Universita’ degli Studi di Perugia
Dipartimento di Matematica e Informatica

Corso di Laurea in Informatica

Ingegneria del Software

Prof. Alfredo Milani

Sequence Diagrams – Diagrammi di Sequenza

Materiale note, grazie al contributo di: Alfredo Milani, Fabrizio Montecchiani, Carlo Ghezzi, Alessandro Riccardi, Paolo Mengoni et al.

Diagrammi di **Sequenza**

Sequence Diagrams

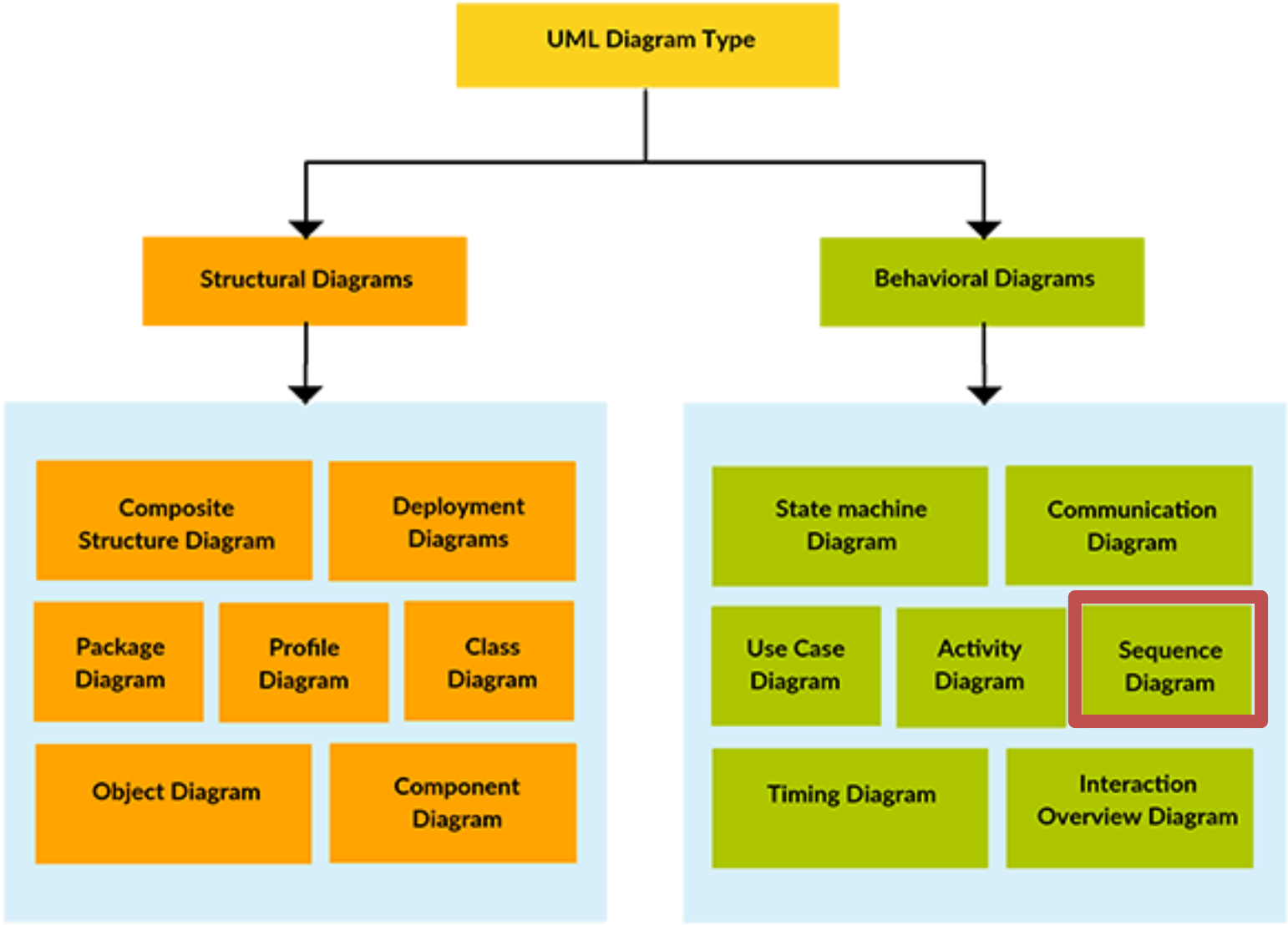
SOMMARIO

- Introduzione
- Partecipanti e messaggi
- Concetti avanzati
- Diagrammi di comunicazione/collaborazione

