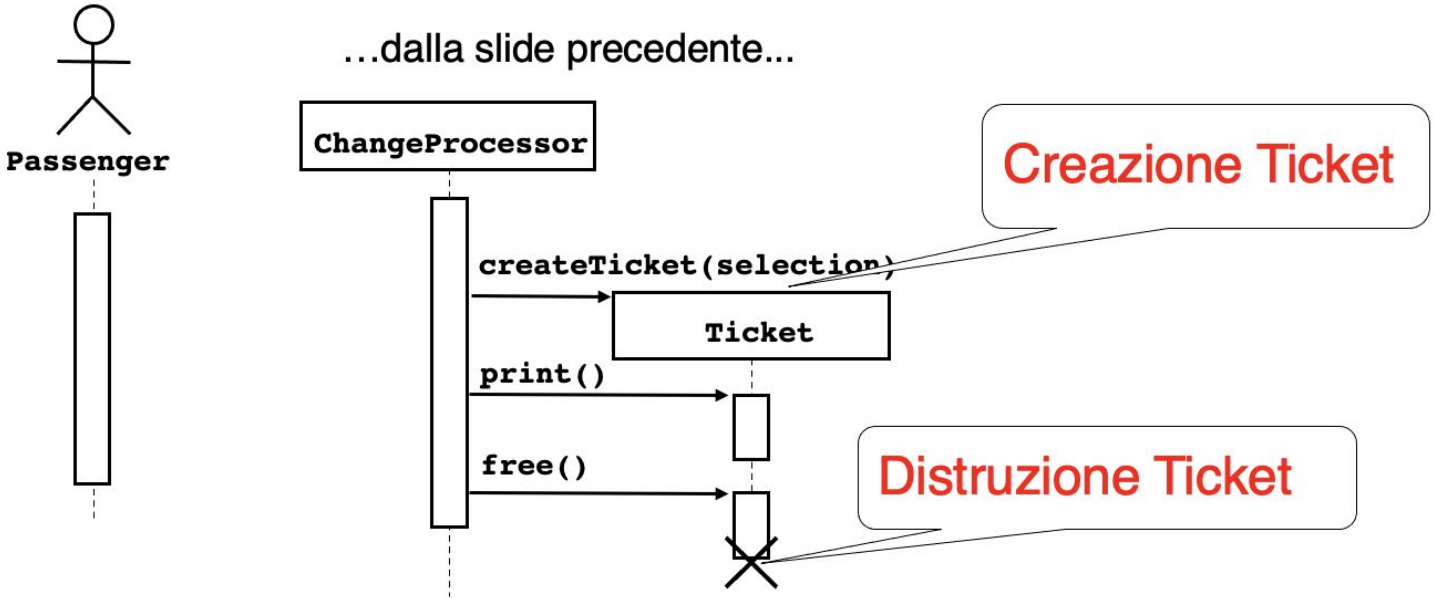
  - eventualmente con esplicito consumo di tempo

Sintassi dei messaggi



attributo = nomeMessaggio(arg1, arg2, …) : valore di ritorno

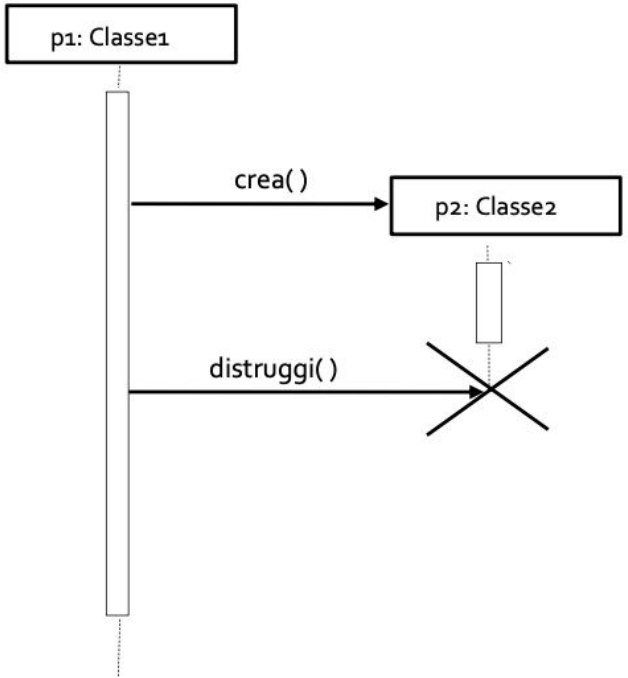
Creazione e distruzione



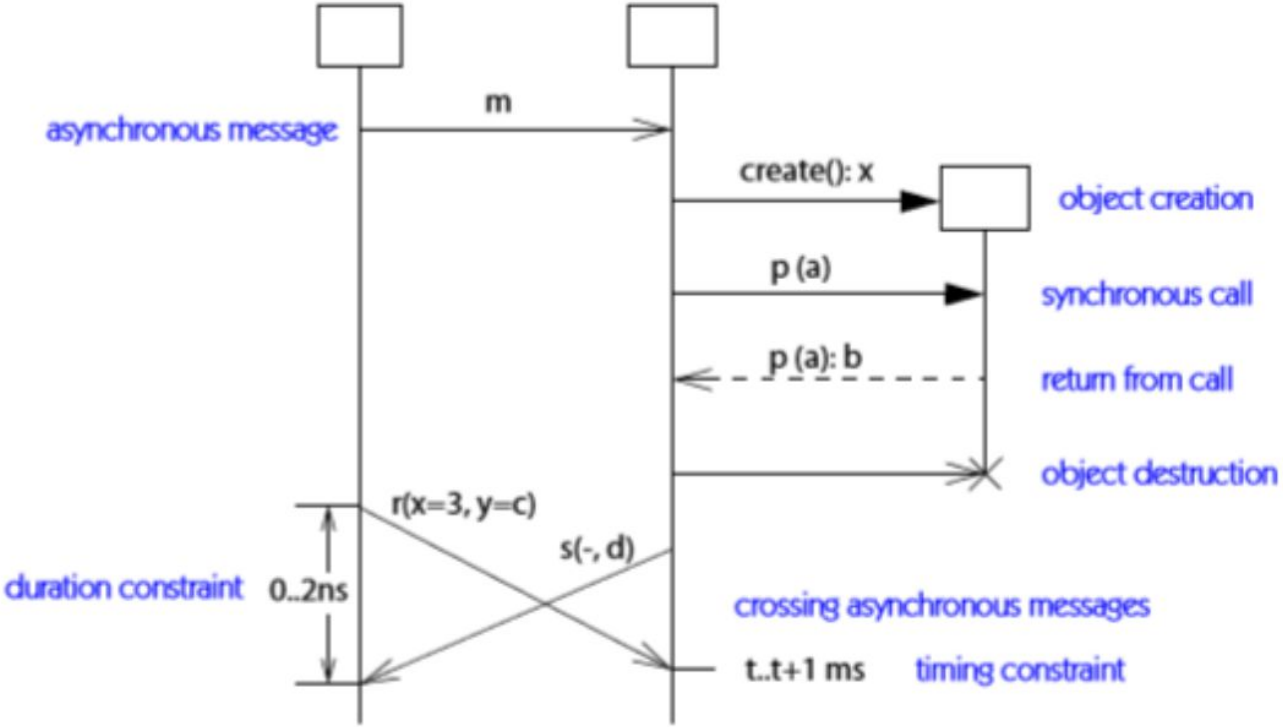
- La creazione è denotata da un messaggio freccia che punta all'oggetto
- La distruzione è denotata da una x alla fine dell'attivazione

Creare e distruggere partecipanti

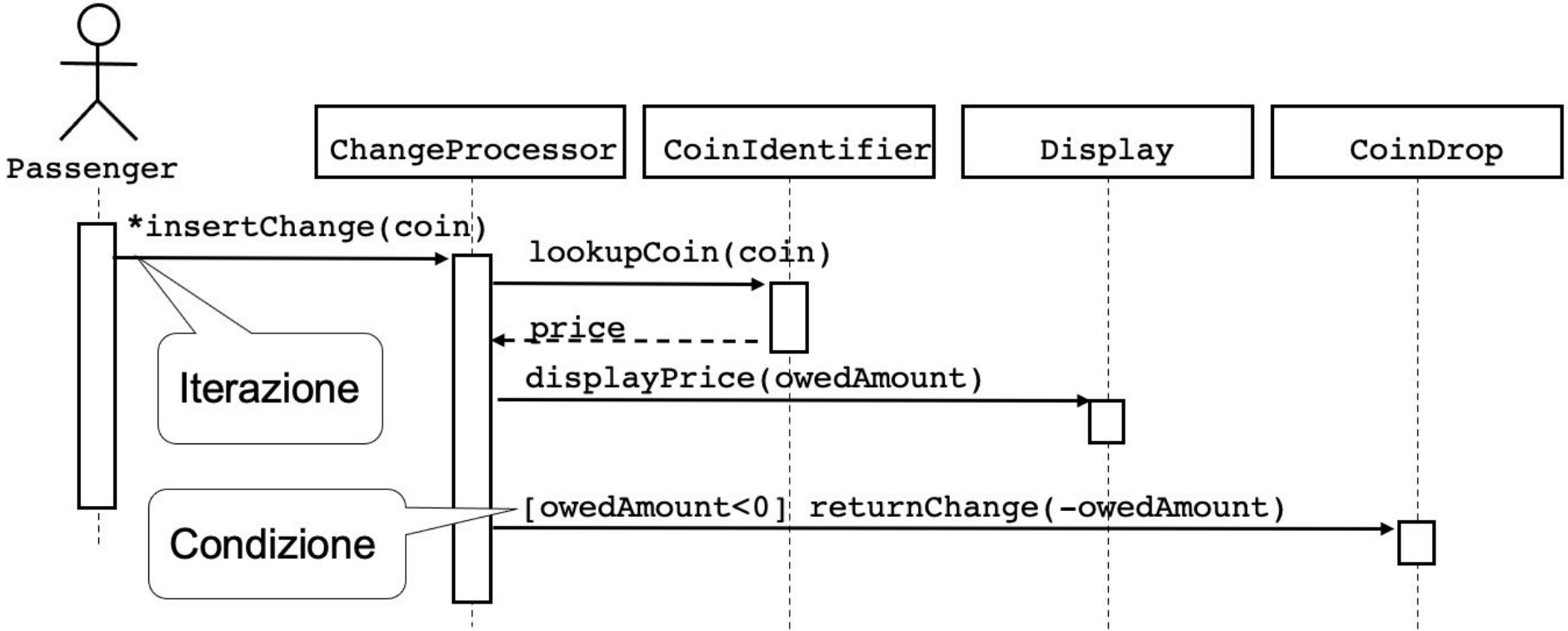
- Alcuni partecipanti possono essere
  - aggiunti dinamicamente all'interazione
  - cancellati



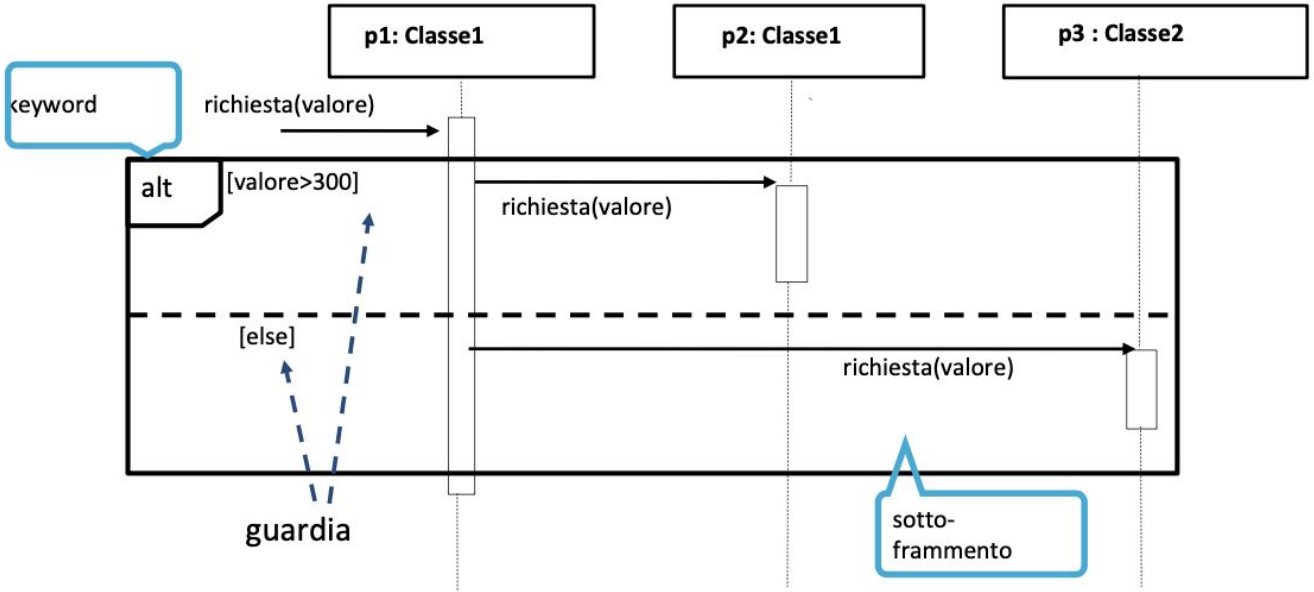
Esempio


m
asynchronous message
create(): x
object creation
p (a)
synchronous call
p (a): b
return from call
object destruction
r(x=3, y=c)
s(-, d)
duration constraint
0..2ns
crossing asynchronous messages
t..t+1 ms
timing constraint

Iterazioni e condizioni

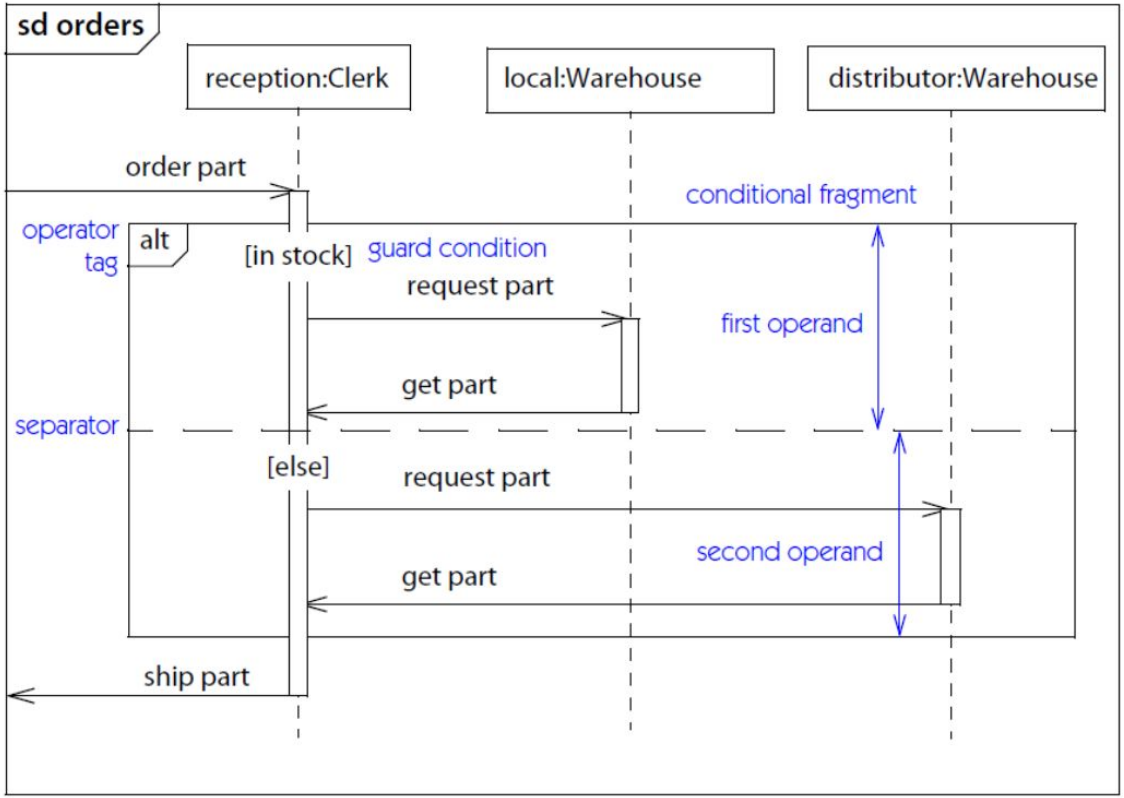


Frame condizionale



- senza guardia = [true]
- più guardie vere: scelta non-deterministica
- tutte le guardie false: il frame viene saltato

