DIAGRAMMI DI SEQUENZA

Francesco Poggi

fpoggi@cs.unibo.it

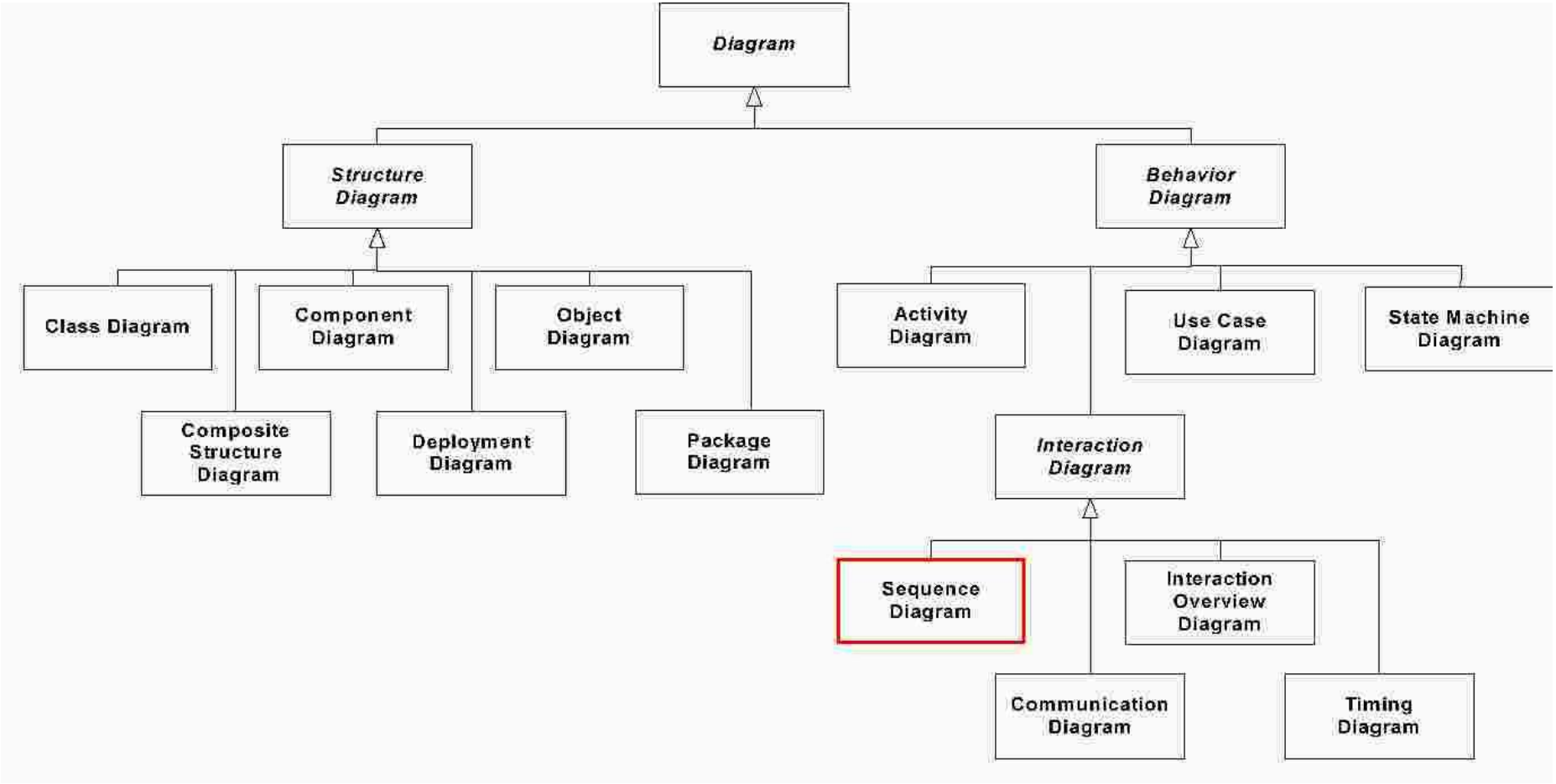
A.A. 2015-2016

Premessa

"As always, there is never a "correct" solution to any modelling problem. It's more that some models are more precise, and more informative, than others. You may find your solutions differ slightly from the ones below, but these solutions demonstrate good practice in model building."

[*dal corso "The Models Are the Code - Executable UML", prof. Paul Krause, University of Surrey*]

I diagrammi di interazione



Diagrammi di sequenza

- Sono diagrammi di comportamento che modellano le interazioni tra varie entità di un sistema.
- Visualizzano lo **scambio di messaggi** tra entità nel **tempo**.
- Il loro scopo è mostrare **come** un certo comportamento viene realizzato dalla collaborazione delle entità in gioco.
- La parola chiave è **realizzare**: questi diagrammi mostrano la realizzazione di un comportamento offerto.
- In altri termini: il comportamento del sistema da una **prospettiva interna**
- Come gli altri diagrammi, possono avere **diversi livelli di astrazione**.

Come procedere

Un diagramma di interazione necessita di due cose:

- Un comportamento da realizzare tratto da un classificatore (classificatore di contesto), ad esempio...
  - un caso d'uso
  - un'operazione di classe
- Una serie di elementi che realizzano il comportamento, ad esempio...
  - attori
  - istanze di classe

Questi ingredienti provengono da diagrammi creati precedentemente.

I diagrammi

Ci sono 4 tipi di diagrammi di interazione, noi ci concentreremo sul primo:

- **Diagramma di sequenza:** adatto a mostrare la sequenza temporale degli avvenimenti per ogni entità nel diagramma.
- **Diagramma di comunicazione:** adatto a mostrare le relazioni strutturali e i collegamenti tra le entità e i messaggi che vi transitano.
- **Diagramma di interazione generale:** adatto a mostrare la costruzione di interazioni complesse a partire da interazioni più semplici.
- **Diagramma di temporizzazione:** adatto a mostrare l'evoluzione dell'interazione in tempo reale.

Elementi dei diagrammi d'interazione

Nei diagrammi di interazione generalmente non compaiono direttamente classificatori come le classi: al loro posto ci sono

- **Istanze** di classificatori (oggetti, istanze di attori, etc.)
- **Linee di vita** (lifeline) di classificatori (esprimono ruoli, non specifici oggetti)

La differenza tra istanze e linee di vita è sottile, ma in generale è preferibile usare le linee di vita.

Questi diagrammi includono anche (anzi, ne sono gli elementi principali) i **messaggi**, ossia i segnali che si scambiano le istanze di classificatori e/o le linee di vita

Istanze vs linee di vita

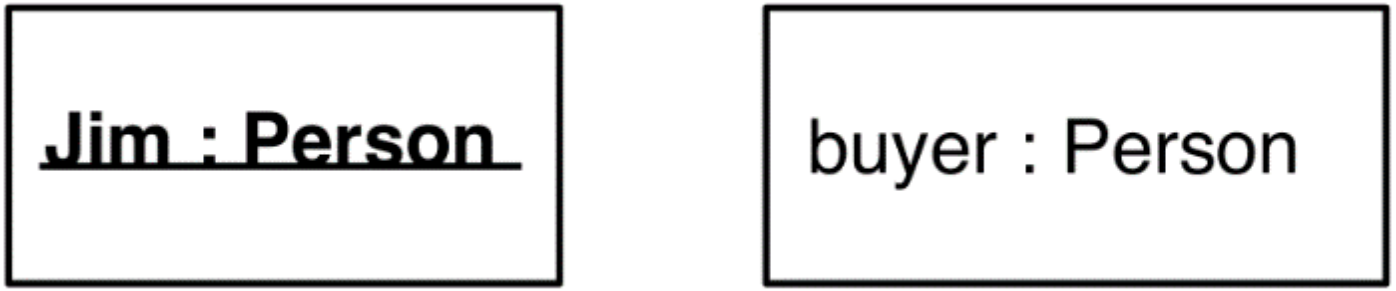
Un'**istanza**:

- rappresenta uno specifico oggetto, e solo quello;

Una **linea di vita**:

- è più generale;
- rappresenta un'istanza arbitraria di un classificatore;
- fornisce modi per specificare come quest'istanza viene scelta;
- esprime il ruolo giocato da un'istanza senza preoccuparsi della sua identità.

Istanze e linee di vita



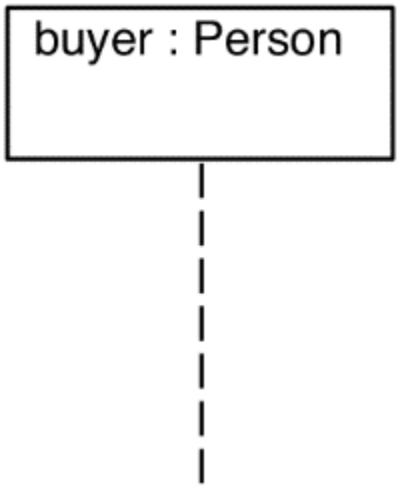
- Entrambe usano la notazione del loro classificatore, ma le linee di vita non sono sottolineate.
- La notazione completa per una linea di vita è:

  nome [selettore] : tipo

- Il selettore è un'espressione booleana opzionale che permette di scegliere un particolare rappresentante del classificatore, ad es. [id="1234"].
- Senza selettore la linea di vita rappresenta un'arbitraria istanza.