SOMMARIO

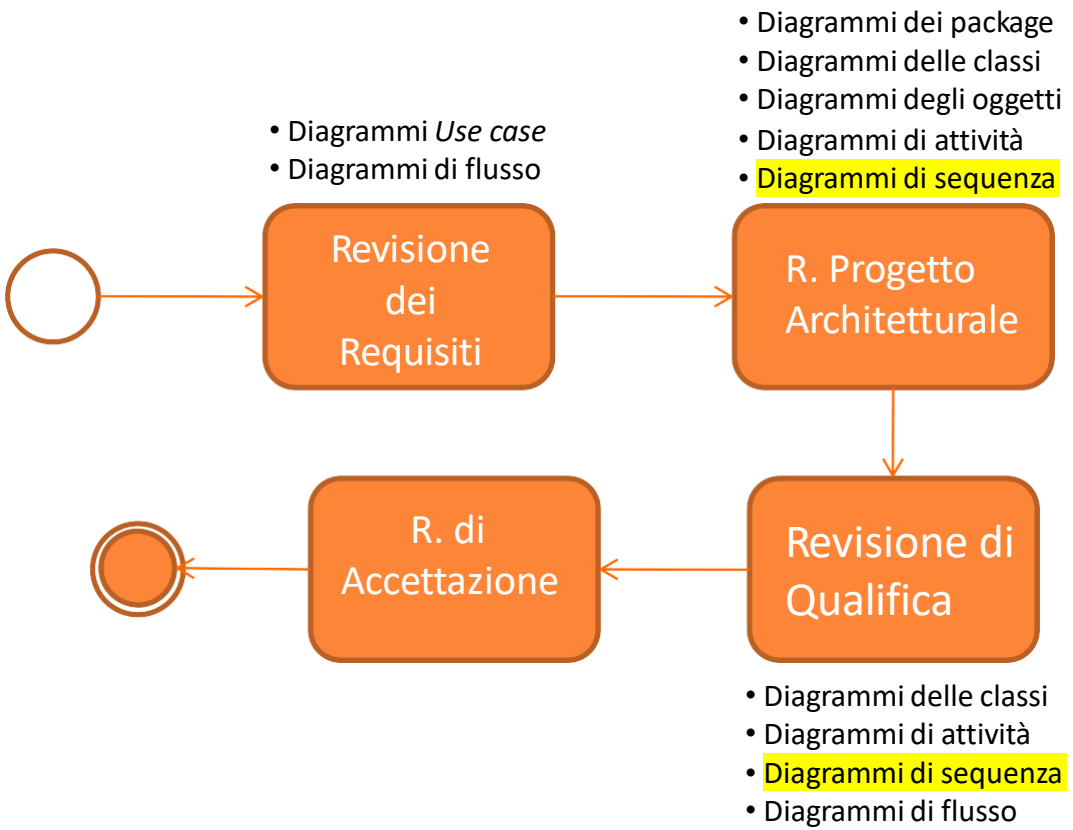
- Introduzione
- Partecipanti e messaggi
- Concetti avanzati
- Diagrammi di collaborazione/comunicazione