Frame condizionale : un altro esempio

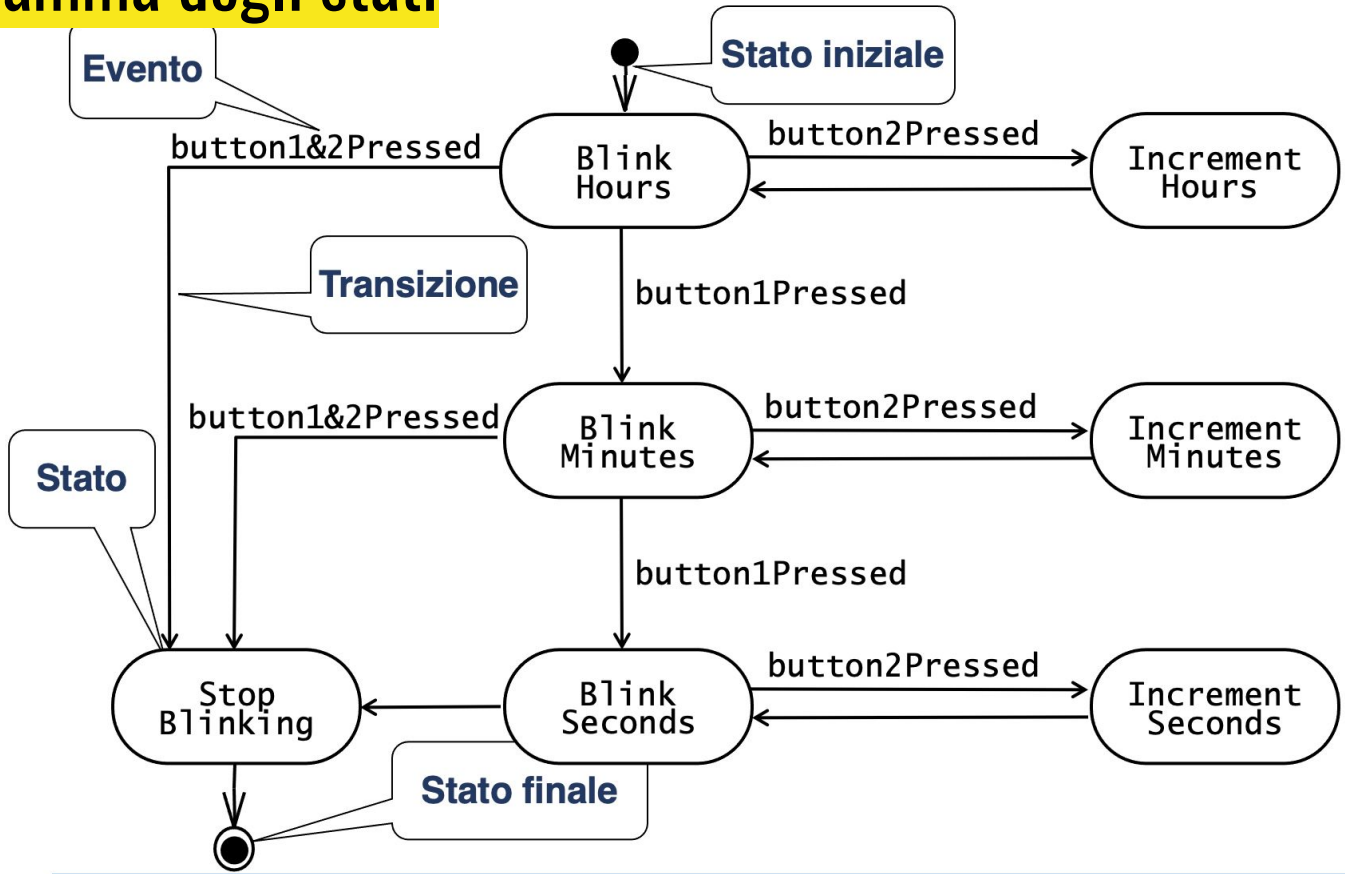


Proprietà diagrammi delle sequenze

- Rappresentano il comportamento in termini di interazioni
- Utile per identificare o trovare oggetti mancanti
- Comporta tempo per la preparazione, ma vale l'investimento
- Complementari al diagramma delle classi (che rappresenta la struttura)

Diagramma degli stati

Diagramma degli stati



Rappresenta il comportamento di un singolo oggetto con un comportamento dinamico interessante

Diagrammi degli stati

- Descrivono la sequenza di stati di un oggetto in seguito all'occorrenza di eventi esterni
- Sono estensioni del modello della macchina a stati finiti
  - Forniscono le notazioni per innestare stati e macchine degli stati
  - Forniscono notazioni per legare le transizioni agli invii di messaggi e le condizioni sugli oggetti
- Uno stato è una condizione soddisfatta dagli attributi di un oggetto
  - Ad esempio, un oggetto *Incident* in FRIEND può trovarsi in uno di quattro stati: **Active**, **Inactive**, **Closed**, **Archived**
    - Stato **Active**: situazione che richiede una risposta
    - Stato **Inactive**: situazione gestita ma i cui rapporti non sono stati ancora scritti
    - Stato **Closed**: situazione gestita e documentata
    - Stato **Archived**: incidente in stato Closed la cui documentazione è stata archiviata

La macchina a stati

- Una macchina a stati descrive il comportamento dinamico delle istanze di un classificatore (per esempio degli oggetti istanza di una classe).

- Per costruire una macchina a stati dobbiamo individuare gli stati significativi in cui si puo' trovare un oggetto durante la sua vita.

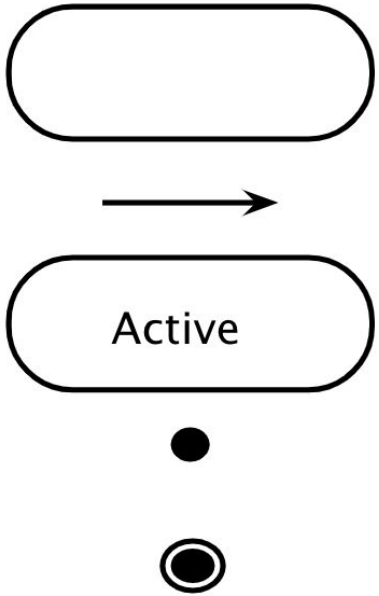
- Inoltre dobbiamo descrivere come da ciascuno di questi stati l'oggetto può passare (transire) in un altro.

- Le transizioni avvengono in risposta al verificarsi di un evento. Gli eventi sono tipicamente;
  - messaggi inviati da altri oggetti
  - eventi generati internamente

- Una macchina a stati è rappresentata con un grafo di stati e transizioni, associata a un classificatore

Transizioni e notazioni

- Una transizione rappresenta un cambiamento di stato attivato da eventi, condizioni o dal tempo
- Uno stato è rappresentato da un rettangolo arrotondato
- Una transizione è rappresentata da frecce che connettono due stati
- Gli stati sono etichettati col proprio nome
- Un circoletto nero rappresenta lo stato iniziale
- Un circoletto che circonda un circoletto nero rappresenta uno stato finale



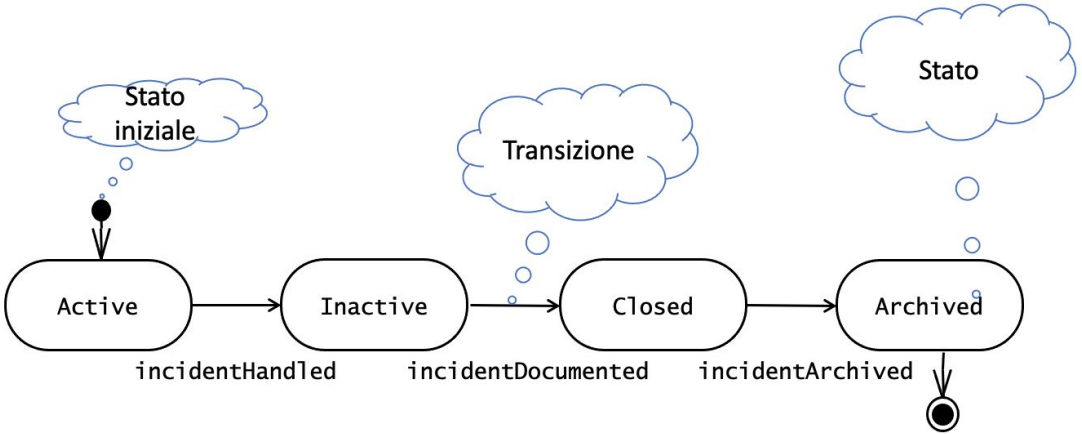
Esempio degli stati della vita di una lampadina

Descriviamo la vita di una lampadina

Diagramma degli stati per la classe Incident

I quattro stati possono essere rappresentati con un singolo attributo, **status,** nella classe Incident che può assumere uno dei quattro valori {*Active, Inactive, Closed, Archived*}

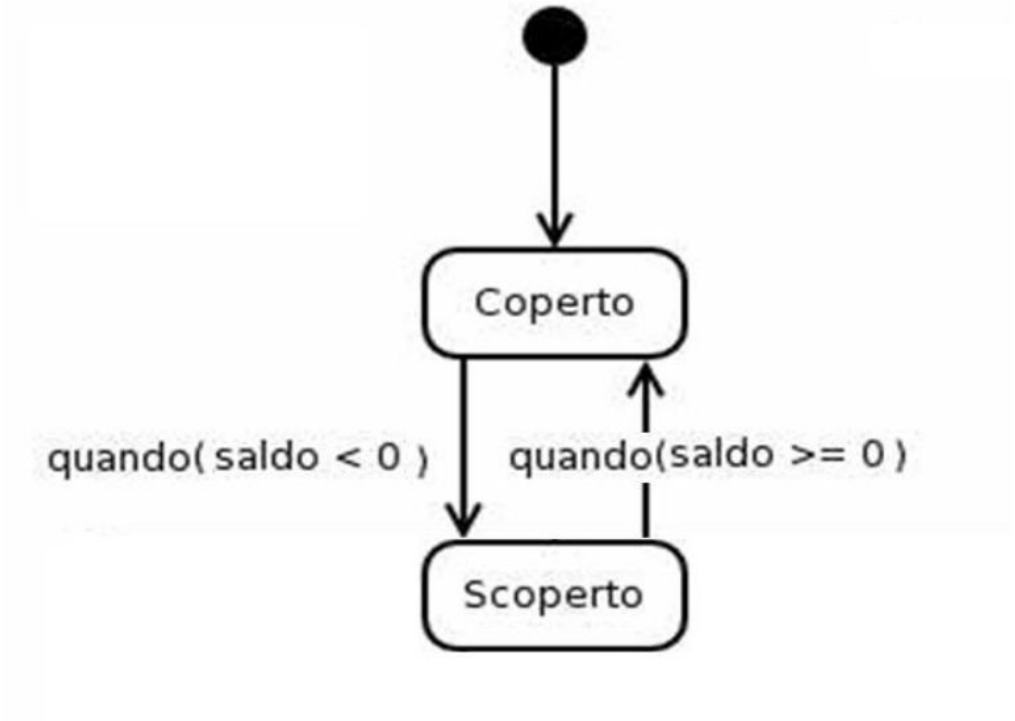
In generale uno stato può essere determinato dai valori più attributi



Evento

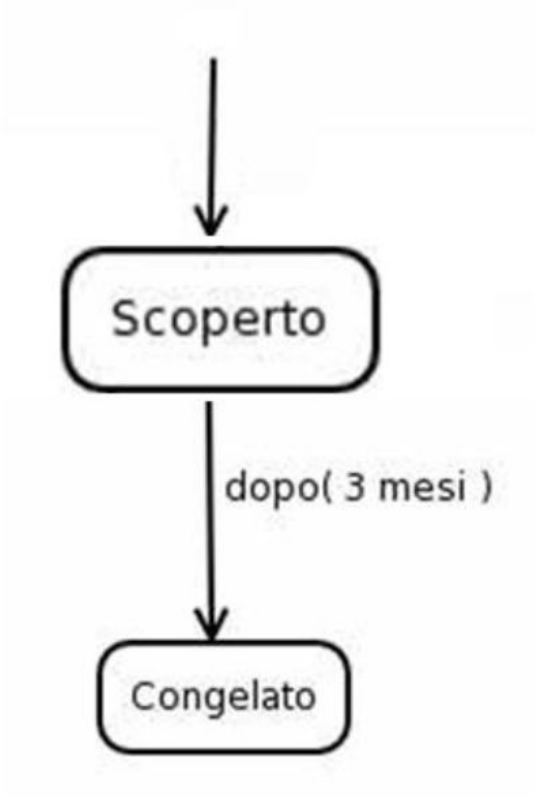
- Un evento è l'occorrenza di un fenomeno collocato nel tempo e nello spazio
- Un evento occorre istantaneamente
- Modellate qualcosa come un evento se ha delle conseguenze
- Gli eventi che arrivano in uno stato per cui non è prevista alcuna transizione etichettata con quell'evento vengono ignorati
- È ammesso il non–determinismo: un evento può fare da trigger a più transizioni:
  - Se le due transizioni escono dallo stesso stato, ne viene scelta una non-deterministicamente

Eventi variazioni dello stato



- Un evento occorre in modo istantaneo
- una condizione non è istantanea
- è istantaneo il momento in cui diventa vera