I diagrammi di sequenza

- Evidenziano l'ordine temporale delle invocazioni dei metodi
- Una linea verticale tratteggiata indica una sequenza temporale.
- Gli eventi hanno luogo nel loro ordine sulla linea; le distanze sono irrilevanti.
- Più linee di vita interagiscono per realizzare il comportamento offerto, scambiandosi messaggi



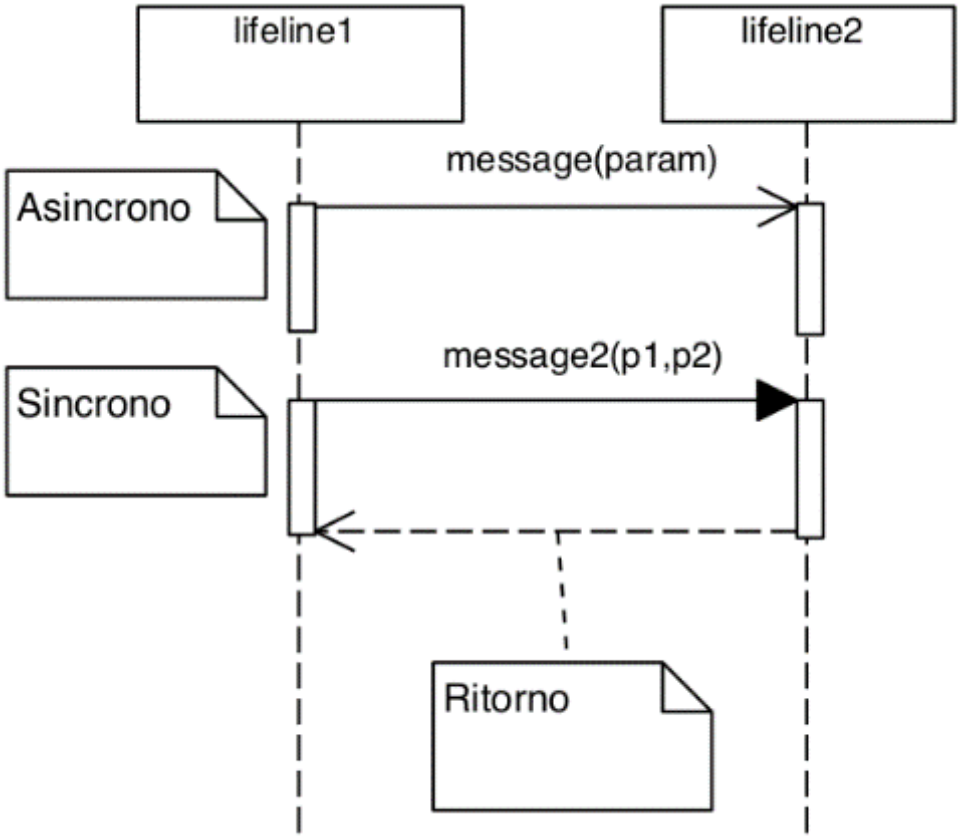
Messaggi

Rappresentano comunicazioni tra linee di vita: segnali, chiamate di operazioni, creazione e distruzione di oggetti.

Sette tipi definiti:

- chiamata sincrona
- chiamata asincrona
- ritorno da chiamata
- creazione
- distruzione
- messaggio trovato
- messaggio perso

Tipi di messaggi: chiamata



- Comunicazione sincrona vs. comunicazione asincrona.
- I rettangoli di attivazione (indicano un focus di controllo) sono opzionali.

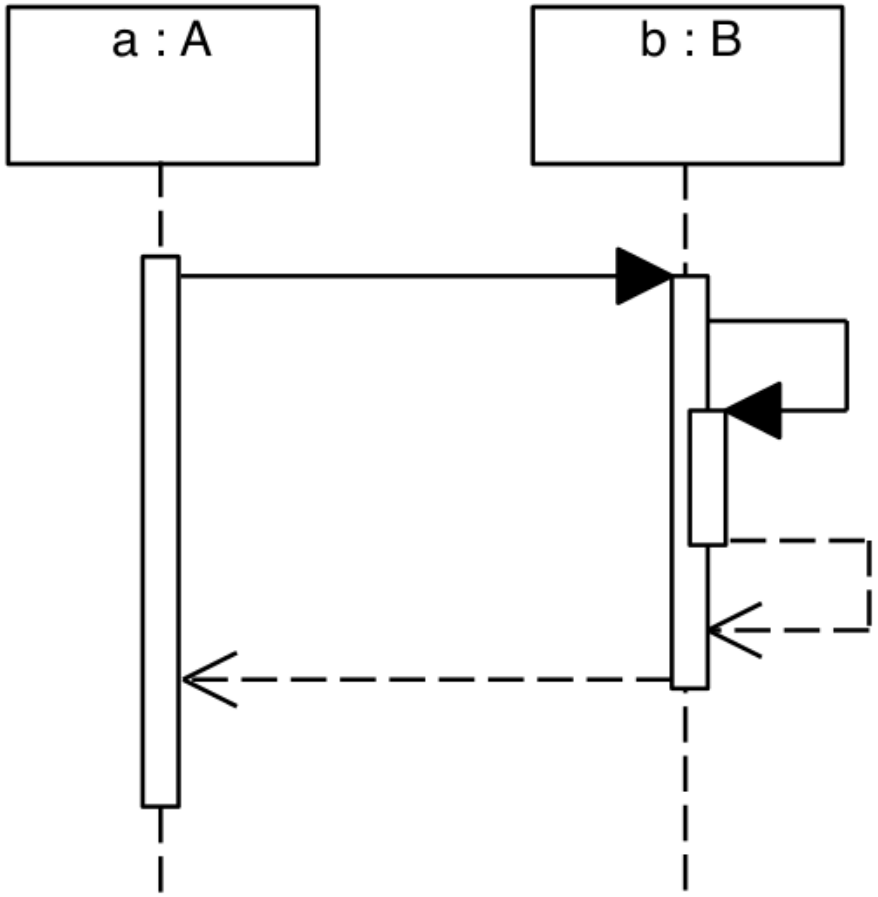
Tipi di messaggi: chiamata

- I messaggi **sincroni** fanno bloccare chi li manda fino al messaggio di ritorno del destinatario.
- Nei messaggi **asincroni** il mittente non si aspetta un messaggio di ritorno e prosegue immediatamente.
- I **messaggi di ritorno** sono **opzionali** e in generale inclusi solo quando non ostacolano la leggibilità del diagramma oppure per segnalare valori di ritorno.
- La distinzione tra messaggi sincroni e asincroni in genere emerge in fase di progettazione.
- In fase di analisi, conviene indicare tutti i messaggi come sincroni (è il caso più vincolante).

Sintassi dei messaggi

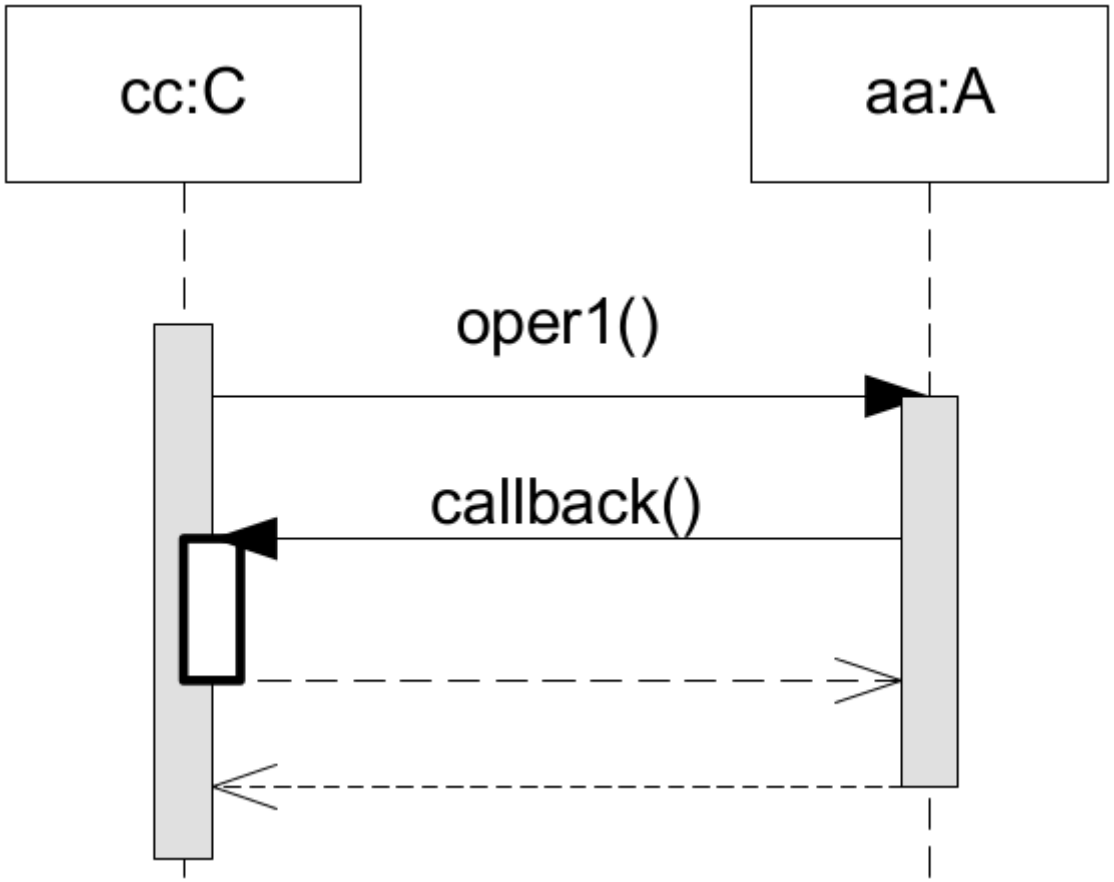
- In fase di analisi generalmente basta includere il nome del messaggio.
- Se si desidera maggiore dettaglio, si include una lista di **parametri** tra parentesi.
- Si possono anche usare **guardie** per verificare condizioni
- Si possono indicare i parametri formali, quelli attuali, o entrambi (in quest'ordine: getArea(shape=g)).

Attivazioni annidate



- Le linee di vita possono mandare messaggi a se stesse.
- Si tratta di un caso frequente (es. invocazione di operazioni private).