DIAGRAMMI DI SEQUENZA



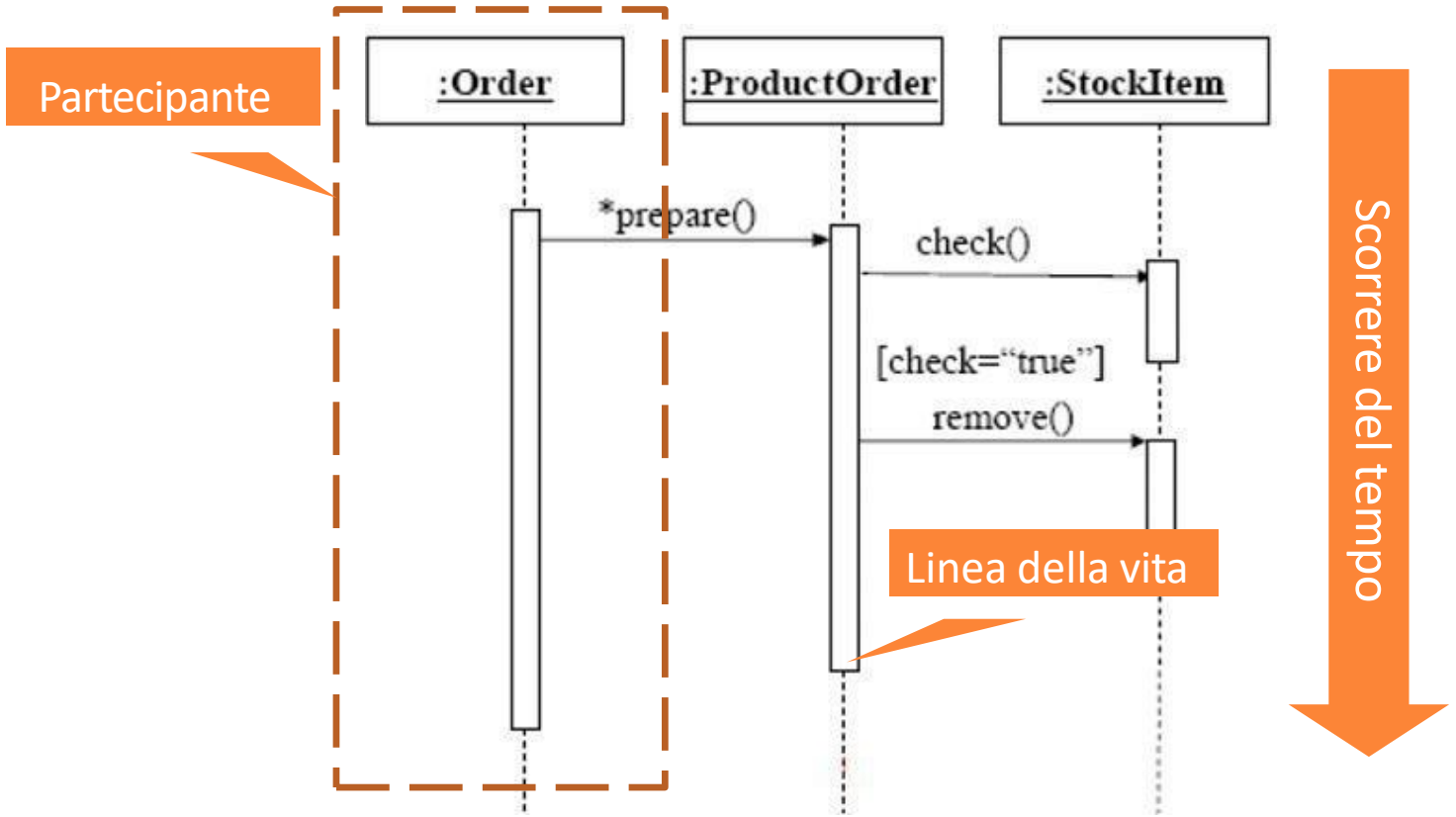
DIAGRAMMI DI SEQUENZA

- Ogni fase, i suoi diagrammi



DIAGRAMMI DI SEQUENZA

- Definizione
  - Descrivono la collaborazione di un gruppo di **oggetti (non classi!!!)** che devono **implementare** collettivamente un comportamento solitamente relativo a uno **scenario** di un caso d'uso