Eventi temporali

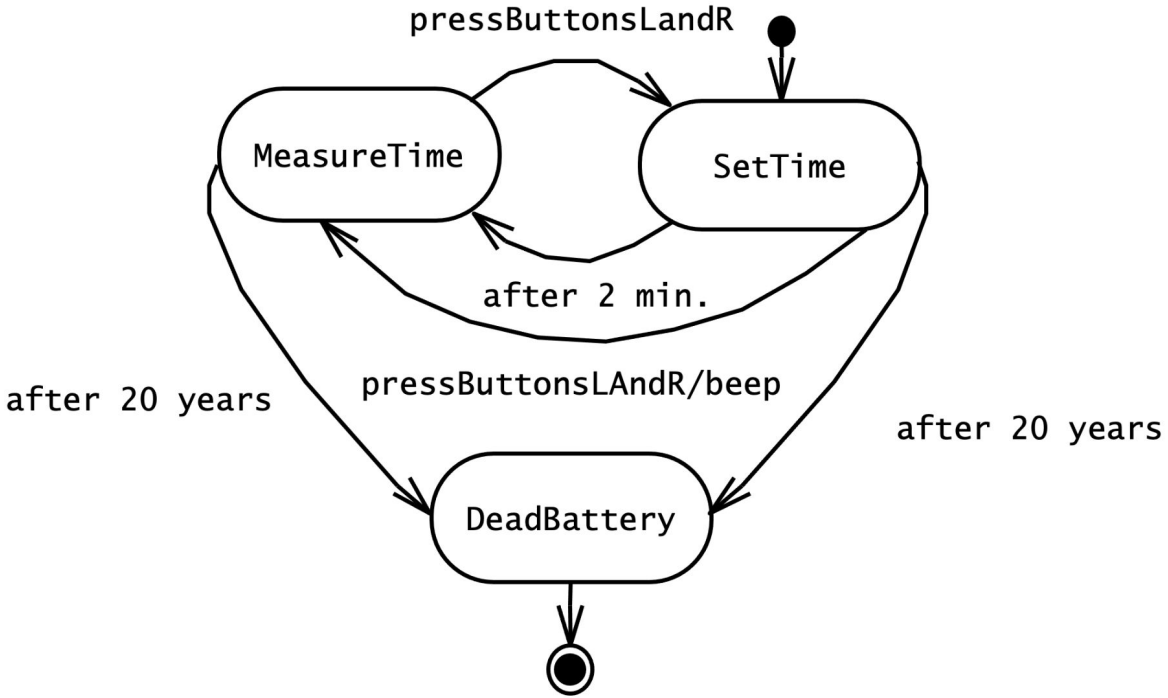


Dopo che l'oggetto è stato
3 mesi nello stato Scoperto,
transisce nello stato
Congelato

Esempio 2BWatch

- Al più alto livello di astrazione, 2Bwatch ha due stati, *MeasureTime* e *SetTime*
- 2Bwatch cambia stato quando l'utente preme e rilascia entrambi i pulsanti simultaneamente
- Durante la transizione dallo stato SetTime allo stato MeasureTime, 2Bwatch emette un beep
  - Indicato dall'azione **/beep** sulla transizione
- Quando 2Bwatch è acceso per la prima volta si trova nello stato *SetTime*
  - Indicato dal cerchietto nero che indica lo stato iniziale
- Quando la batteria si esaurisce, l'orologio non funziona
  - Cerchietto nero circondato da un altro cerchietto, lo stato finale

Esempio 2BWatch SetTime



Azioni

- Le azioni sono piccoli comportamenti atomici eseguiti in punti specifici nella macchina a stati
- Le azioni richiedono per l'esecuzione una breve quantità di tempo e **non possono essere interrotte**
- Le azioni possono verificarsi in tre luoghi
  - durante una transizione
  - quando si entra in uno stato
  - quando si esce da uno stato

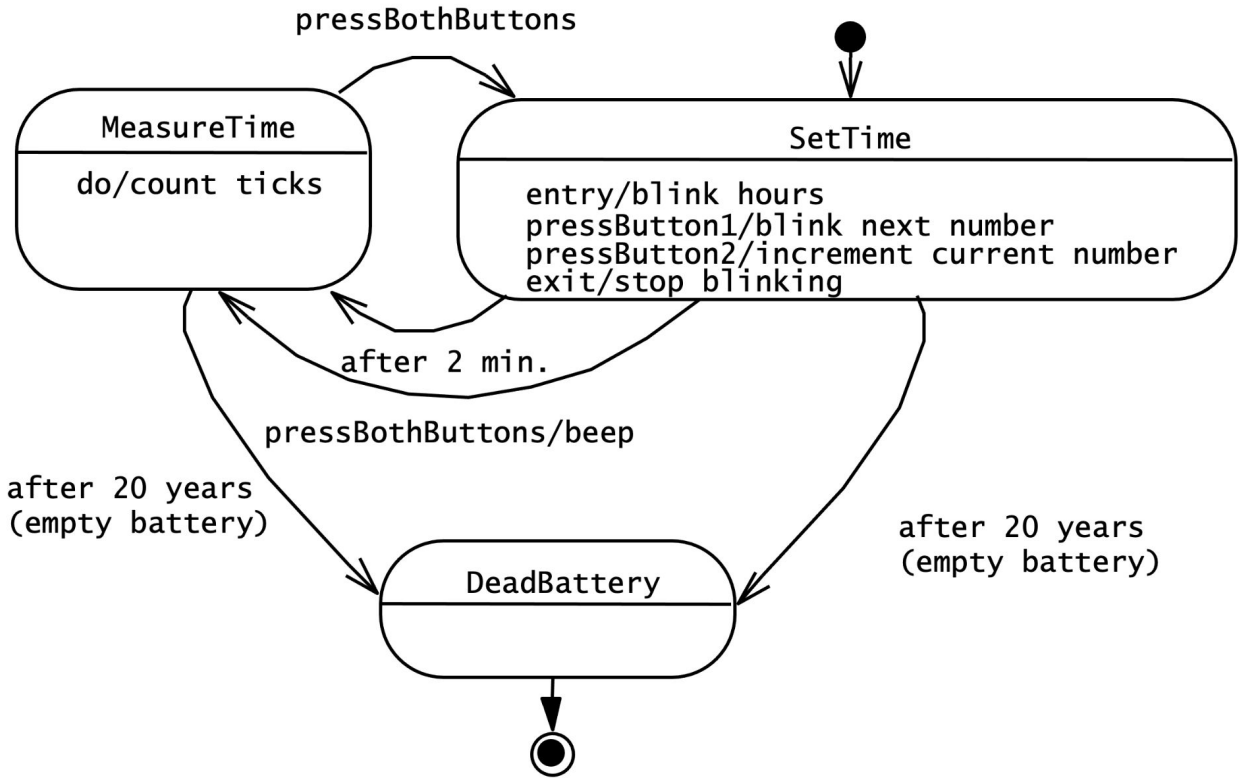
Transizioni interne

- Durante una transizione, le azioni di uscita dello stato sorgente sono eseguite per prime, poi sono eseguite le azioni associate con le transizioni e poi sono eseguite le azioni di entrata dello stato di destinazione
- Una **transizione interna** è una transizione che non lascia lo stato
  - Sono causate da eventi e possono avere delle azioni associate
  - L'attivazione di una transizione interna non comporta alcuna azione di uscita o entrata

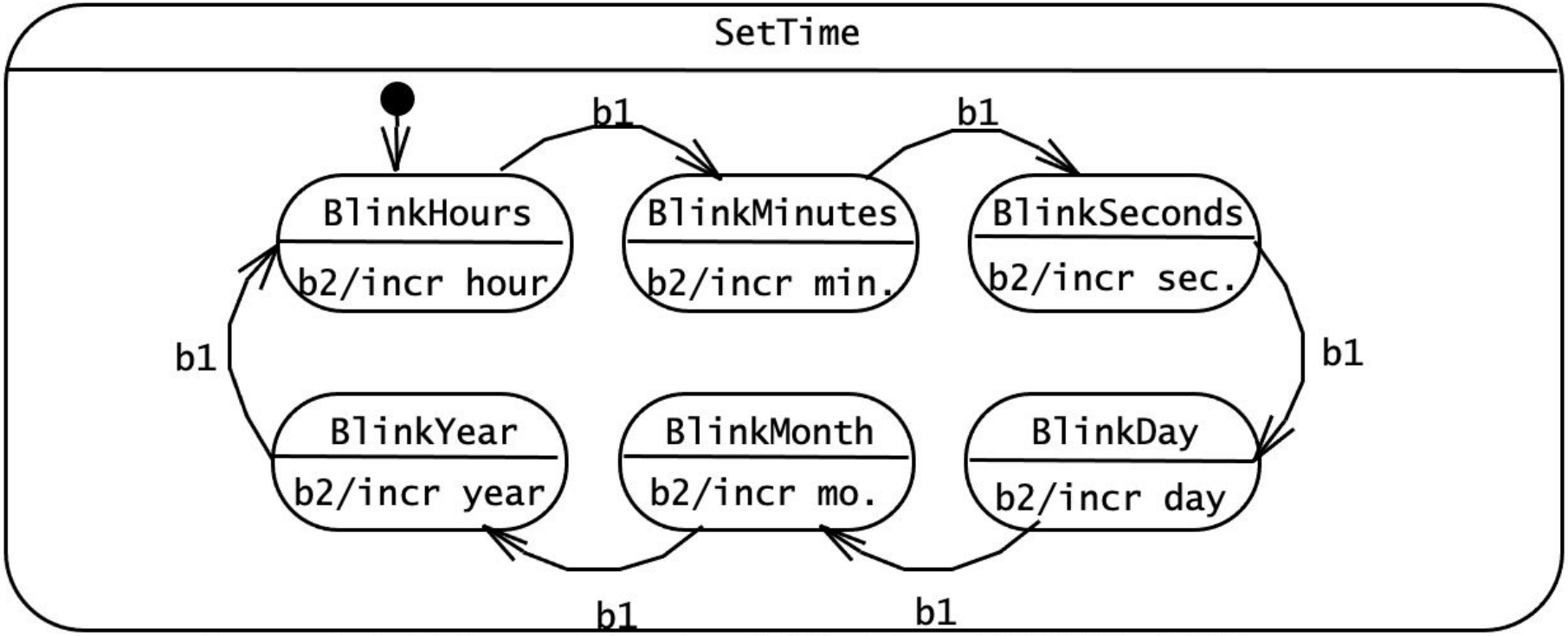
Attività

- Un’attività è un comportamento che è eseguito fintantoché un oggetto si trova in un dato stato
  - Un’azione è breve e non interrompibile, un’attività può richiedere un certo quantitativo di tempo ed è interrotta quando ha inizio una transizione che esce da uno stato
- Le attività sono rappresentate con l’etichetta *do* posizionata all’interno dello stato in cui è eseguita

Transizioni interne associate con lo stato SetTime



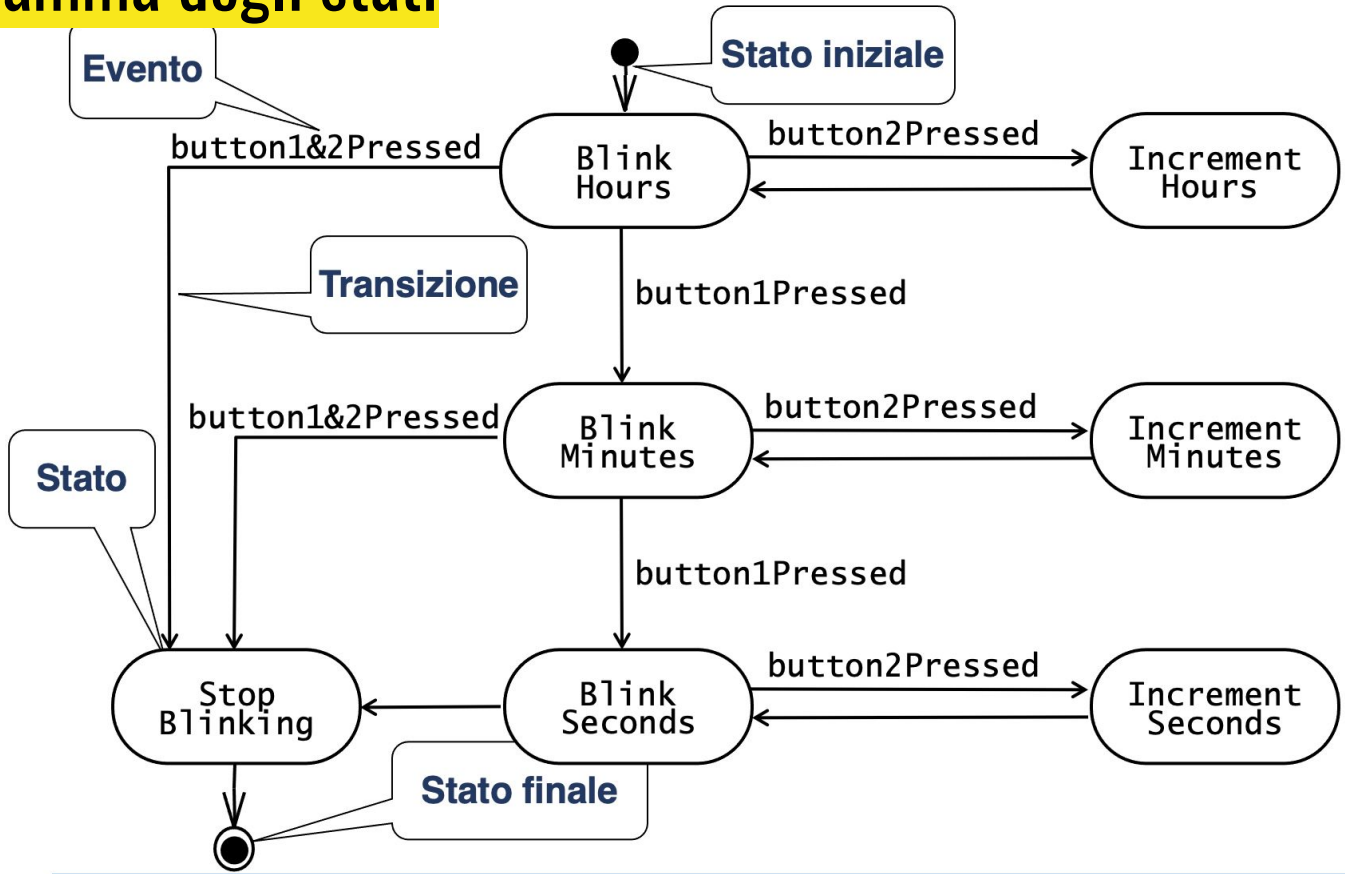
Diagrammi di stato annidati



Applicazioni diagrammi di stato

- Usati per rappresentare comportamenti non banali di un sottosistema o un oggetto
- A differenza dei diagrammi di interazione, che si focalizzano sugli eventi che influenzano il comportamento di un insieme di oggetti, i diagrammi di stato mettono a fuoco quale o quali attributi influenzano il comportamento di un singolo oggetto
- Usati per identificare attributi di oggetti e per rifinire tutte le descrizioni dei comportamenti di un oggetto
  - Invece, i diagrammi di interazione sono usati per identificare gli oggetti partecipanti ed i servizi che forniscono