Attivazioni annidate



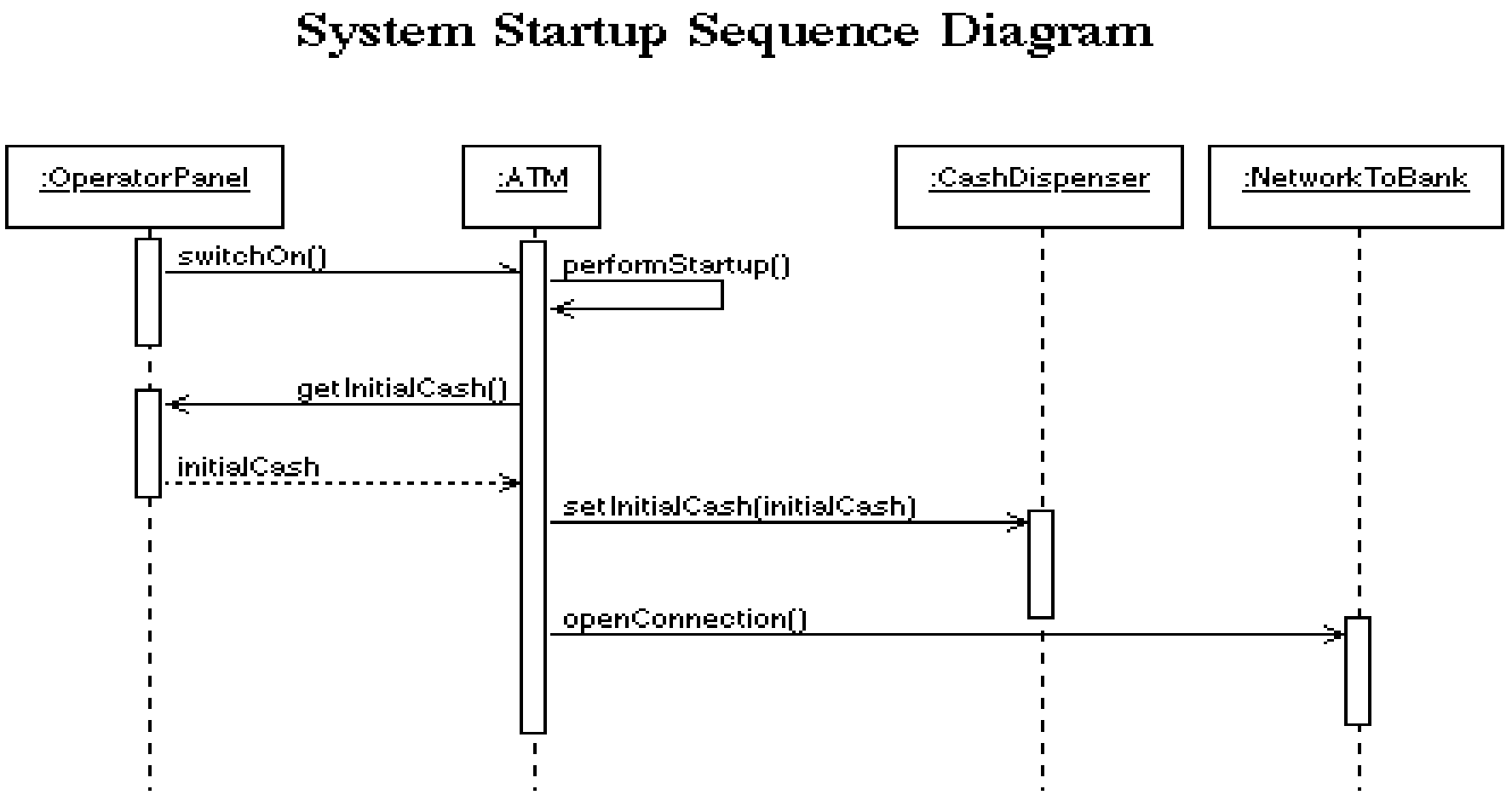
Le attivazioni possono generare altre attivazioni sulla linea vita chiamante (es. callback).

Esercizio

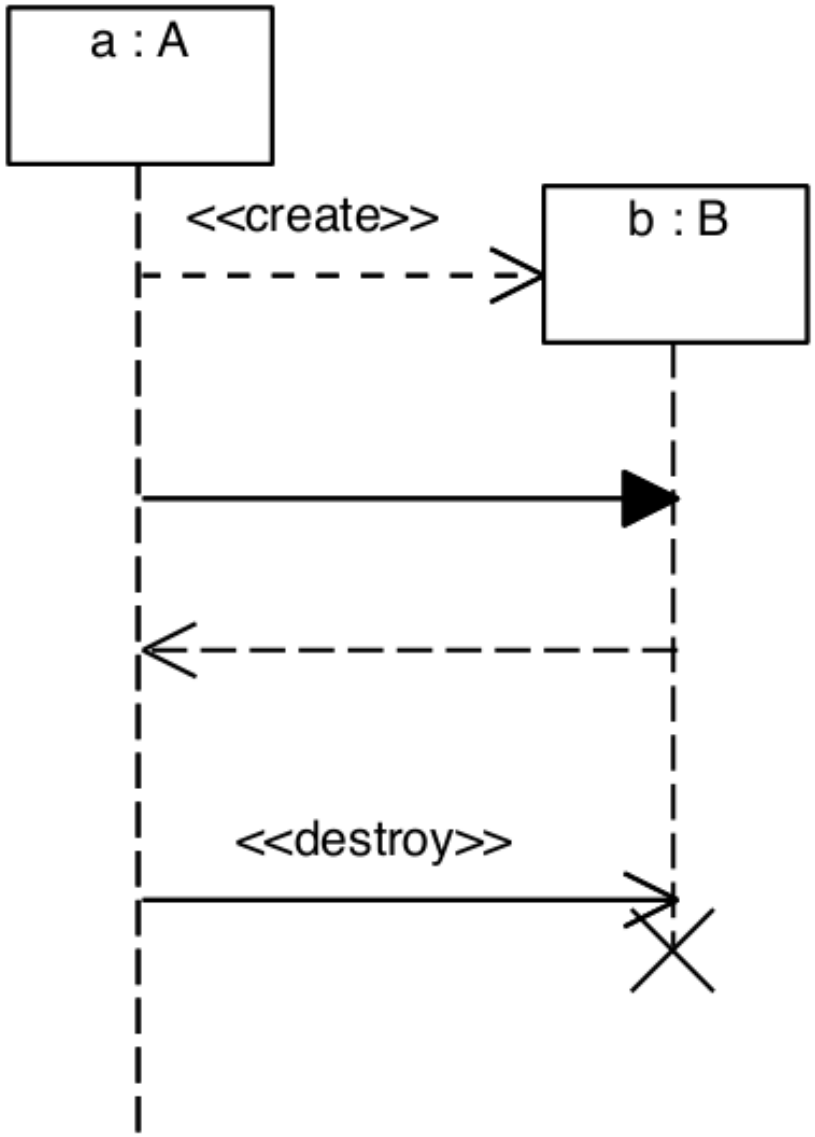
Si realizzi un diagramma di sequenza per modellare il seguente dominio:

- Il Sistema si attiva quando l'operatore posiziona l'interruttore sullo stato "acceso". All'operatore viene richiesto di inserire la quantità di contante che si trova nella cassa del bancomat, e in seguito viene stabilita una connessione con la banca. A questo punto i clienti possono venire serviti.

Eseercizio: soluzione



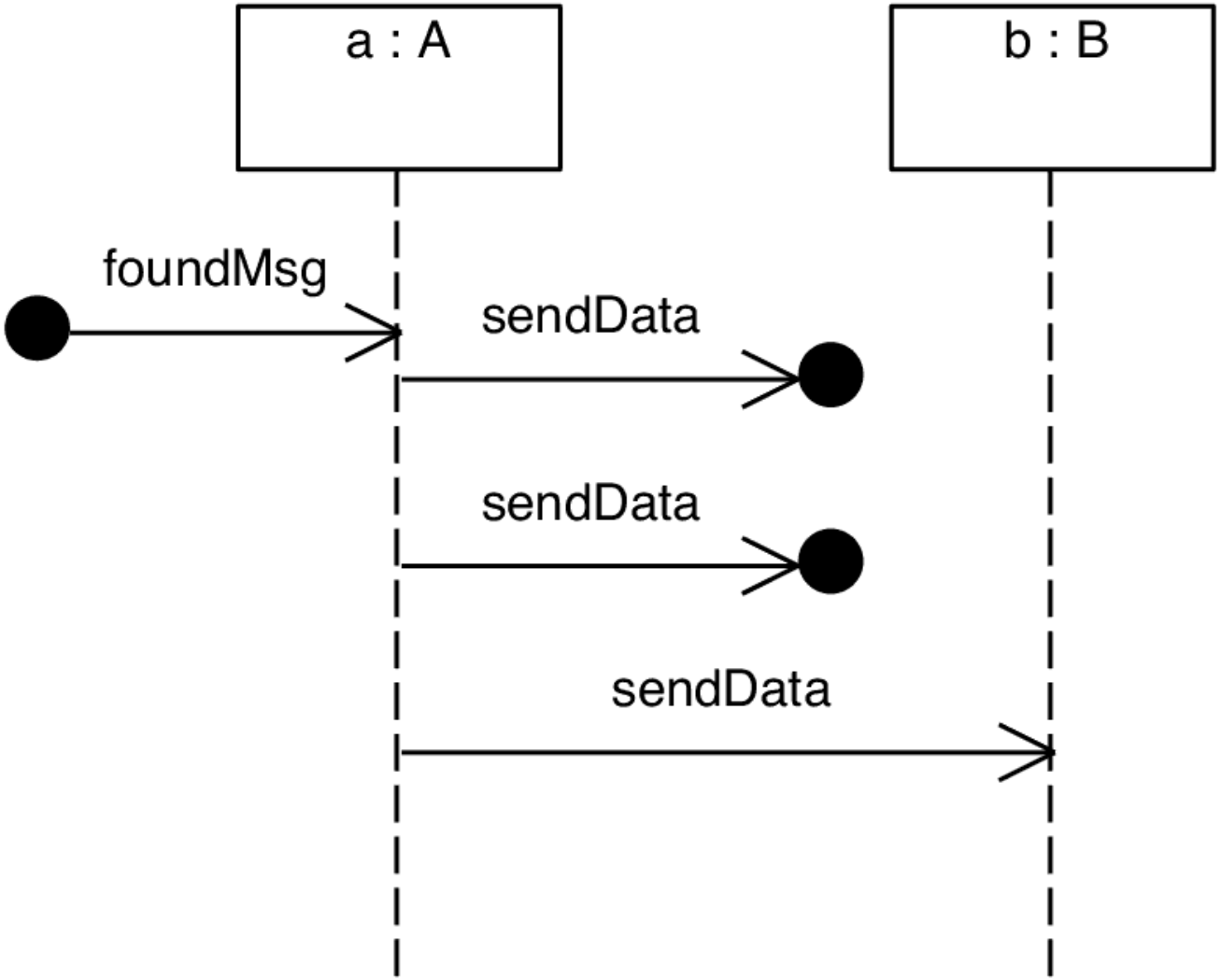
Tipi di messaggi: creazione/distruzione



Tipi di messaggi: creazione/distruzione

- Nel messaggio di creazione lo stereotipo può essere seguito dal nome di un'operazione e relativi parametri (costruttore).
- Questo si rende necessario nel caso si lavori con linguaggi di programmazione senza costruttori espliciti.
- Differenti linguaggi di programmazione OO gestiscono la distruzione in modi diversi (esplicita, garbage collector).
- In fase di analisi, basta immaginare che dopo la distruzione l'oggetto non sia più accessibile