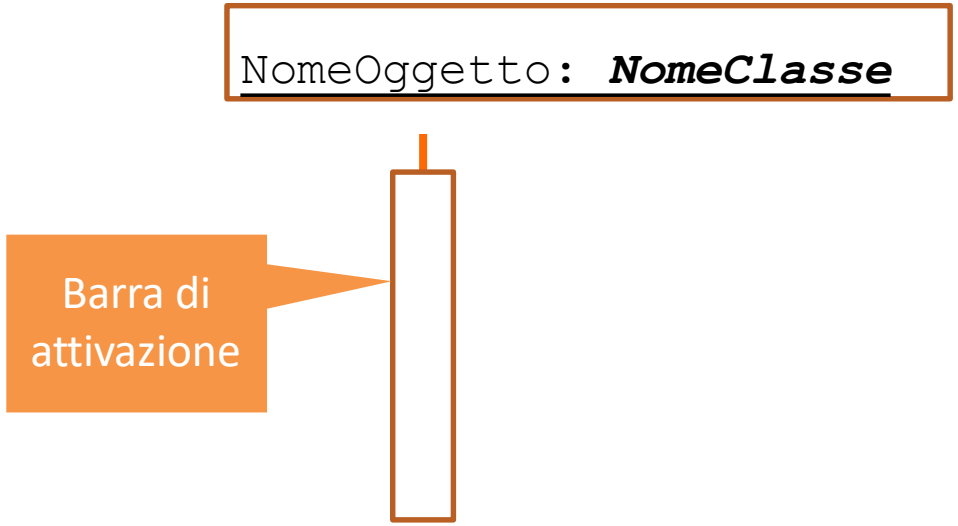


SOMMARIO

- Introduzione
- Partecipanti e messaggi
- Concetti avanzati
- Diagrammi di comunicazione/collaborazione

PARTECIPANTI

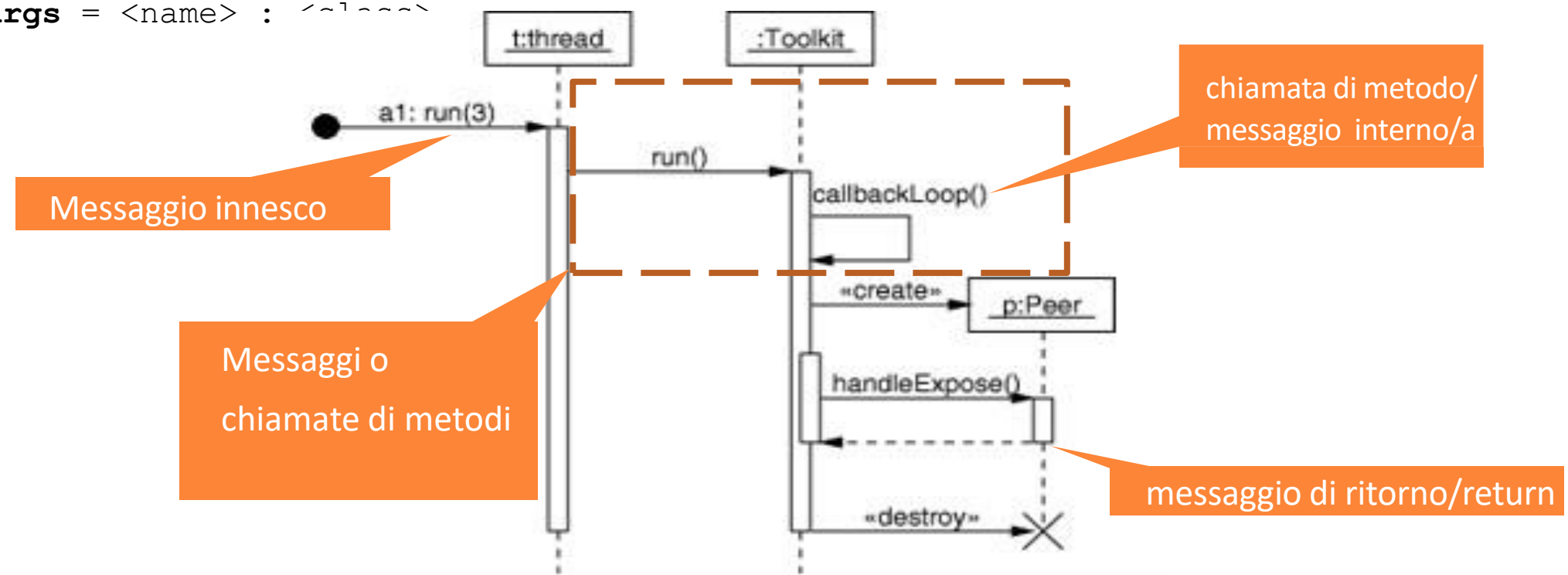
- Entità che detengono il flusso del caso d'uso
  - UML 1.x -> Istanze di classi (**oggett**i)
  - UML 2.x -> Concetto più ampio
    - Eliminata la sottolineatura
  - Barra di attivazione
    - Indica in quale momento un partecipante è attivo il tempo scorre dall'alto in basso
  - Opzionale, ma molto utile