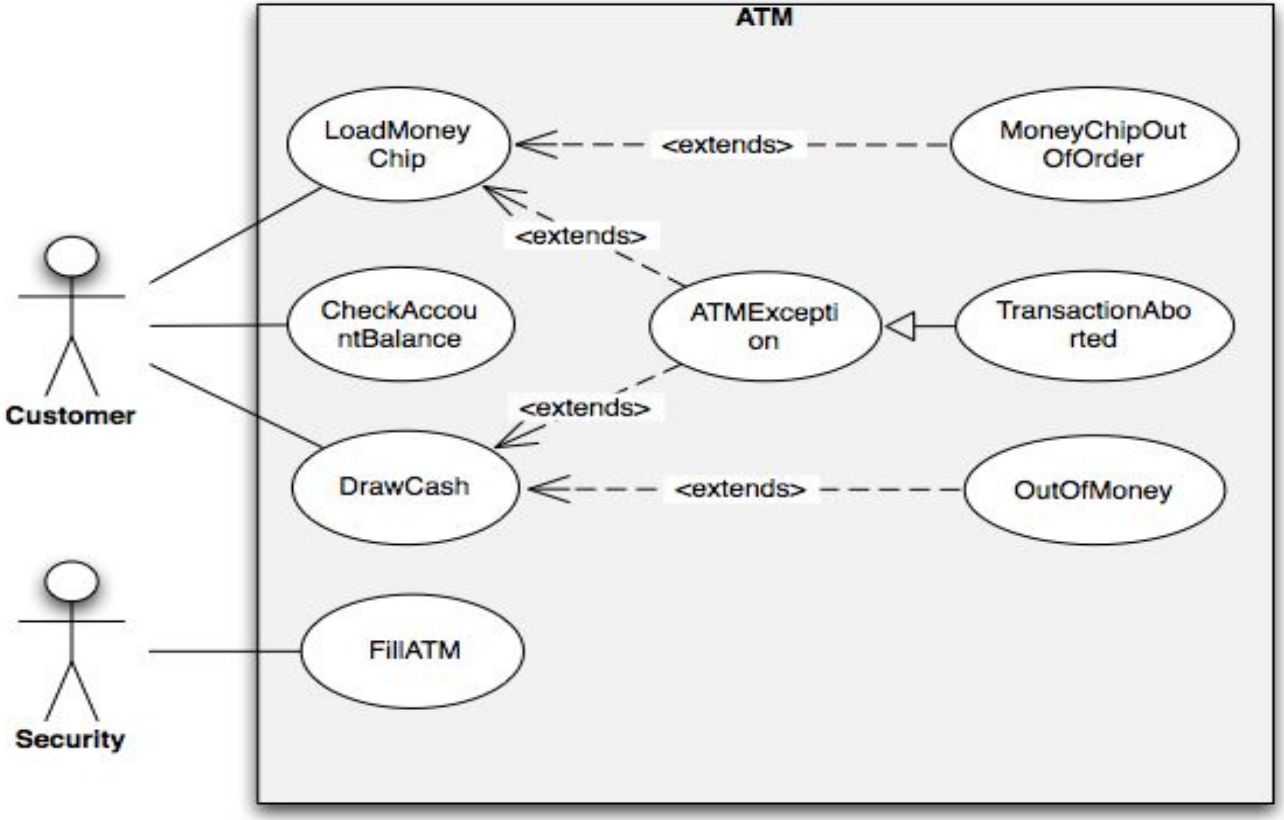
Diagramma degli stati



Rappresenta il comportamento di un singolo oggetto con un comportamento dinamico interessante

Esercizio : modellare un Bancomat

- ***Modellare un Bancomat***
- Disegnare un diagramma dei casi d'uso.
  - Il sistema comprende due attori: un **cliente**, che preleva denaro dal suo conto, in contanti o ricaricando su chip il suo bancomat, e un **addetto alla sicurezza**, che inserisce denaro nel bancomat.
  - I casi d'uso dovrebbero includere: *PrelevaContanti*, *CaricaDenaro*, *ControllaSaldo*, *RiempiBancomat*. Include anche i seguenti casi eccezionali: *DenaroEsaurito*, *TransizioneAnnullata* (cioé, il cliente ha selezionato il tasto Cancella senza completare la transazione) e *DenaroSuChipFuoriServizio*.
  - E' possibile usare l'ereditarietà tra casi d'uso
- Scrivere il flusso di eventi e specificare tutti i campi per il caso d'uso *ControllaSaldo*
  - Specificare le relazioni


ATM
LoadMoney Chip
<extends>
MoneyChipOut OfOrder
<extends>
CheckAccou ntBalance
ATMExcepti on
TransactionAbo rted
<extends>
DrawCash
<extends>
OutOfMoney
FillATM
Customer
Security

Use case name	CheckAccountBalance	
Participating Actors	Initiated by Customer	
Flow of events	1. The Customer inserts his bank card into the ATM.	
		2. The ATM welcomes the Customer and opens the main menu.
		3. The ATM waits for the selection from the Customer.
	4. The Customer selects the “CheckAccountBalance” function by pressing the appropriate button.	
		5. The ATM system shows the enter PIN screen to the Customer.
	6. The Customer enters his correct PIN and confirms it.	
		7. The ATM system shows the actual balance of the Customer’s account.
Entry condition	The ATM is working and Customer is in front of the terminal.	
Exit condition	The Customer can view his account balance.	
Quality requirements	The ATM stays online for the whole period of the CheckAccountBalance use case.	

Diagrammi delle attività

Diagrammi delle attività

- Forniscono la sequenza di operazioni che definiscono un'attività più complessa
- Permettono di rappresentare processi paralleli e la loro sincronizzazione
- Possono essere considerati Diagrammi di stato particolari
  - Ogni stato contiene (è) un'azione
- Un diagramma delle attività può essere associato
  - A una classe
  - All'implementazione di un'operazione
  - Ad un caso d'uso

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)

L'attività

- 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'.

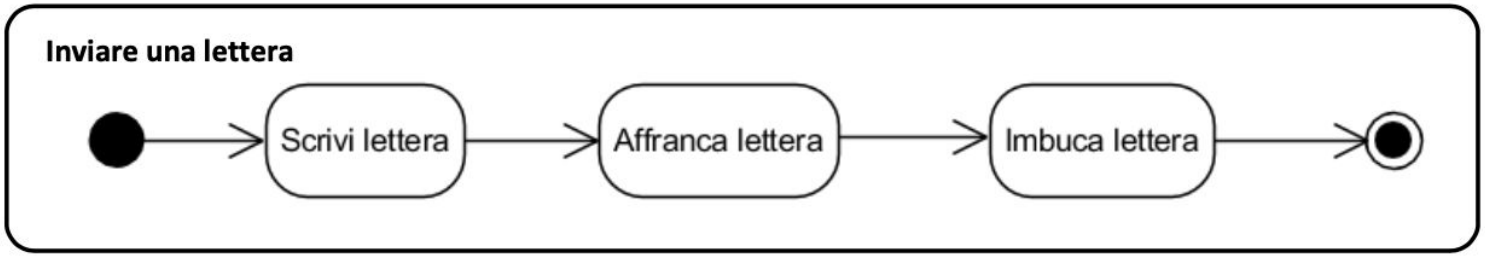
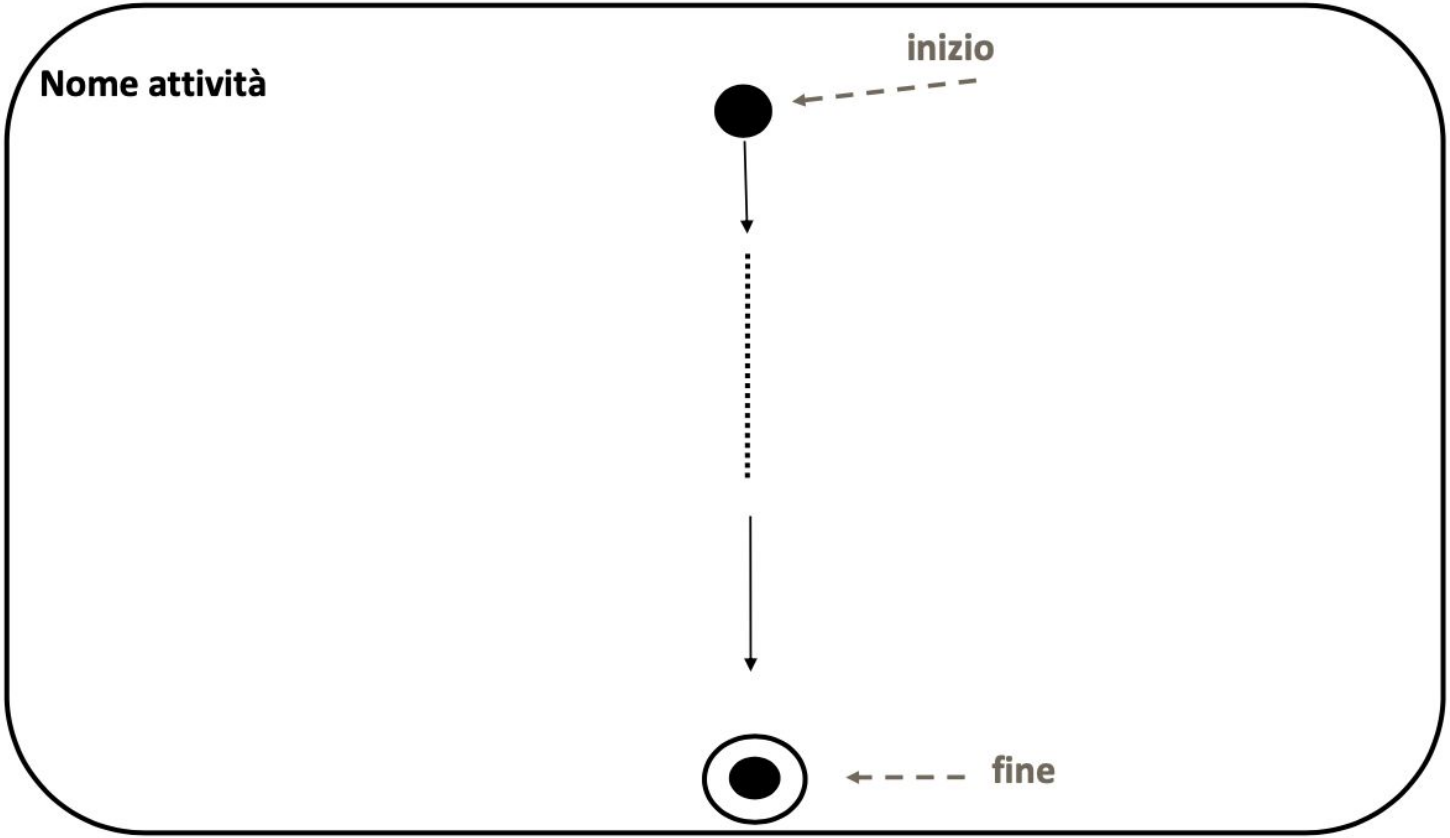


Diagramma delle attività

- Ogni diagramma delle attività ha due nodi particolari: *Inizio* e *Fine*
  - **Inizio** è il punto di partenza del diagramma, indica la prima azione da eseguire ed è rappresentato da un cerchio con solo archi in uscita
  - **Fine**: indica la conclusione dello scenario descritto e ha solo archi in entrata. L'azione finale del diagramma deve puntare sempre al nodo finale

Diagrammi di attività: inizio e fine

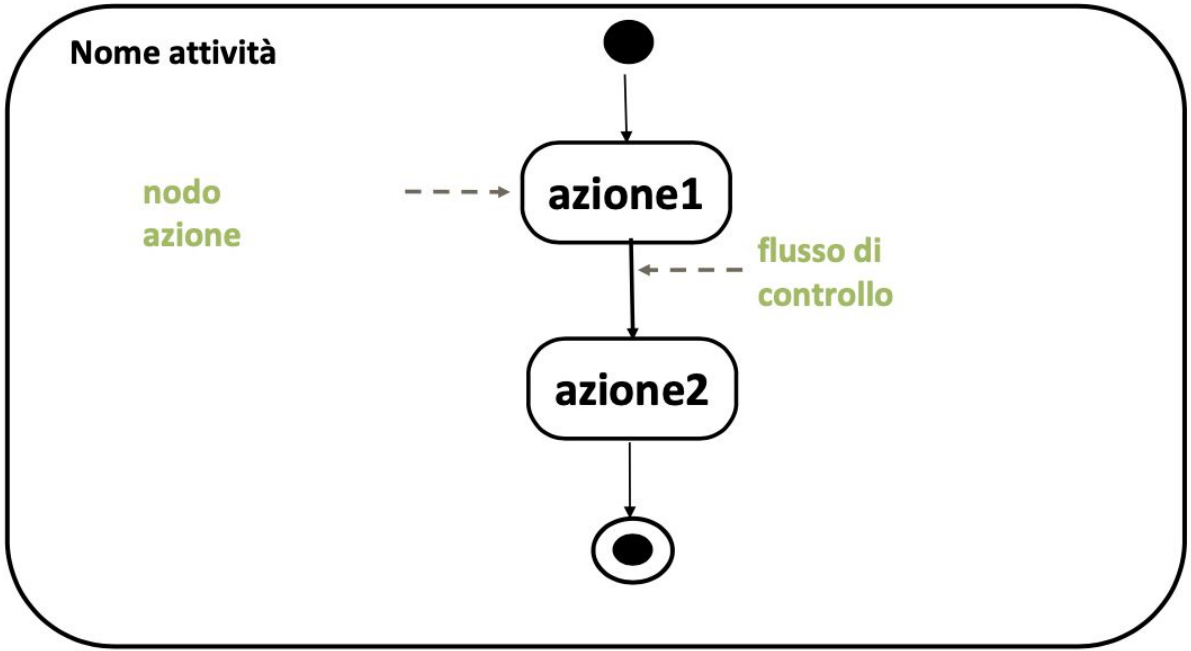


Le azioni

- 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)

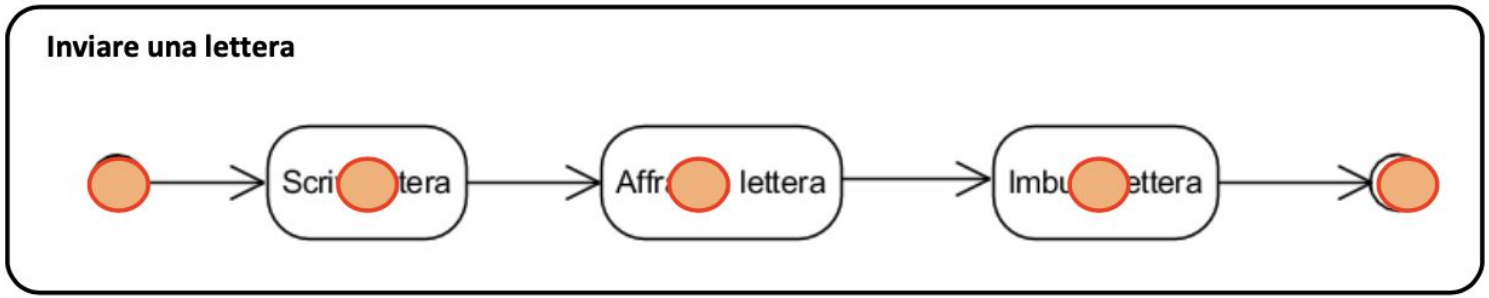
Diagrammi di attività: nodo azione



- 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



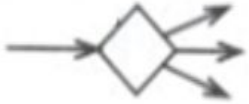
- La semantica è descritta con il token game: l'azione può essere eseguita quando riceve il token