Tipi di messaggi: trovati/persi (1)



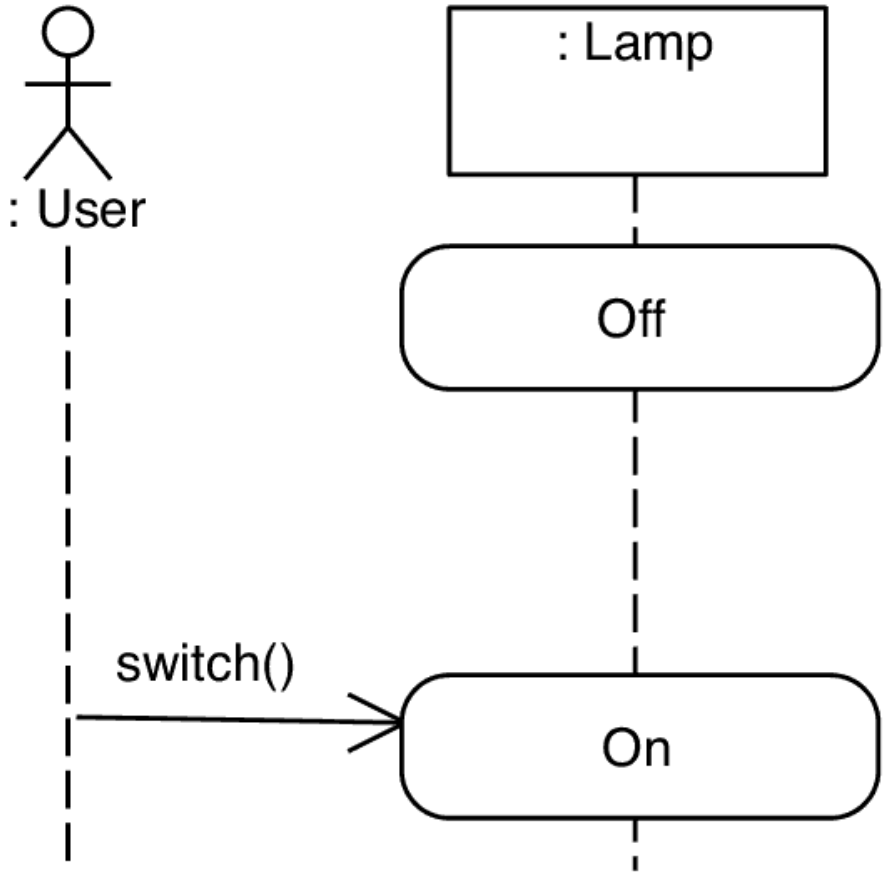
Tipi di messaggi: trovati/persi (2)

- Raramente usati in fase di analisi.
- I *messaggi trovati* indicano situazioni in cui il mittente è sconosciuto al momento della ricezione, proviene dall'esterno del diagramma o i dettagli della provenienza non interessano.
- I *messaggi persi* permettono di visualizzare il comportamento nel caso un messaggio non giunga a destinazione (situazione di errore).

Stati (1)

- Uno stato è definito come una condizione o situazione durante la vita di un oggetto in cui esso soddisfa una condizione, esegue un'attività o aspetta un evento.
- Ogni oggetto in UML può avere una macchina a stati associata che ne descrive gli stati possibili.
- I diagrammi di stato (state machine diagram) sono i più adatti a rappresentare gli stati e loro transizioni, ma può essere conveniente mostrare alcuni cambiamenti di stato nei diagrammi di sequenza.
- Generalmente, ci si limita agli stati principali all'oggetto, mettendo in risalto i messaggi che provocano un cambiamento di stato.

Stati (2)



- In UML, gli stati si rappresentano con rettangoli arrotondati.
- I cambiamenti di stato sono generalmente la conseguenza di uno o più messaggi.

Realizzare i casi d’uso

- Le primitive introdotte fin qui permettono di rappresentare sequenze di eventi senza **ramificazioni**; tuttavia, per realizzare i casi d’uso è necessario poter descrivere sequenze più complesse.
- Le sequenze degli eventi in un caso d’uso contengono parole chiave come if, then, else, for e while.
- I diagrammi di sequenza permettono di tradurre questi costrutti ed inserirli nel diagramma.
- Si ricorre ai **frammenti combinati**.

Frammenti combinati

Un frammento combinato è una sottoarea di un diagramma di sequenza che racchiude una parte dell'interazione e le modalità della sua esecuzione.

Gli elementi di un frammento combinato sono:

- un **operatore**, che specifica come il frammento viene eseguito
- uno o più **operandi**, sottoaree del diagramma di sequenza che possono essere eseguite
- zero o più **condizioni di guardia**, espressioni che determinano quali operandi sono eseguiti.

Frammenti combinati

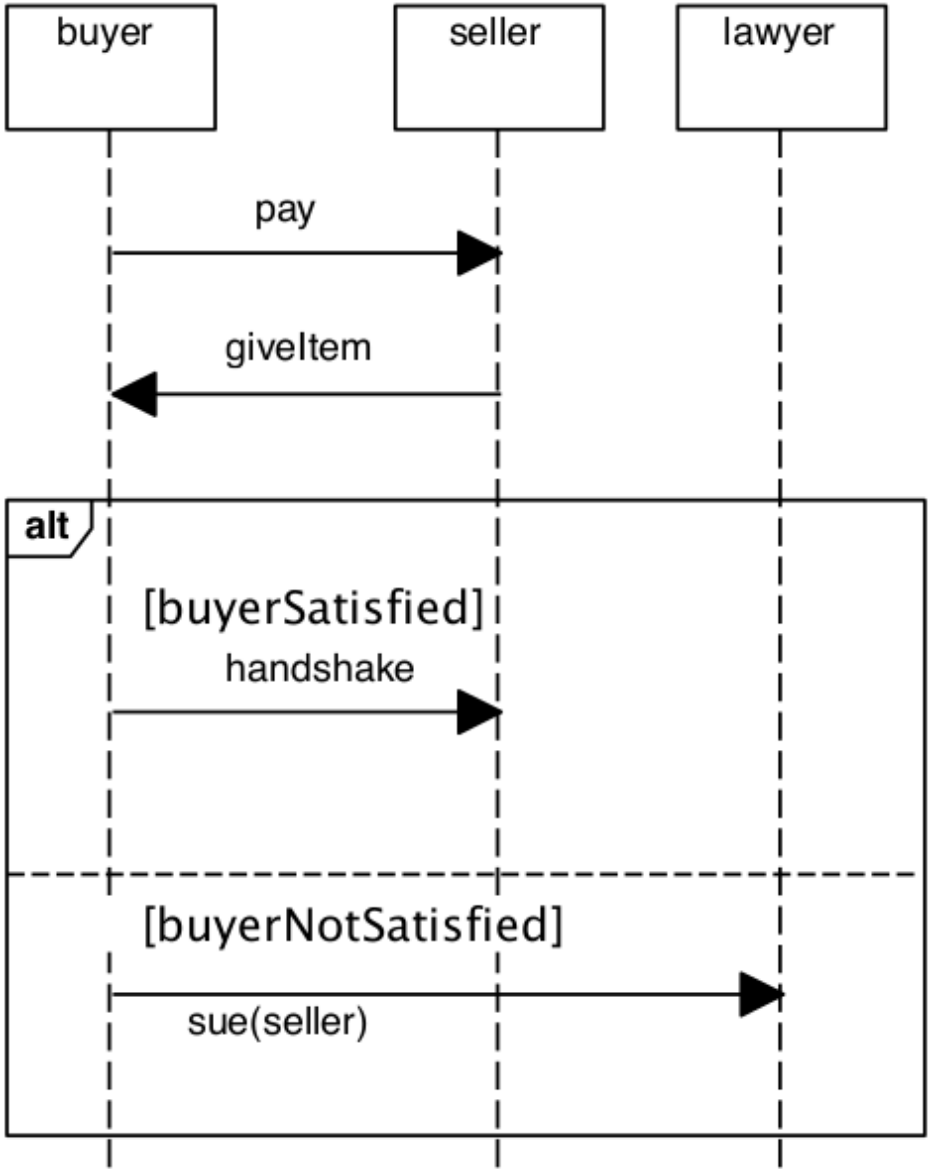
La **sintassi** UML per un frammento combinato è la seguente:

- un rettangolo con il nome dell'**operatore** in alto a sinistra racchiuso nel simbolo di diagramma (rettangolo con angolo tagliato)
- gli **operandi** seguono come strisce orizzontali in sequenza, separati da linee tratteggiate
- la **guardia**, se presente, compare dopo l'operatore oppure nella parte alta di un operando tra parentesi quadre

Il rettangolo del frammento combinato si estende orizzontalmente per includere le linee di vita interessate dall'interazione.