Nodi di controllo

nodo iniziale

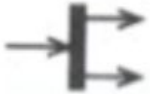
nodo finale

nodo di fine flusso



nodo decisione (con guardie sulle frecce uscenti)

nodo fusione



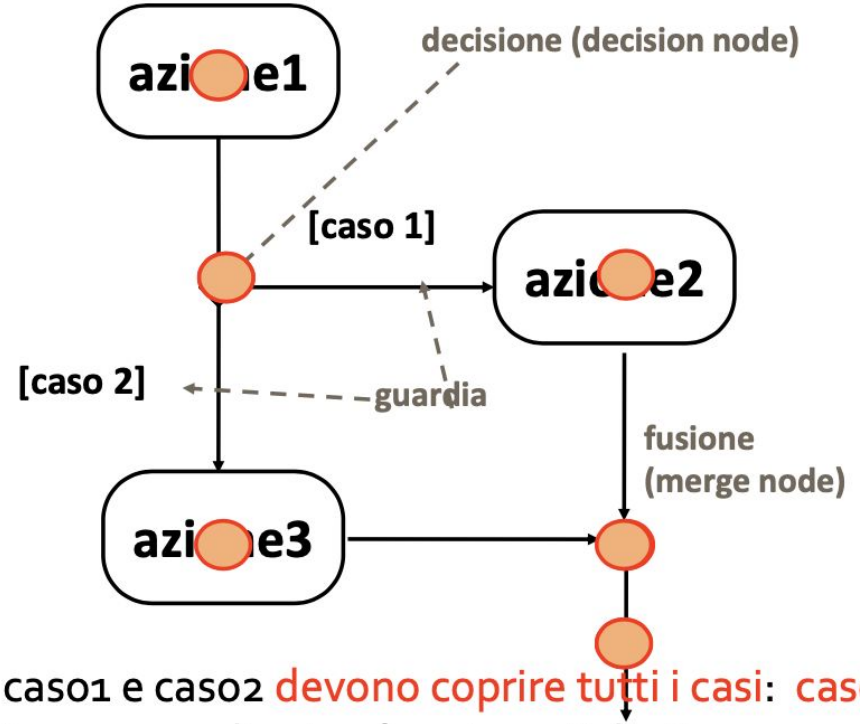
nodo di biforcazione (fork)

nodo di sincronizzazione (join)

Diagramma delle attività: 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"

Esempio : classe Incident

- Un'azione è un'attività atomica: è rappresentata nel diagramma con un rettangolo arrotondato ed è identificata con una voce verbale che descrive l'azione stessa
  - *HandleIncident*: il *Dispatcher* riceve i rapporti e alloca risorse
  - *DocumentIncident*: tutti i *FieldOfficer* partecipanti e i *Dispatcher* documentano l'incidente dopo che è stato chiuso
  - *ArchiveIncident*: archiviazione delle informazioni relative all'incidente su dispositivi di memorizzazione

- Gli archi tra le attività rappresentano il flusso di controllo. Un'attività può essere eseguita solo dopo che tutte le attività precedenti sono state completate

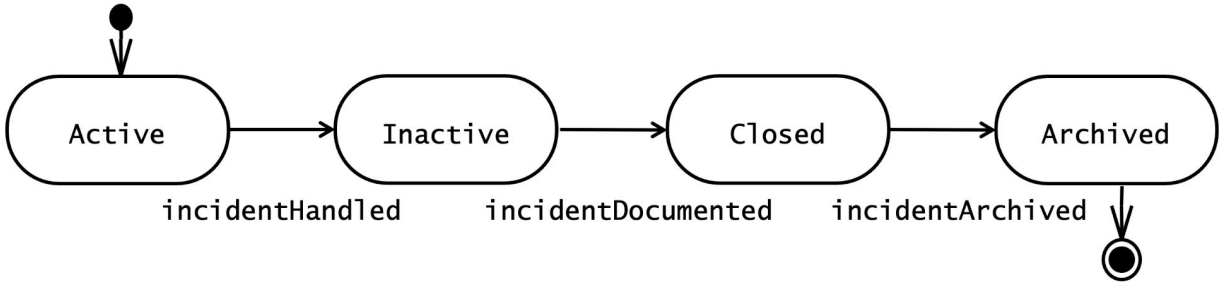


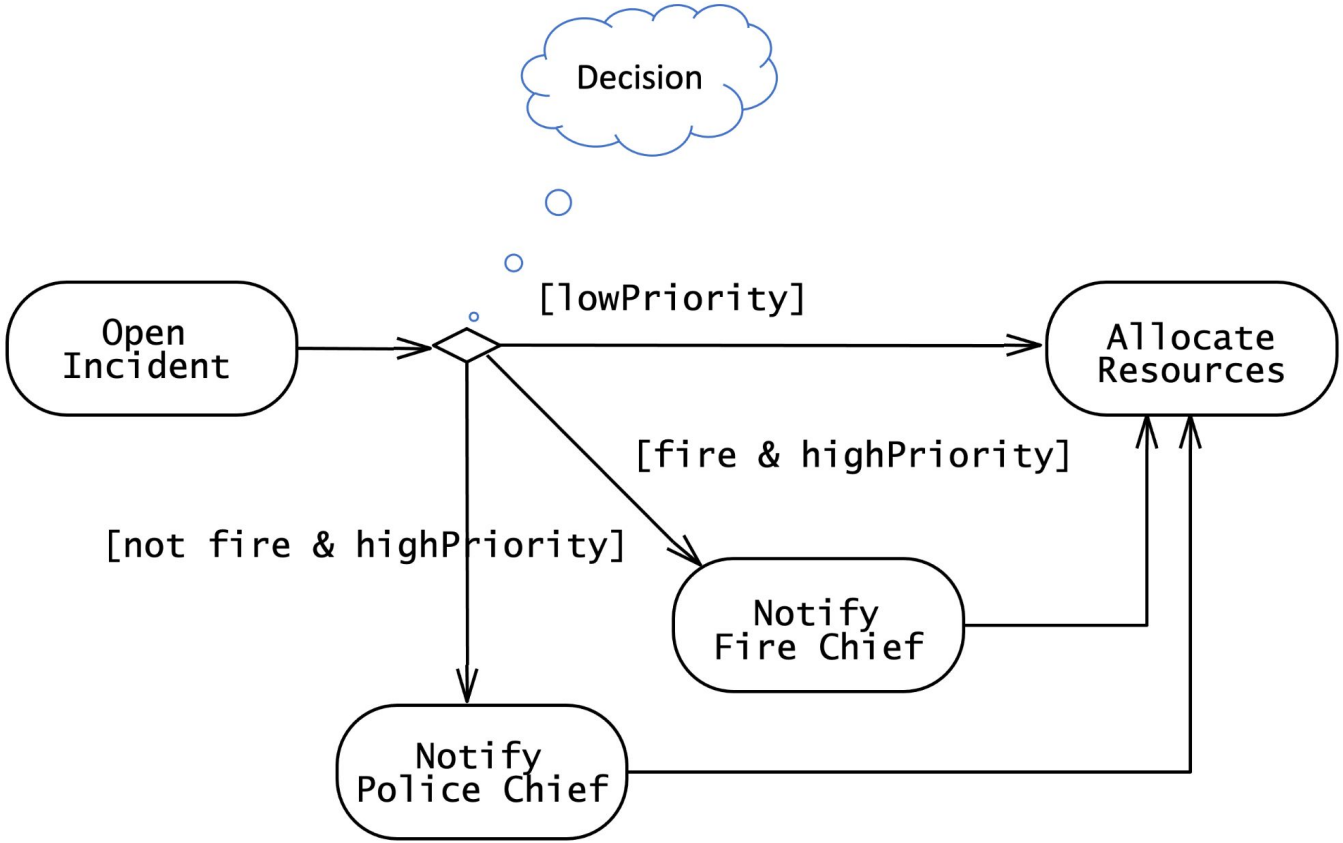
Diagramma delle attività

- Un arco, o flusso, collega tra loro i nodi
  - L'insieme degli archi del diagramma rappresenta il flusso di esecuzione complessivo
  - E' rappresentato con una freccia
- Il flusso delle azioni progredisce solo nel momento in cui l'azione considerata è completata
- Una funzionalità complessa non è costituita da una semplice successione di azioni in sequenza:
  - può presentare azioni da eseguire contemporaneamente o sotto particolari condizioni

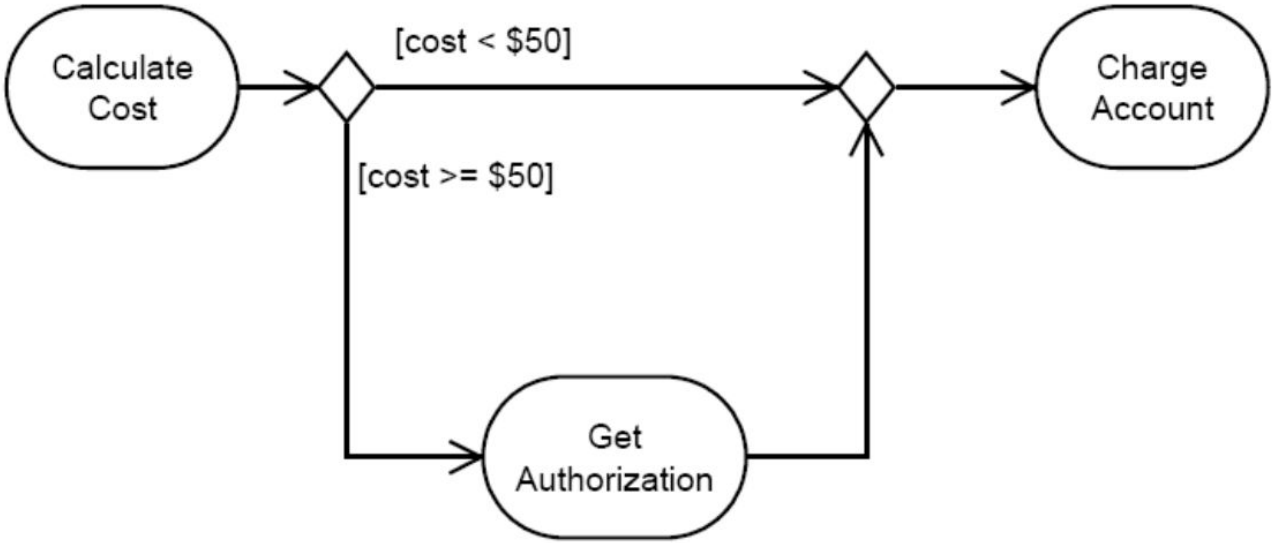
Nodi di controllo

- **Decision**
  - Indica che l'esecuzione di un'azione dipende dal verificarsi di una determinata condizione chiamata **guard**. Descrive il fatto che al termine di una particolare azione, lo scenario può proseguire in modo diverso con azioni che dipendono da specifiche situazioni che si vengono a verificare.
    - E' rappresentato con un diamante puntato da un arco e dal quale escono almeno due archi
- **Merge**
  - Duale del nodo decision e descrive un punto del processo in cui si ricongiungono due o più flussi alternativi
    - Rappresentato anch'esso da un diamante

Esempio di decisioni in OpenIncident



Decisione e fusione – decision and merge

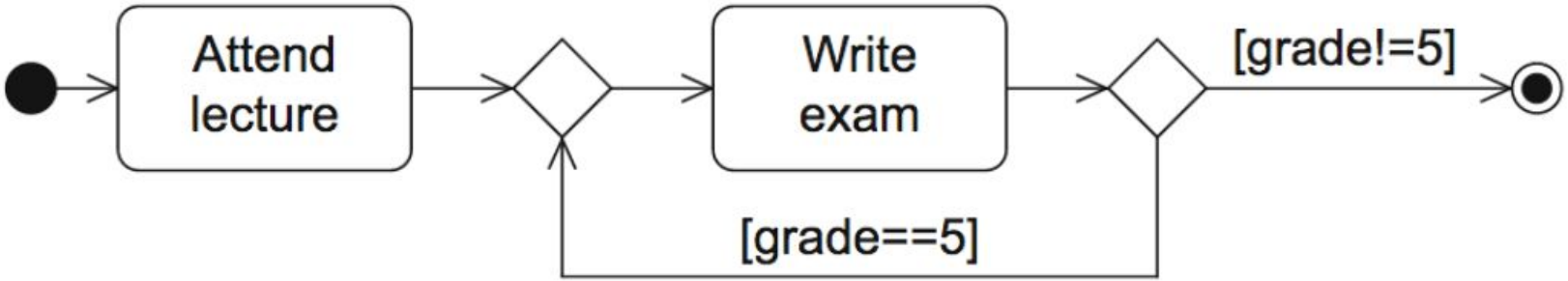


- Il token prende uno dei cammini
- Deve prendere sempre uno dei cammini

Semantica:

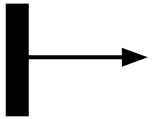
- Le guardie devono coprire tutte le possibilità
  - In caso si usa [else]
- E' bene (ma non necessario) che siano mutualmente esclusive altrimenti comportamento non definito (non deterministico).
- Le condizioni di guardia sempre tra [ ]
  - (in generale in UML)
- Dato un nodo decisione non è obbligatorio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

Loops



Nodi di controllo

- **Fork**
  - Descrive l'esecuzione in parallelo di più azioni: quelle puntate dal nodo fork sono avviate contemporaneamente ed in parallelo
    - rappresentato da una barra nera puntata da un solo arco e dalla quale partono due o più archi verso le azioni da eseguire in parallelo



- **Join**
  - E' il duale del fork. Specifica che un'azione è eseguita solo nel momento in cui le azioni precedenti hanno terminato la propria esecuzione. E' definito anche sincronizzazione perché sincronizza due rami svolti parallelamente, generando un unico flusso di esecuzione.
    - E' rappresentato da una barra nera

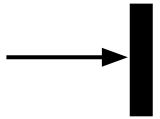


Diagramma di attività: fork e join

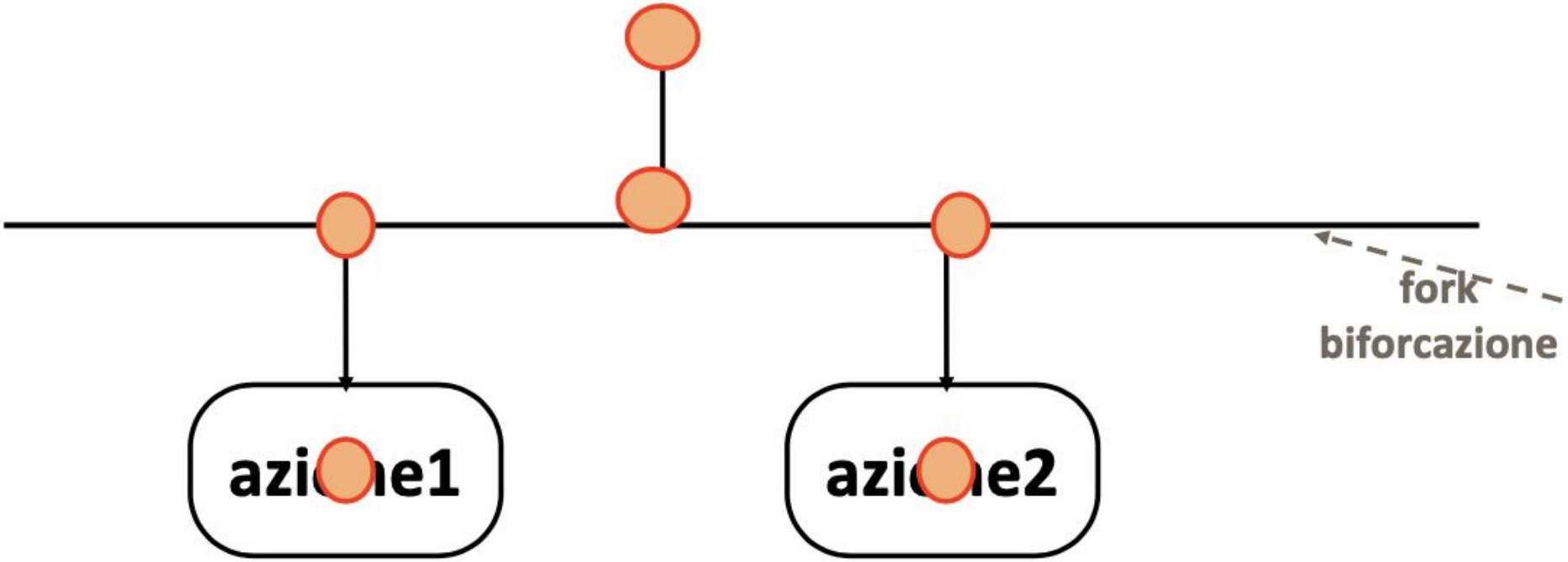
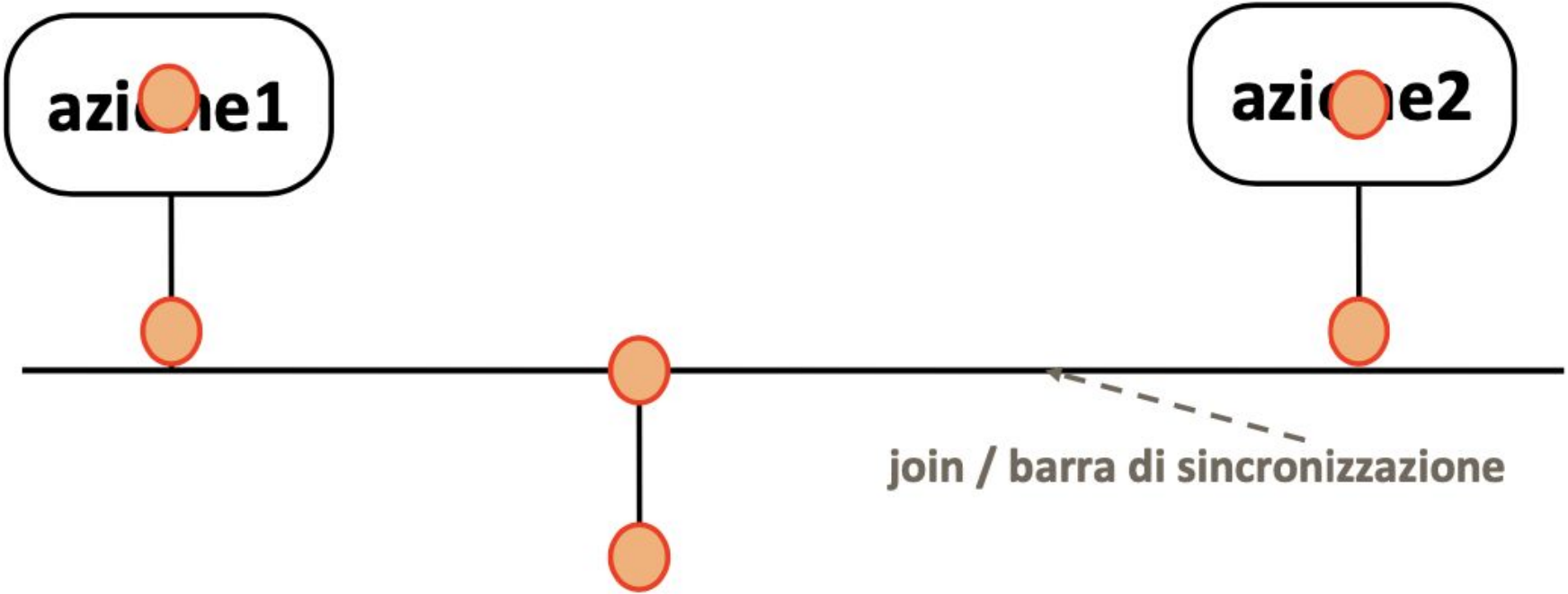


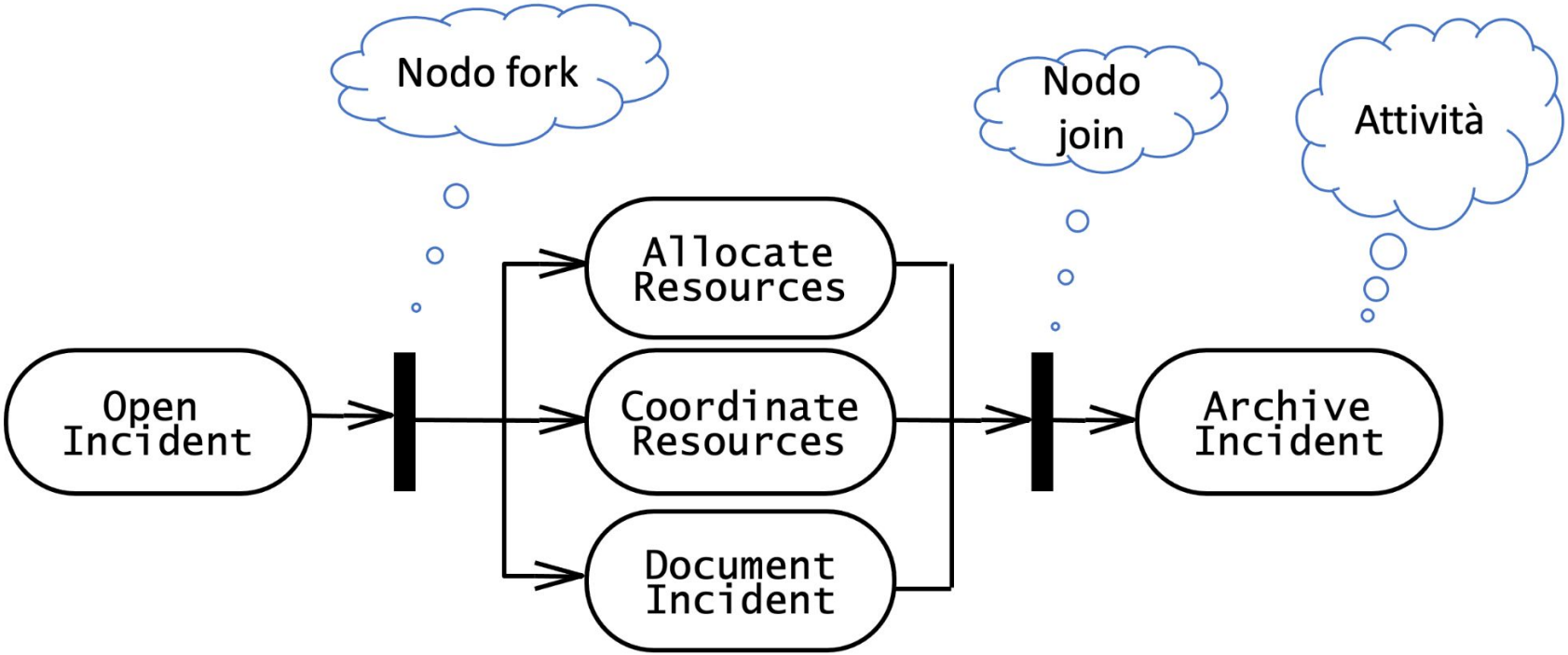
Diagramma di attività: fork e join



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

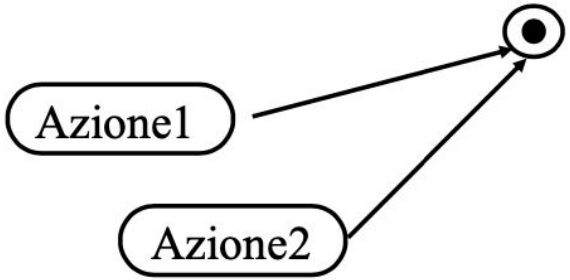
Esempio di fork e join in OpenIncident



Nodo di fine attività

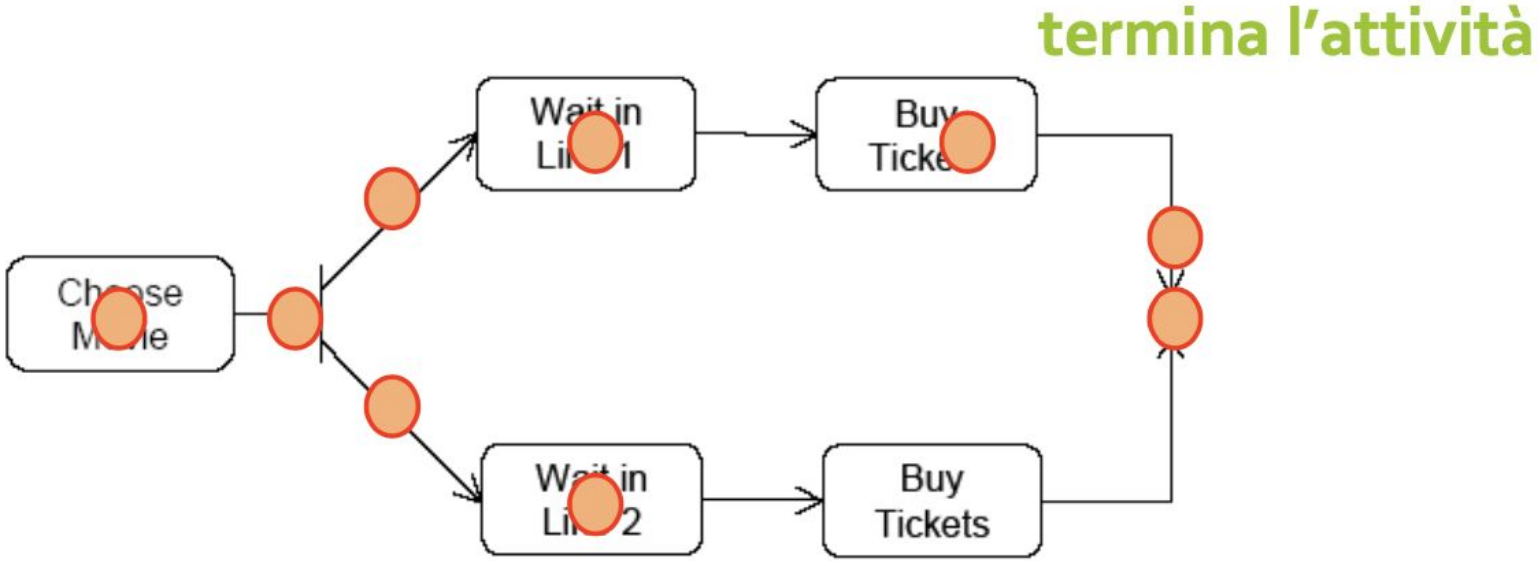
- Se un token raggiunge un nodo di fine attività , l'intera attività è terminata