La guardia può includere qualunque tipo di condizione booleana

Frammenti combinati: esempio



Operatori

- L'operatore determina la semantica dell'esecuzione del frammento.
- UML specifica parecchi tipi di operatori, ma solo alcuni sono usati frequentemente (quelli che corrispondono alle primitive di controllo del flusso definite dai linguaggi di programmazione).
- I più usati:
  - **opt** corrisponde a if/then
  - **alt** corrisponde a case/select
  - **loop** corrisponde a for/while
  - **break** corrisponde a break

Operatori: opt e alt

- **opt** accetta solo un operando, che viene eseguito se e solo se la guardia è valutata true.
- **alt** accetta un qualunque numero di operandi e ne esegue al più uno.
  - esamina la lista a partire dal primo operando, valuta le guardie ed esegue solo il primo operando la cui guardia è vera
  - l'operando opzionale [else] è eseguito se nessuna delle guardie è vera
  - le condizioni di guardia devono essere *mutualmente esclusive*; al massimo una può essere vera nello stesso istante, altrimenti il modello non è corretto

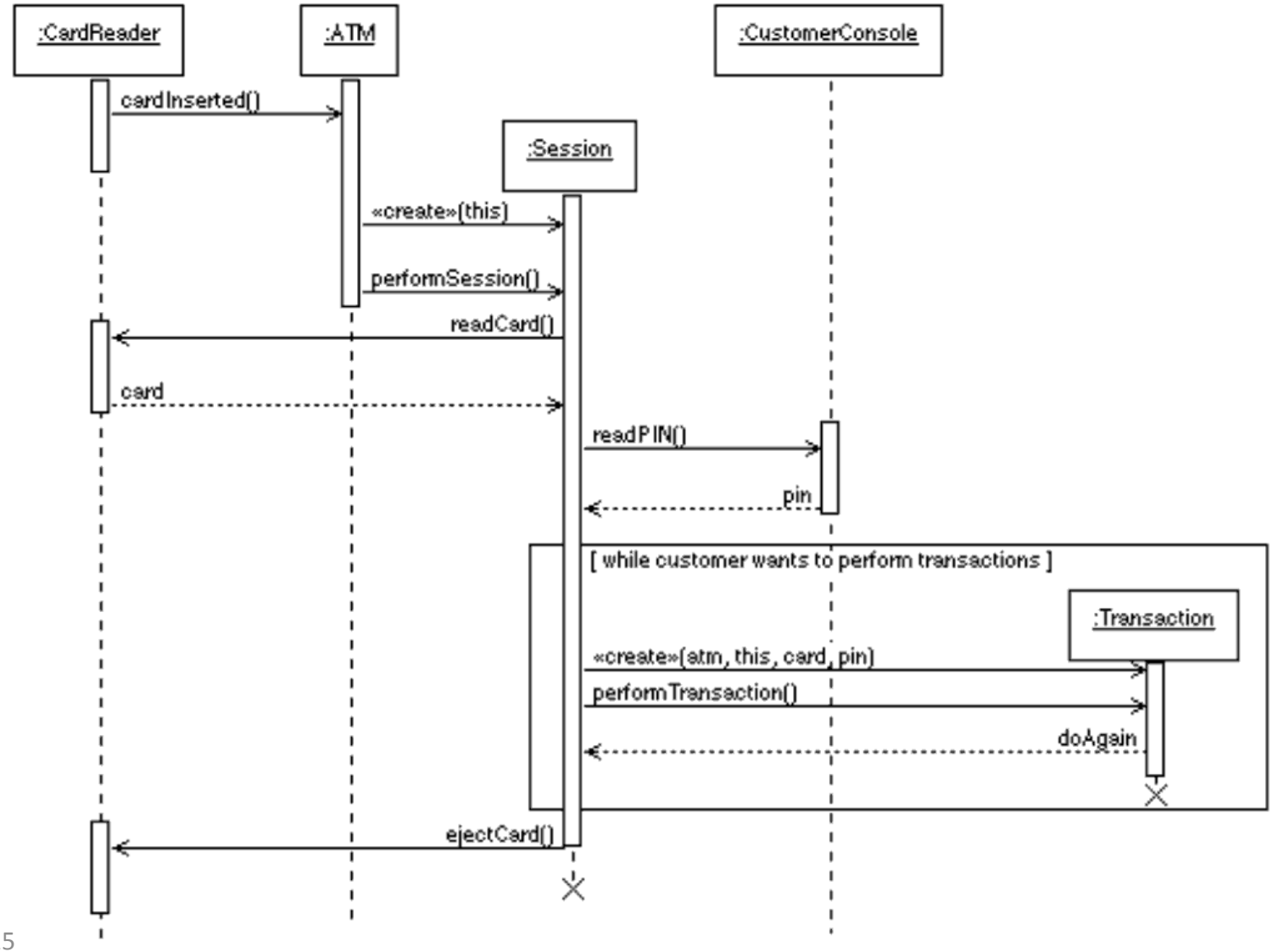
Operatori: loop

- Un solo operando, sintassi speciale:

  `loop min,max [condizione]`

- Esegue l'operando *min* volte, e poi al massimo altre (*max* - *min*) volte mentre la condizione è vera.
- Si possono omettere *min* e *max*.
- La condizione è molto flessibile (può essere espressa in linguaggio naturale), e permette di ciclare su un insieme di oggetti (es. condizione `[forEach oggetto in lista]`).

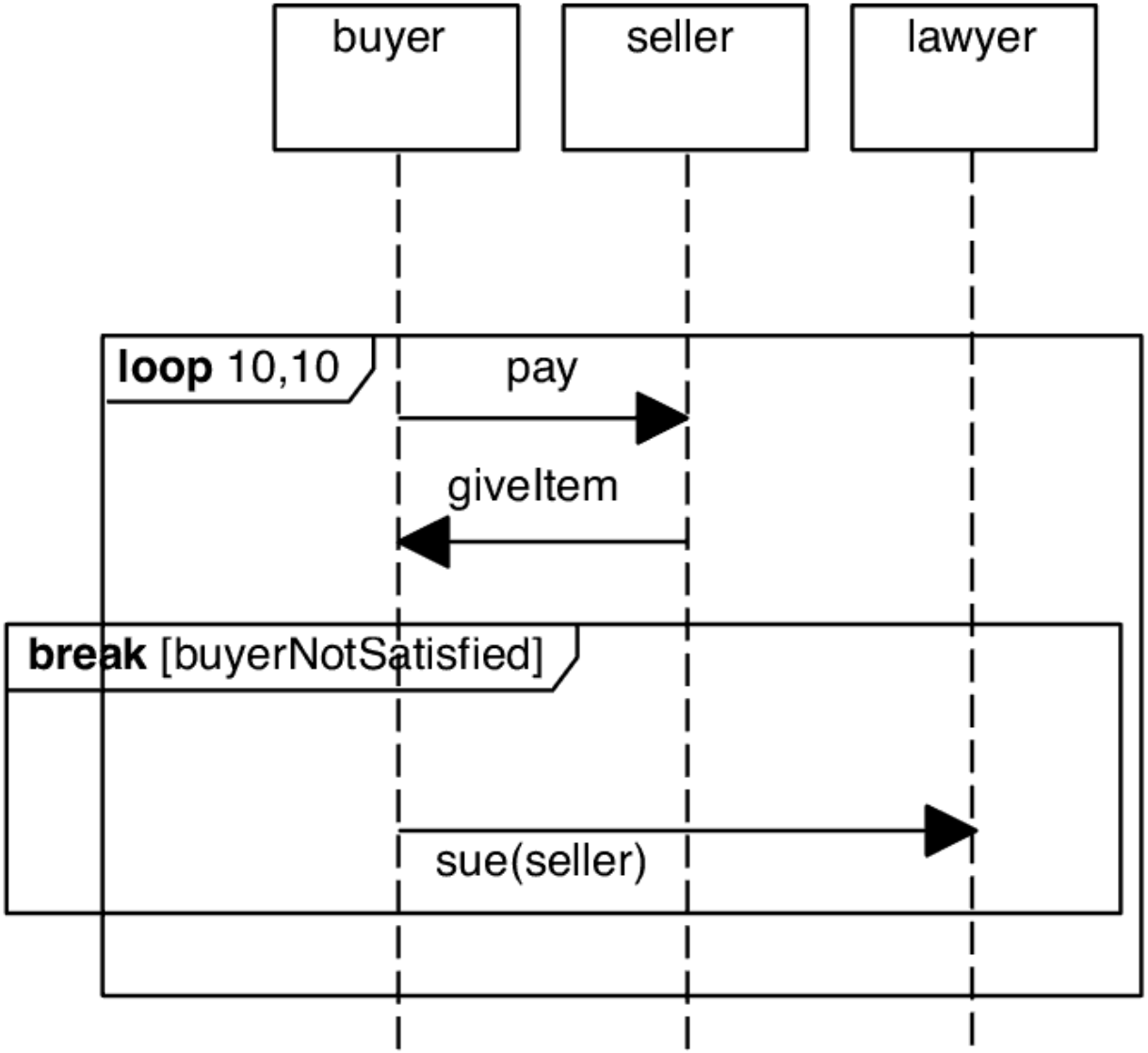
Esempio loop



Operatori: break

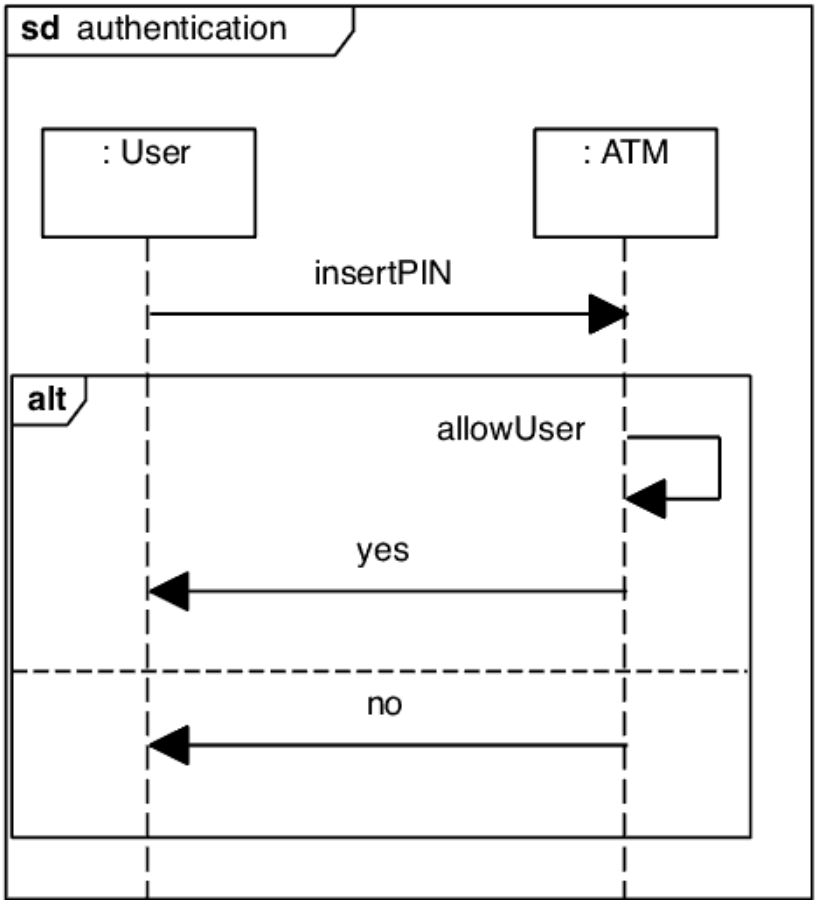
- Operando singolo, si usa per terminare l'esecuzione del frammento di interazione corrente (spesso si tratta di un operatore **loop**).
- Se **break** ha una condizione di guardia ed è vera, viene eseguito l'operando ma non il resto dell'interazione dopo il frammento **break**.
- Se la condizione di guardia esiste ed è falsa, non viene eseguito l'operando e si continua normalmente.
- Se non c'è una guardia, la scelta tra continuare e saltare è non-deterministica.
- Il frammento **break** deve essere globale rispetto a quello che interrompe: nella notazione UML, lo interseca ma si trova ad un livello di nesting superiore.

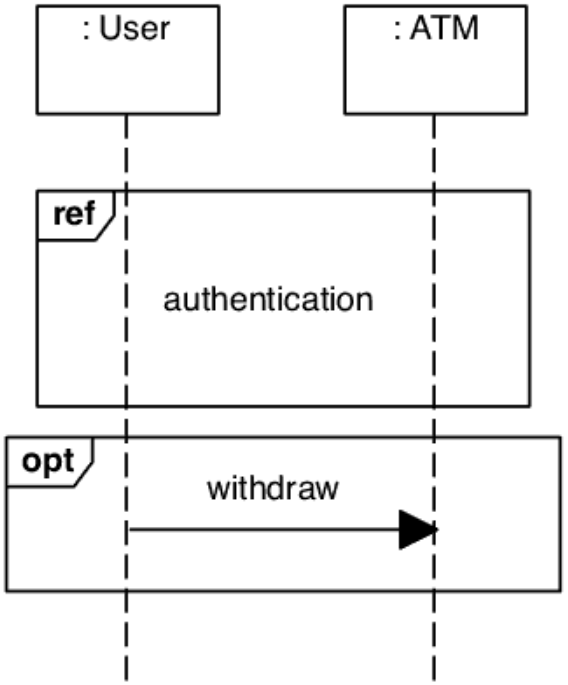
Esempio break



Operatori: Ref

- Per rendere i diagrammi di sequenza modulari si usa **ref**: l'operando contiene il nome di un'altra interazione che viene eseguita qui.

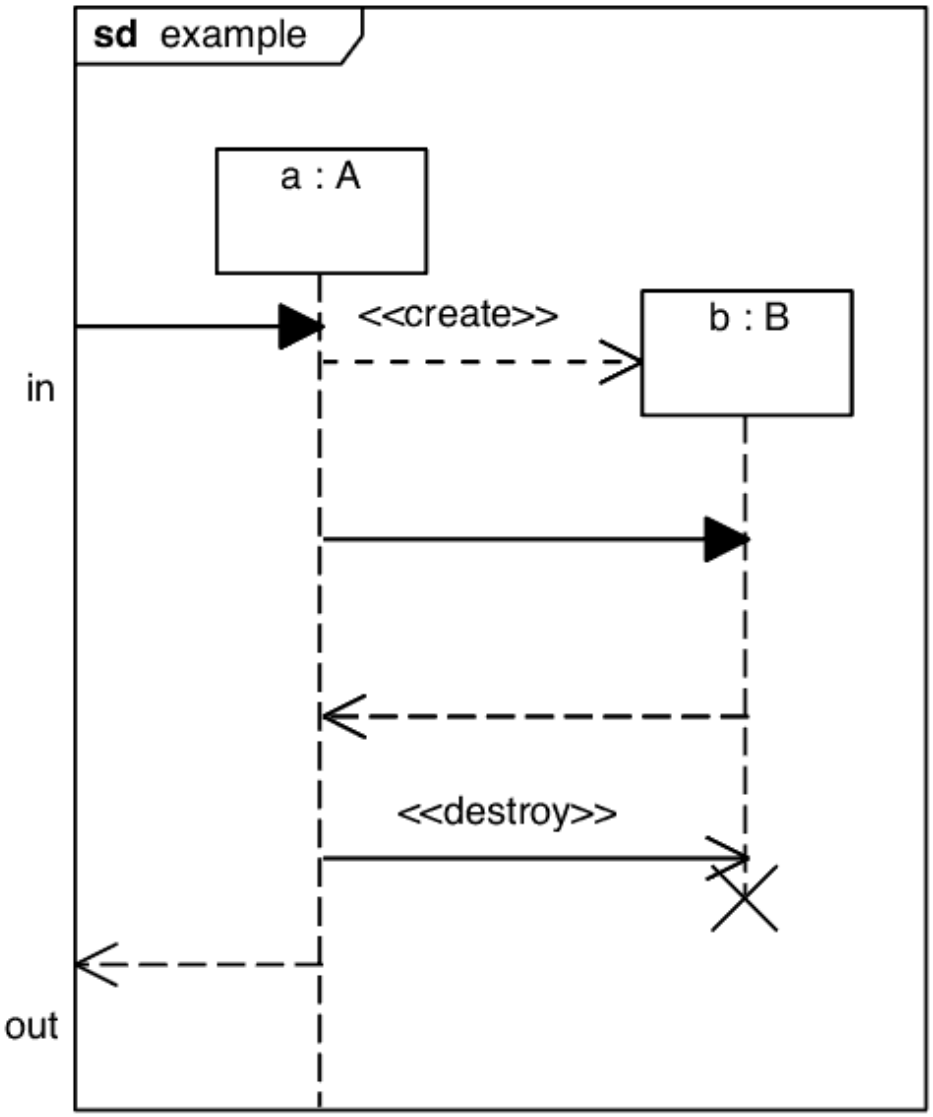




Operatori: Gate

- Si tratta di un qualunque punto sulla cornice esterna del diagramma, che può essere la fonte o la destinazione di una freccia.
- Usati per modellare la comunicazione delle linee di vita del diagramma con l'esterno.
- I gate possono avere un nome e comparire qualunque numero di volte nello stesso diagramma o in diagrammi diversi.
- Possono essere usati in combinazione con ref, forniscono l'interfaccia di collegamento tra l'interazione chiamante e quella chiamata (il frammento combinato può avere gli stessi gate a cui agganciare messaggi).