- 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

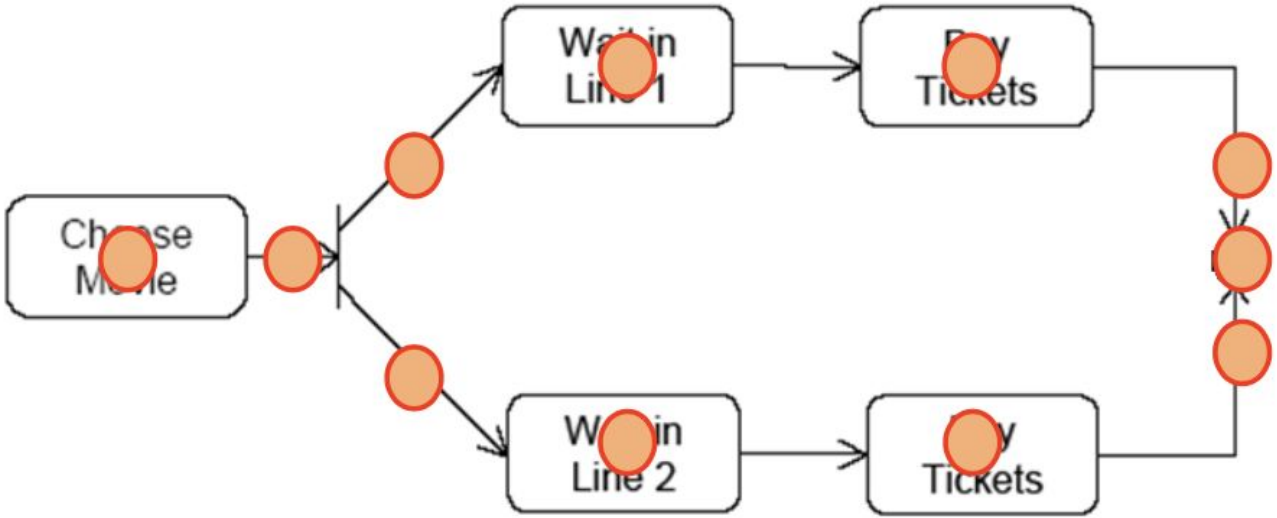


Nodo di fine flusso

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

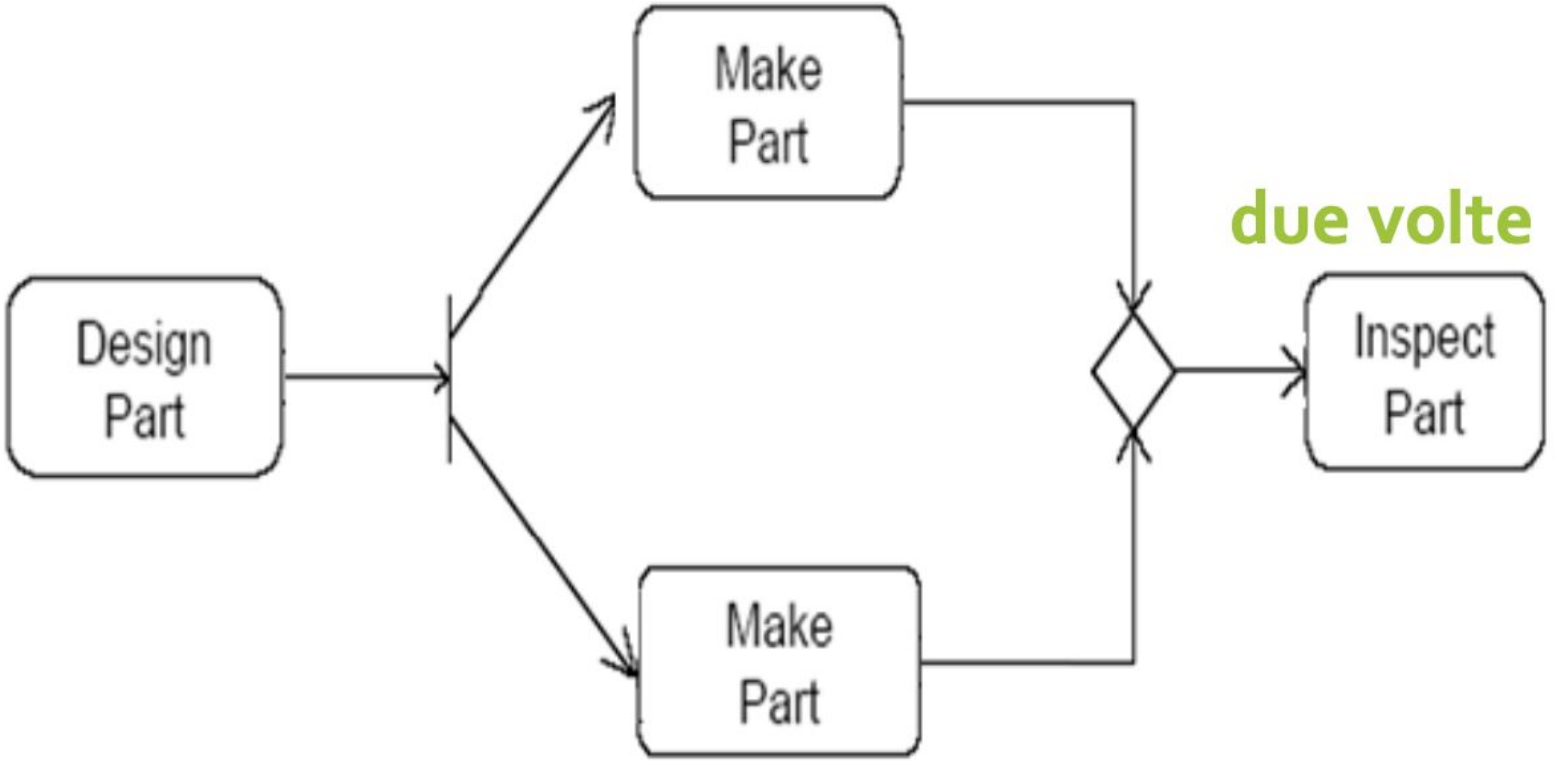
Nodo di fine flusso

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

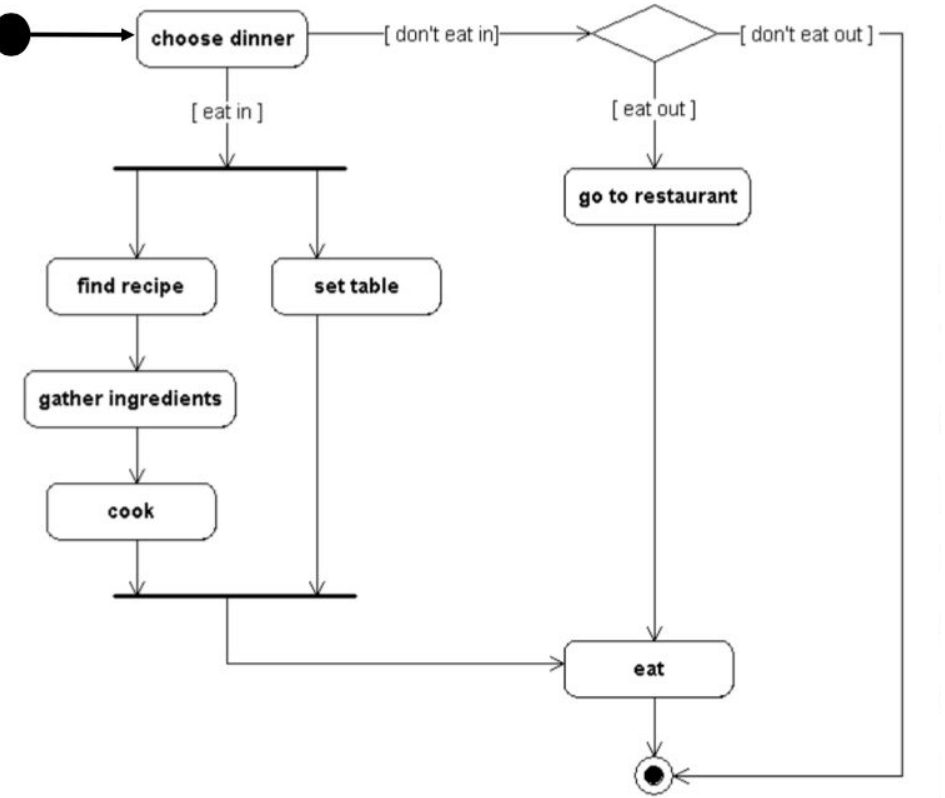


non termina l'attività, solamente si consuma il token

Fork e merge: possibile ma azioni eseguite due volte



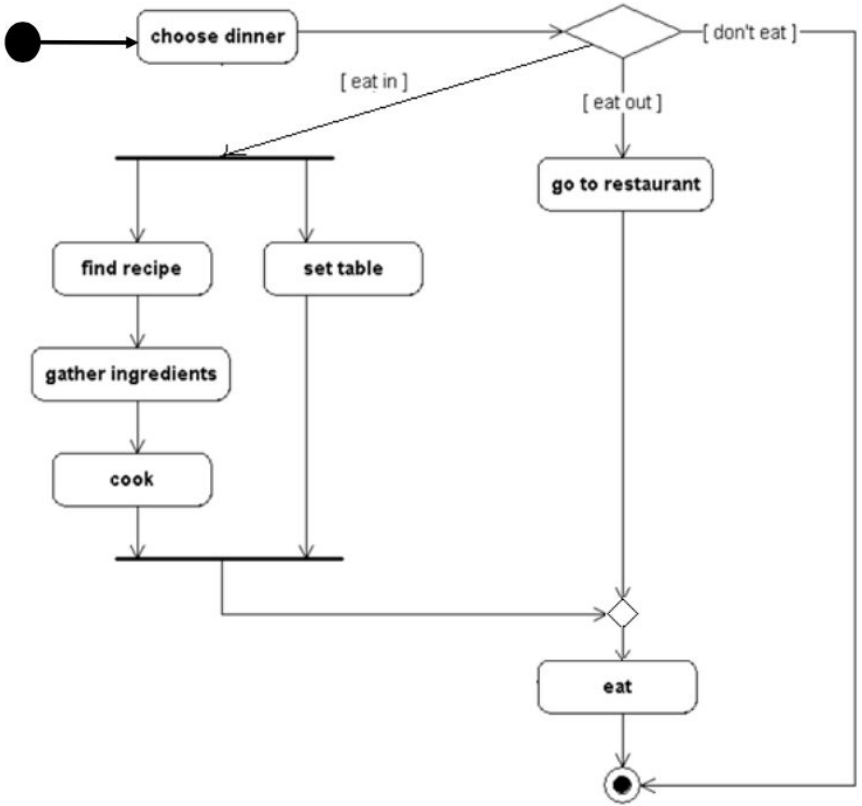
Esempio preso dal web, interessante perchè sbagliato



Anche se UML permette frecce multiple entranti/uscenti in/da un nodo, se ne sconsiglia 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 nodo fusione e dopo choose dinner serve un nodo decisione.

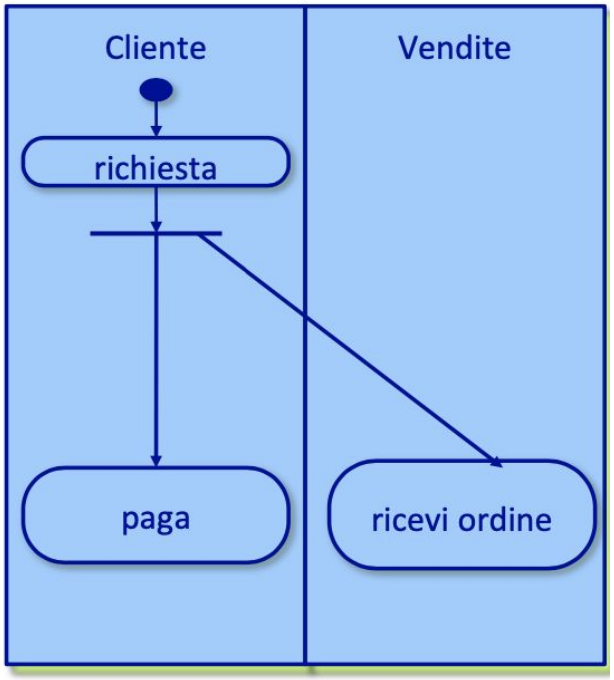
Sono tollerate due frecce entranti nello stato finale.

Swimlane o partizione

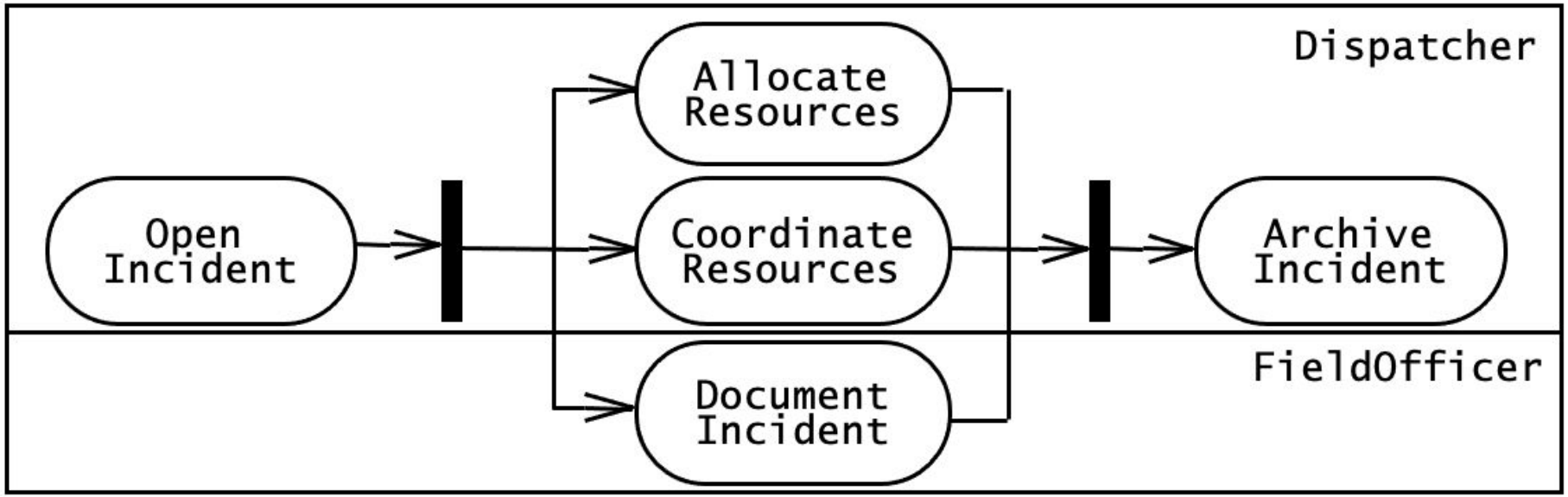
- Le attività possono essere raggruppate in ***swimlane*** per denotare l'oggetto o sottosistema che le implementa
  - Le swimlane sono rappresentate come rettangoli che racchiudono un gruppo di attività
  - Le transizioni possono attraversare le swimlane

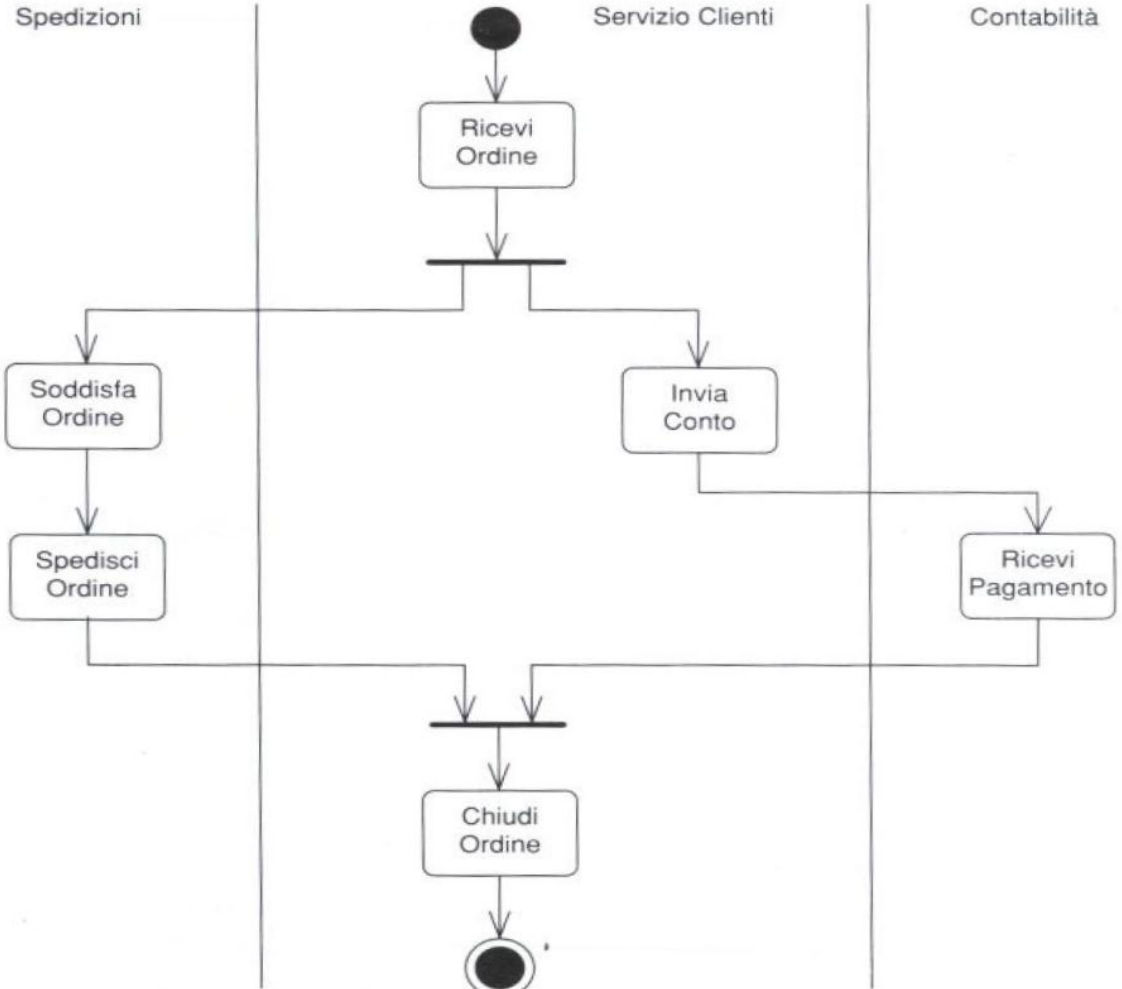
Una partizione (o swimlane) divide le azioni in gruppi.

Permette di assegnare le responsabilità delle azioni.



Esempio di swimlane




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

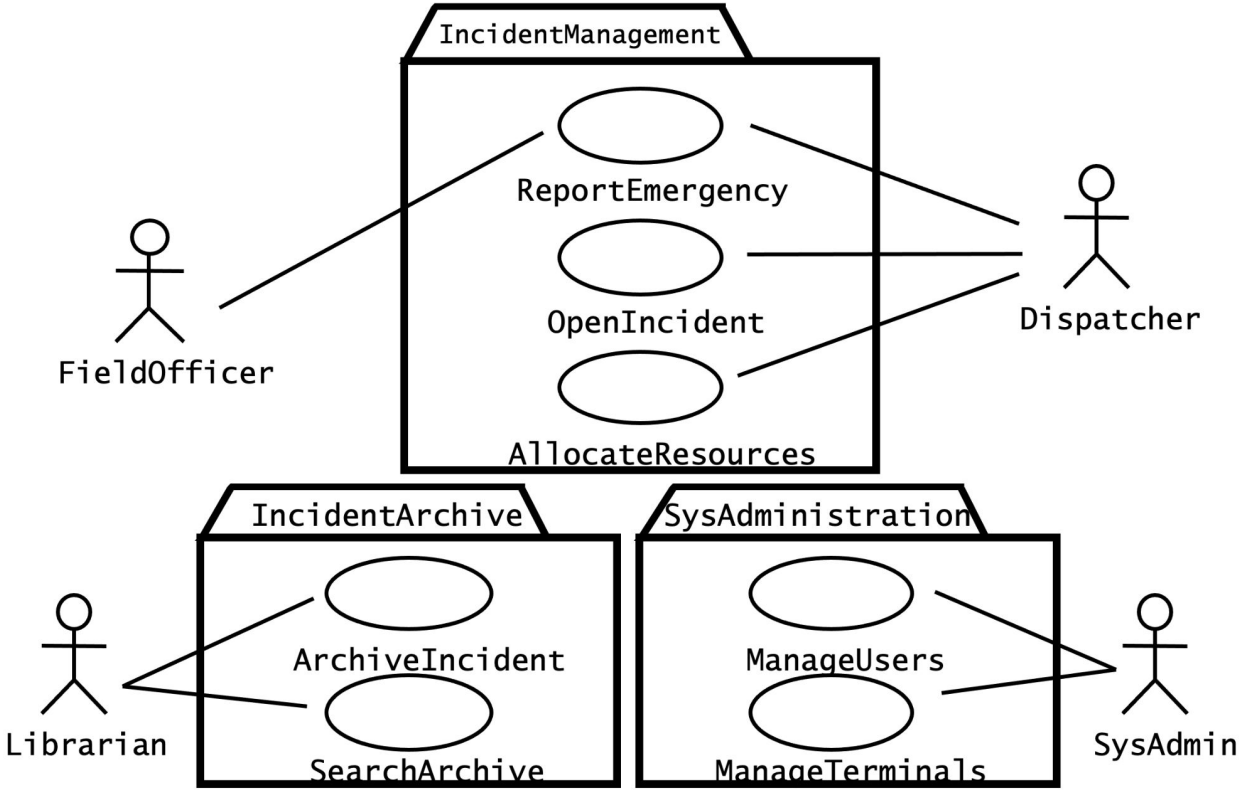
Applicazione dei diagrammi delle attività

- I diagrammi delle attività forniscono una visione task-centrica del comportamento di un insieme di oggetti
- Possono essere usati per descrivere vincoli di sequenza tra i casi d'uso, attività sequenziali tra gruppi di oggetti, o task di un progetto

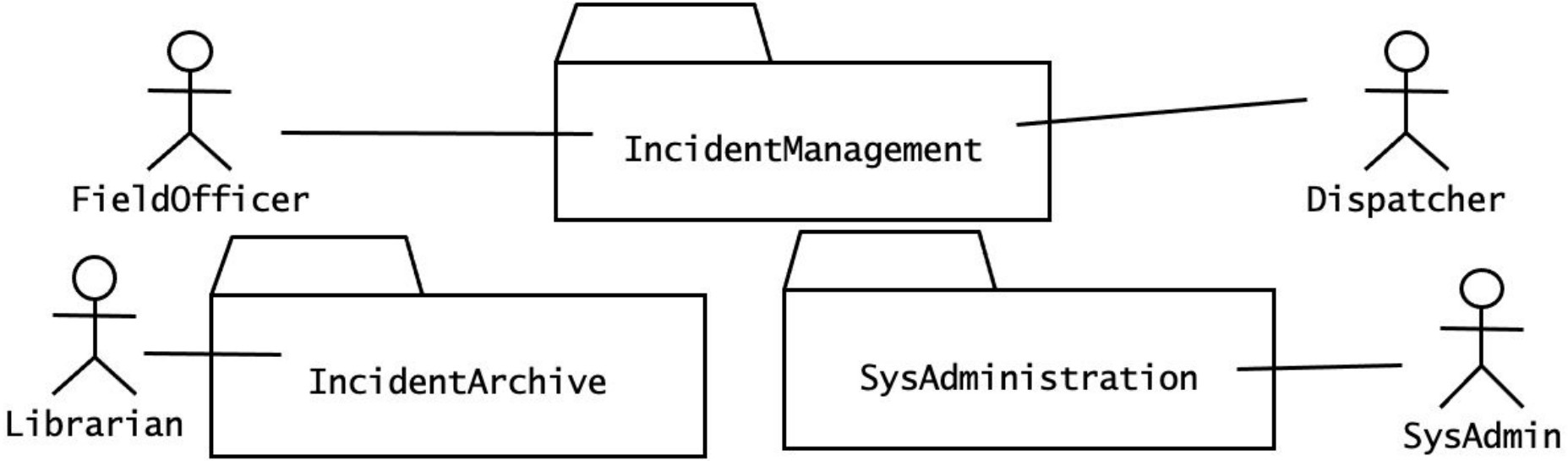
Organizzazione del diagrammi

- I modelli di sistemi complessi diventano velocemente anch'essi complessi quando gli sviluppatori li revisionano e rifiniscono
- La complessità dei modelli può essere gestita raggruppando elementi in relazione tra loro in **package**
- Un package è un raggruppamento di elementi, come casi d'uso, classi, o attività che semplificano la comprensione

Esempio di package: casi d'uso di FRIEND orgnizzati per attori

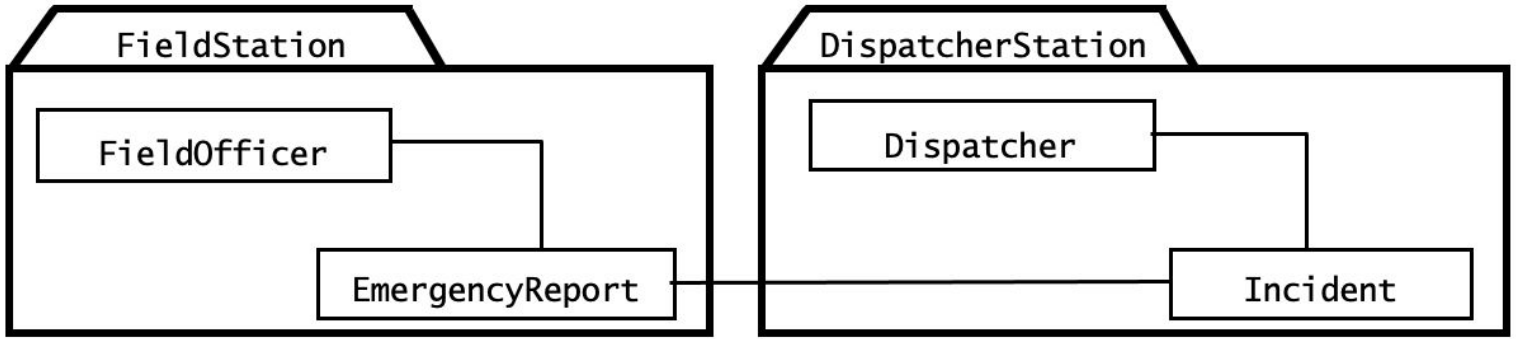


Esempio di package



Package nei diagrammi delle classi

- Anche i diagrammi delle classi possono essere organizzati in package
- Le classi del caso d'uso *ReportEmergency* sono organizzate rispetto al punto in cui gli oggetti sono creati
  - *FieldOfficer* e *EmergencyReport* sono parte del package FieldStation
  - *Dispatcher* e *Incident* sono parte di *DispatcherStation*



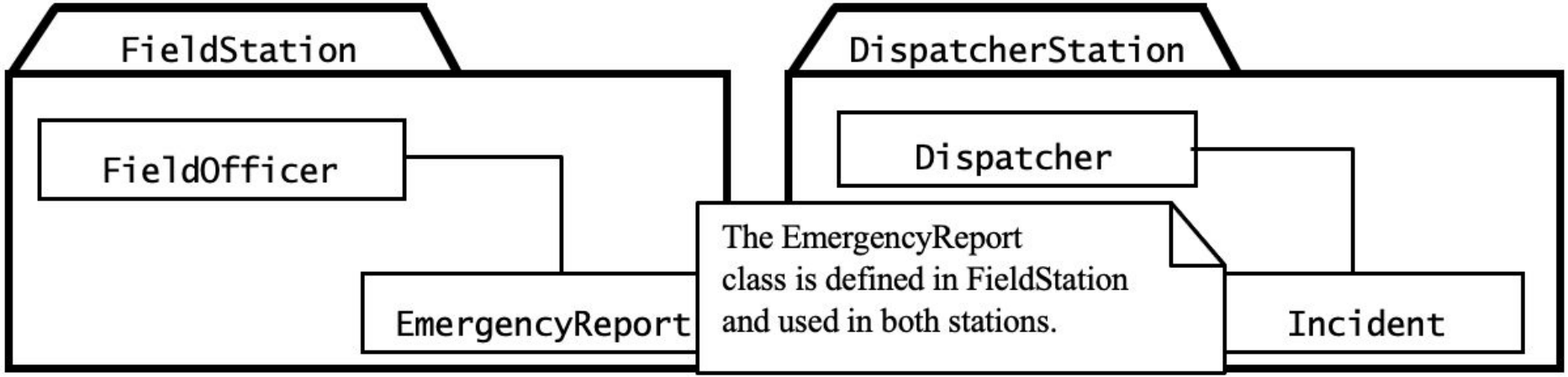
Ancora sul package

- I package possono essere usati per trattare la complessità alla stregua dell'organizzazione di file e directory di un generico utente
  - La differenza è che i package non sono necessariamente gerarchici:
    - la stessa classe può trovarsi in più di un package
  - Per ridurre le inconsistenze, le classi appartengono ad un solo package e gli altri package fanno riferimento ad essa

Note

- Una **nota** è un commento allegato al diagramma
- Sono usate dagli sviluppatori per allegare informazioni ai modelli e agli elementi del modello
- E’ un meccanismo per ricordare questioni rilevanti di un modello, per chiarire un punto complesso e per tener traccia di un punto da svolgere in futuro

Esempio di nota:
le note possono essere allegate a un elemento del diagramma specifico



Estensione dei diagrammi

- L’obiettivo di UML consiste nel fornire un insieme di notazioni per modellare un’ampia classe di sistemi software
- Tuttavia, un insieme prefissato di notazioni non potrebbe consentire di perseguire un tale scopo
  - È impossibile anticipare tutte le necessità che si verificano in tutti i domini applicativi
- UML fornisce un numero di meccanismi che consentono di estendere il linguaggio

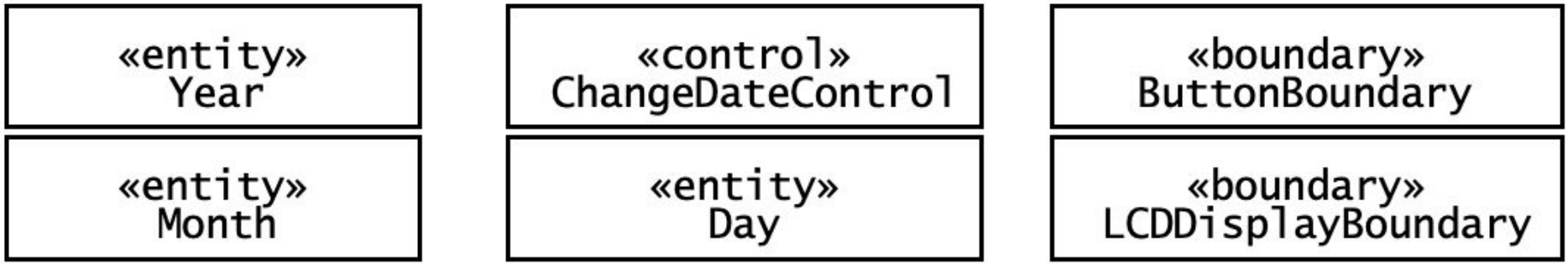
Stereotipi

- Uno **stereotipo** è un meccanismo di estensione che consente di classificare gli elementi del modello in UML
- E’ rappresentato da una stringa racchiusa tra parentesi angolari <<>> e allegate all’elemento a cui si applica, come una classe o un’associazione
- Formalmente, uno stereotipo corrisponde, semanticamente, a creare una nuova classe nel meta-modello UML (ovvero, il modello che rappresenta i costrutti di UML)

Stereotipi

- Esempio:
  - Durante l'analisi, classifichiamo gli oggetti in tre categorie: *entity*, *boundary* e *control*
  - Hanno la stessa struttura ma scopi diversi
- Il linguaggio base di UML include solo un tipo di oggetto. Per rappresentare le tre categorie usiamo gli stereotipi `<<entity>>`, `<<boundary>>` e `<<control>>`

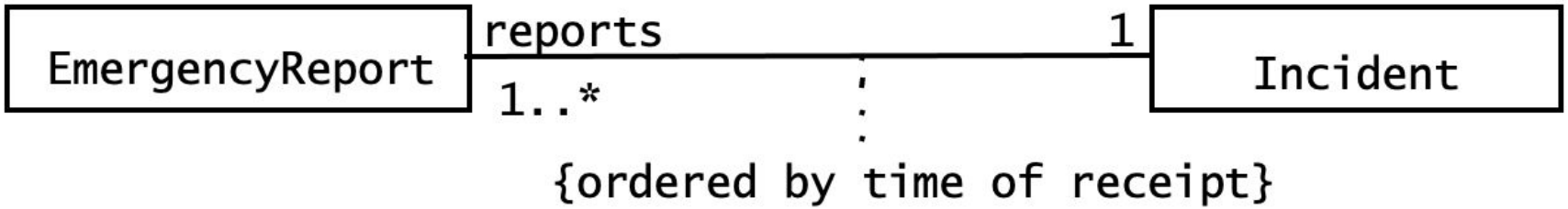
Esempio di stereotipi



Vincoli

- Un vincolo è una regola che viene allegata ad un elemento restringendone la semantica
- Consente di rappresentare fenomeni che non potrebbero essere espressi altrimenti in UML
- Esempio
  - Un incidente (*Incident*) può essere associato con uno o più *EmergencyReport* prodotti dal luogo dell'evento
  - I *Dispatcher* devono essere in grado di esaminare i rapporti in ordine cronologico
  - L'ordinamento cronologico degli *EmergencyReport a Incident* è espresso dal vincolo {*ordinato in base all'ora di ricezione*}
  - I vincoli possono essere espressi in linguaggio naturale o mediante OCL (Object Constraint Language)

Esempio di vincolo



Strumenti per UML

Programma	Descrizione	http
DIA	Gnome Visio-like diagram tool with aUML template	http://www.gnome.org/projects/dia/
Poseidon for UML community	UML diagrams, code generation for Java	http://www.gentleware.com/index.php?id=ce
ArgoUML	Open-source project, written in Java	http://argouml.tigris.org/
Umbrello	KDE-based open source written in C++	http://uml.sourceforge.net/index.php
Visual Paradigm for UML community	Free version with restrictions	http://www.visual-paradigm.com/product/vpuml/communityedition.jsp

https://www.guru99.com/best-uml-tools.html

Attività di sviluppo e relativi prodotti