Gate: esempio

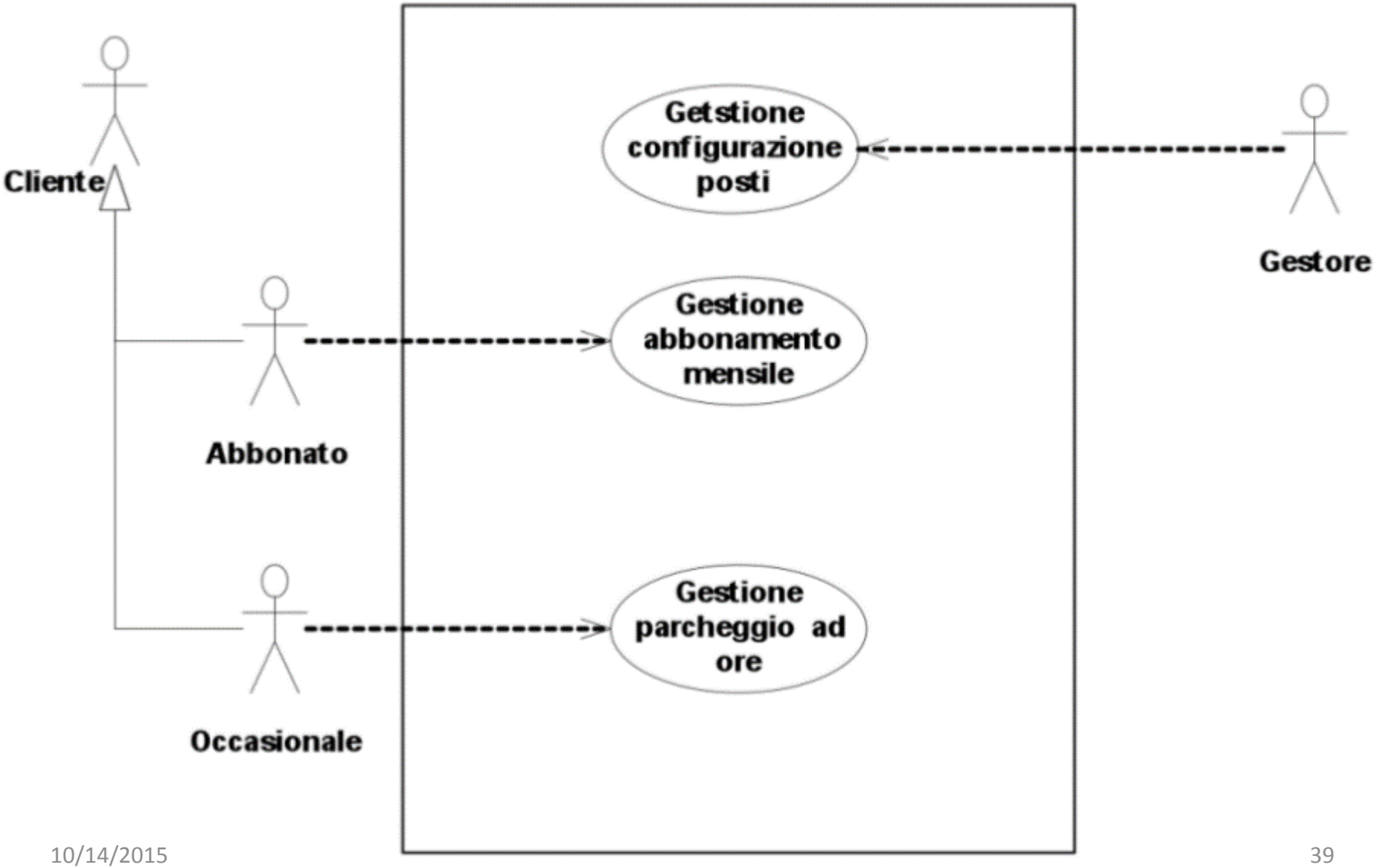


Esempio

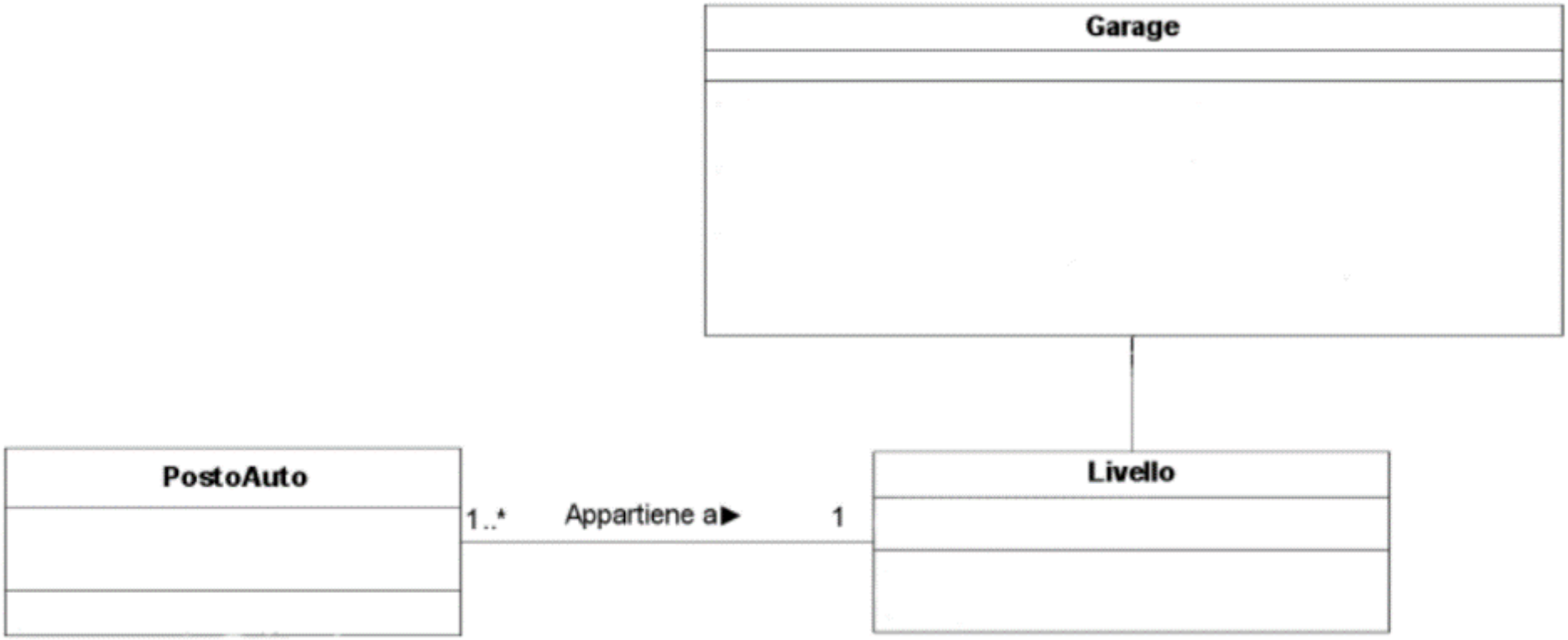
Un garage è composto di diversi livelli. Ogni livello ha un numero di posti disponibili. I posti sono di diversi tipi: auto normali, auto di dimensioni notevoli (van, ...), auto di lusso. Le auto GPL possono parcheggiare solo nel primo piano. È possibile affittare un posto macchina, se disponibile, su base mensile. Non possono essere affittati più del 50% dei posti di ciascuna categoria. I posti non affittati su base mensile sono utilizzati per parcheggi ad ore fino ad un massimo di otto ore. Nel caso si sforino le otto ore, viene applicata una penale al momento del ritiro dell'auto.

Gli utenti del sistema sono sia gli automobilisti che il gestore del sistema che fornisce le informazioni di configurazione (per esempio, il numero di posti di ciascuna categoria).

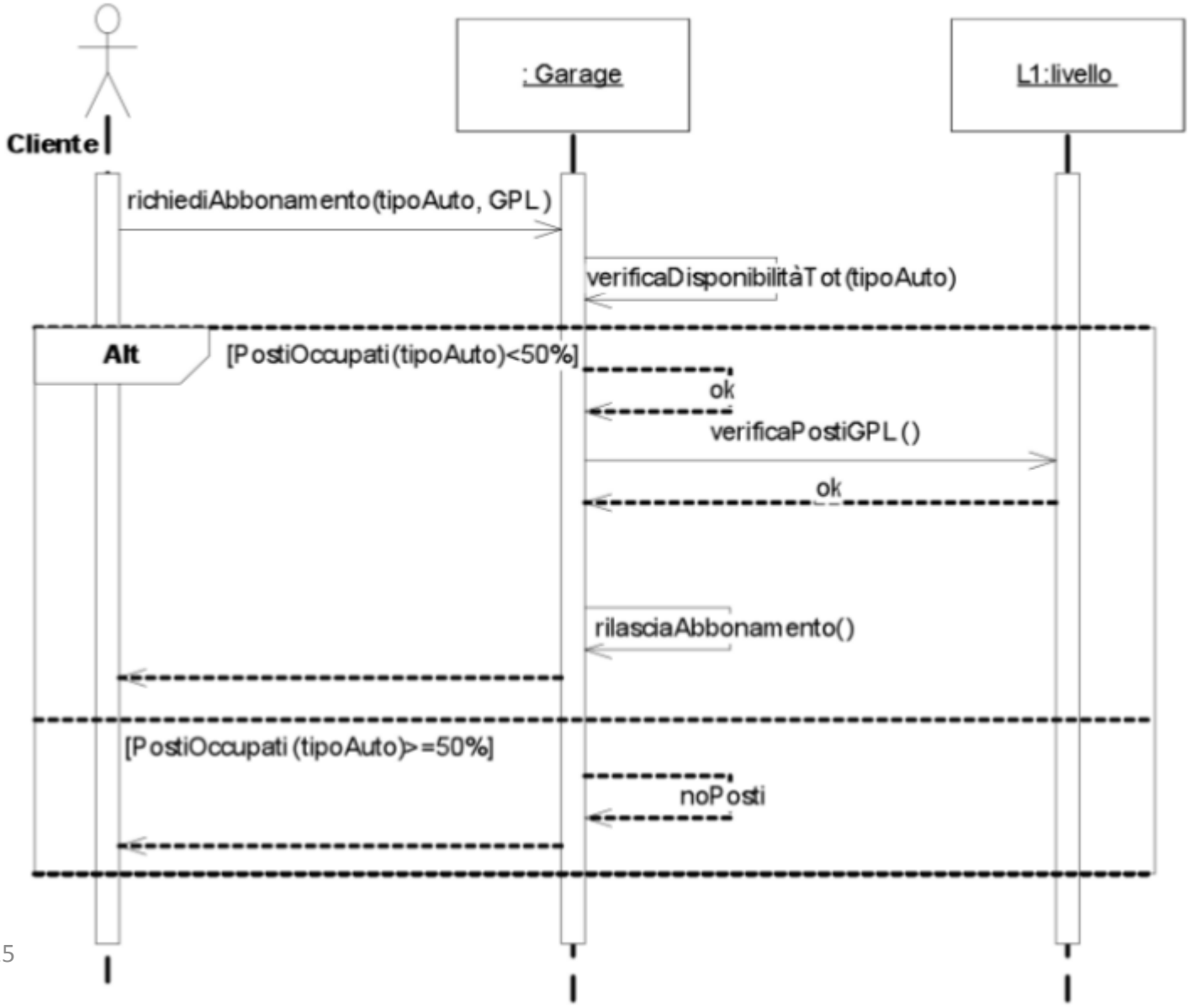
Diagramma dei casi d'uso



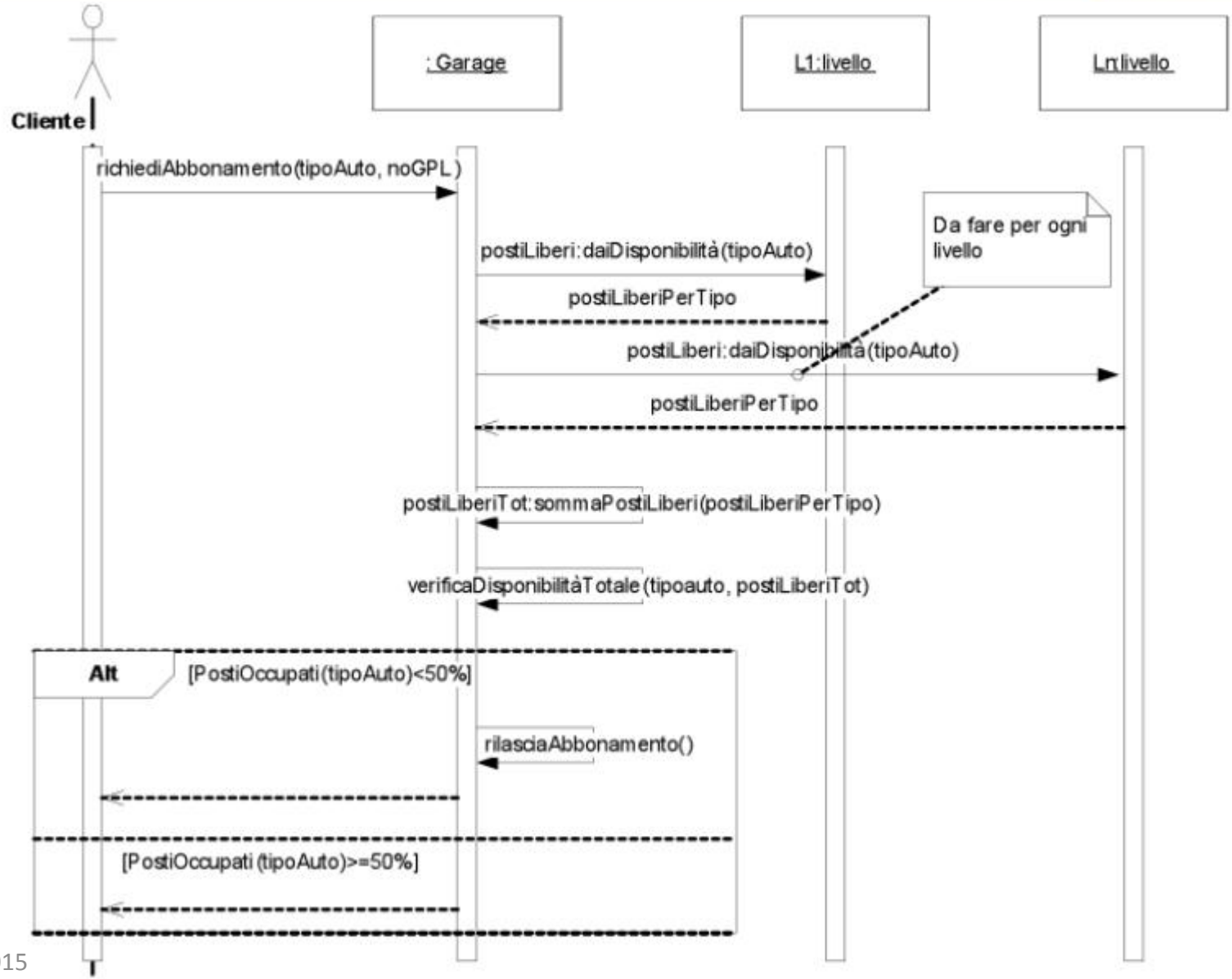
Classi di analisi iniziali



Abbonamento mensile GPL



Abbonamento mensile non GPL



Parcheggio a ore

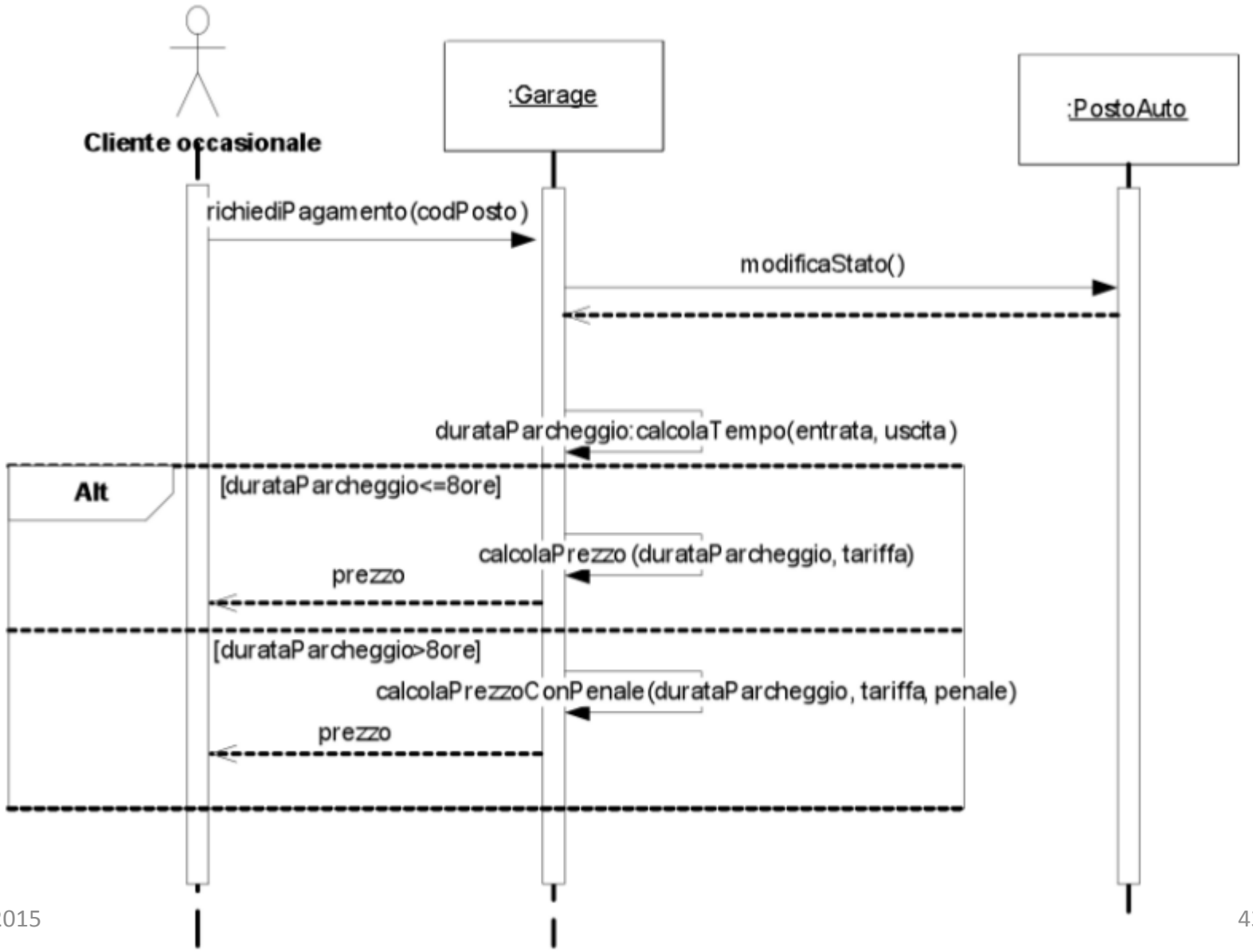
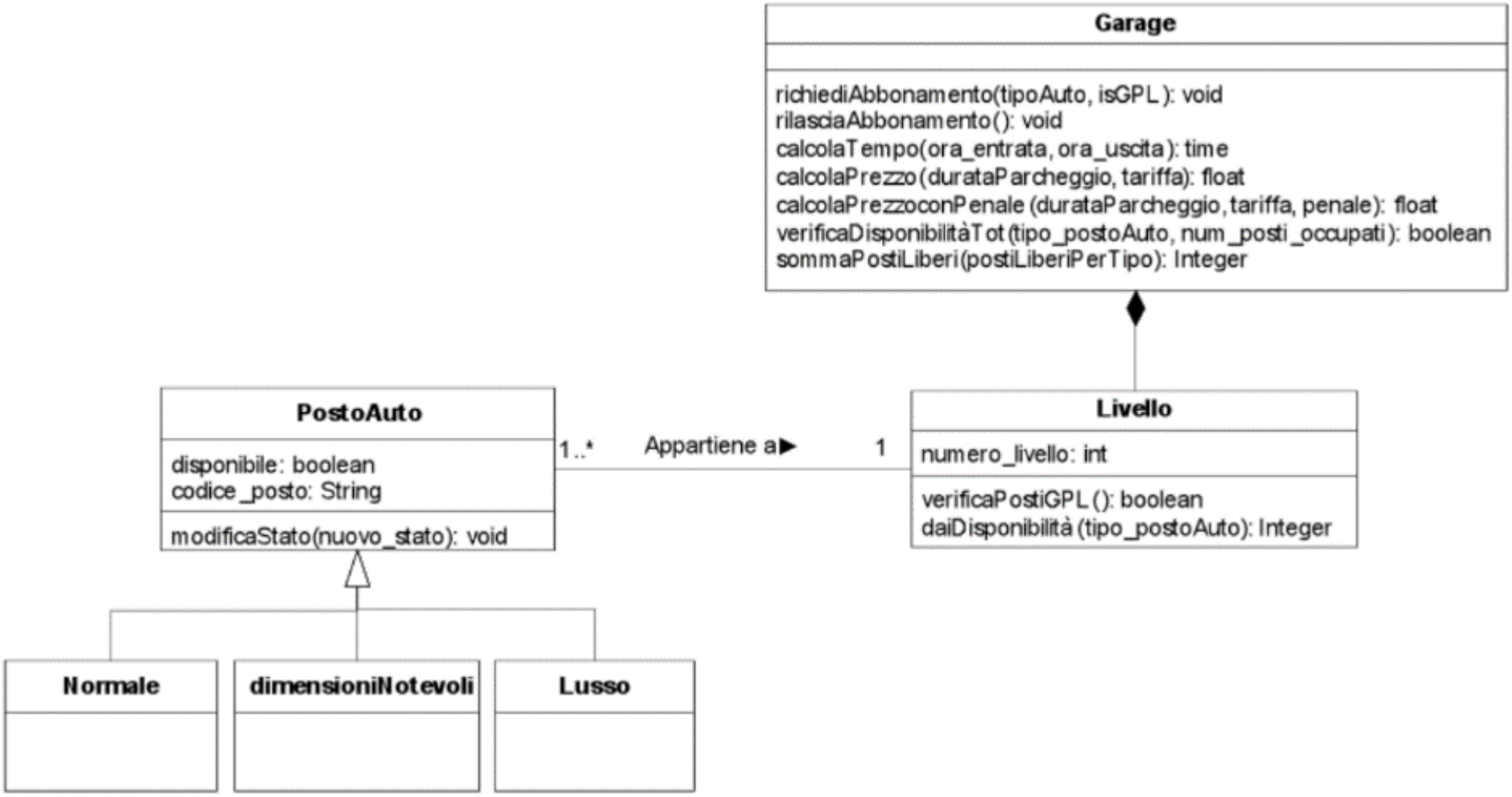


Diagramma di classe completo



Diagrammi di interazione: conclusione

- Si usano i diagrammi di interazione per mostrare come varie entità collaborino nel fornire un comportamento desiderato, ad esempio un caso d'uso.
- Possono essere usati sia in fase di analisi che in fase di progettazione.
- Aiutano a scoprire e correggere inconsistenze nel comportamento desiderato (ad esempio una specifica scorretta del caso d'uso).
- Spesso è utile modellare una situazione semplificata con un diagramma di comunicazione, per poi passare a uno di sequenza per i dettagli.