MESSAGGI (SEGNALI)

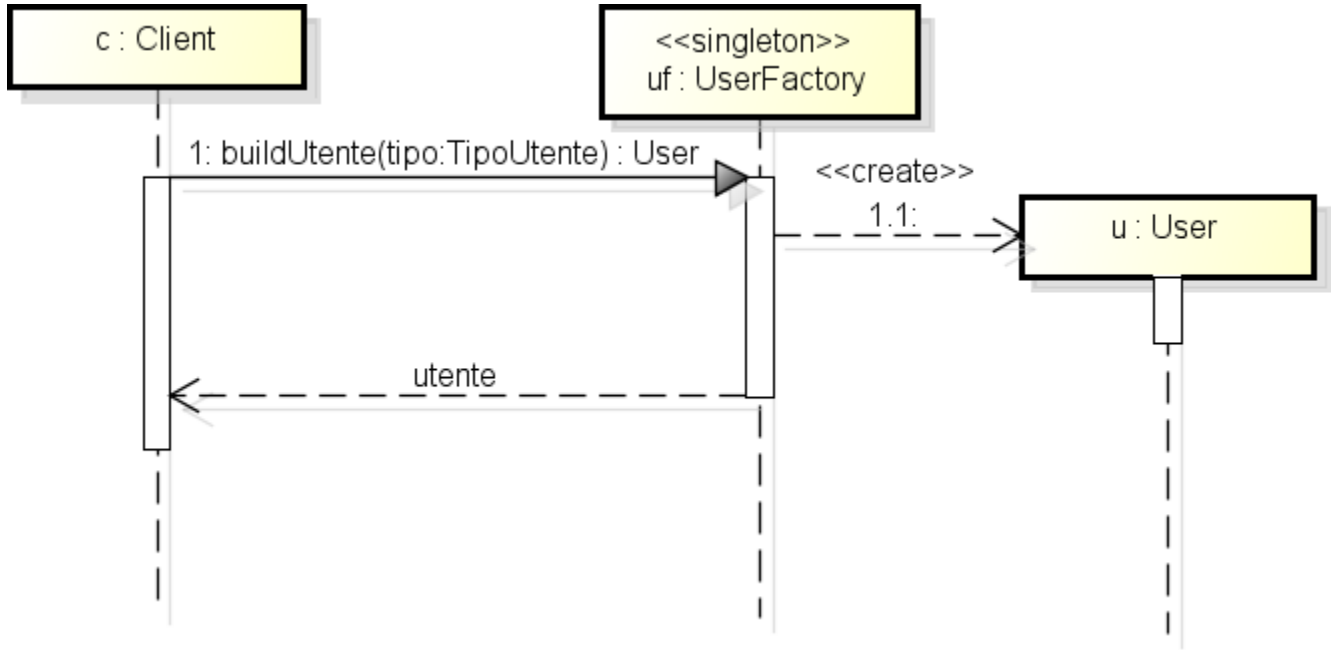
- Dati e operazioni scambiati tra i partecipanti
  - **Chiamata a metodi** degli oggetti da parte di altri oggetti
  - Messaggio di innesco/trigger msg
    - Primo messaggio che scaturisce dall'esterno del diagramma di sequenza

```
attribute = signal_name (args) : return_type
args = <name> : <class>
```



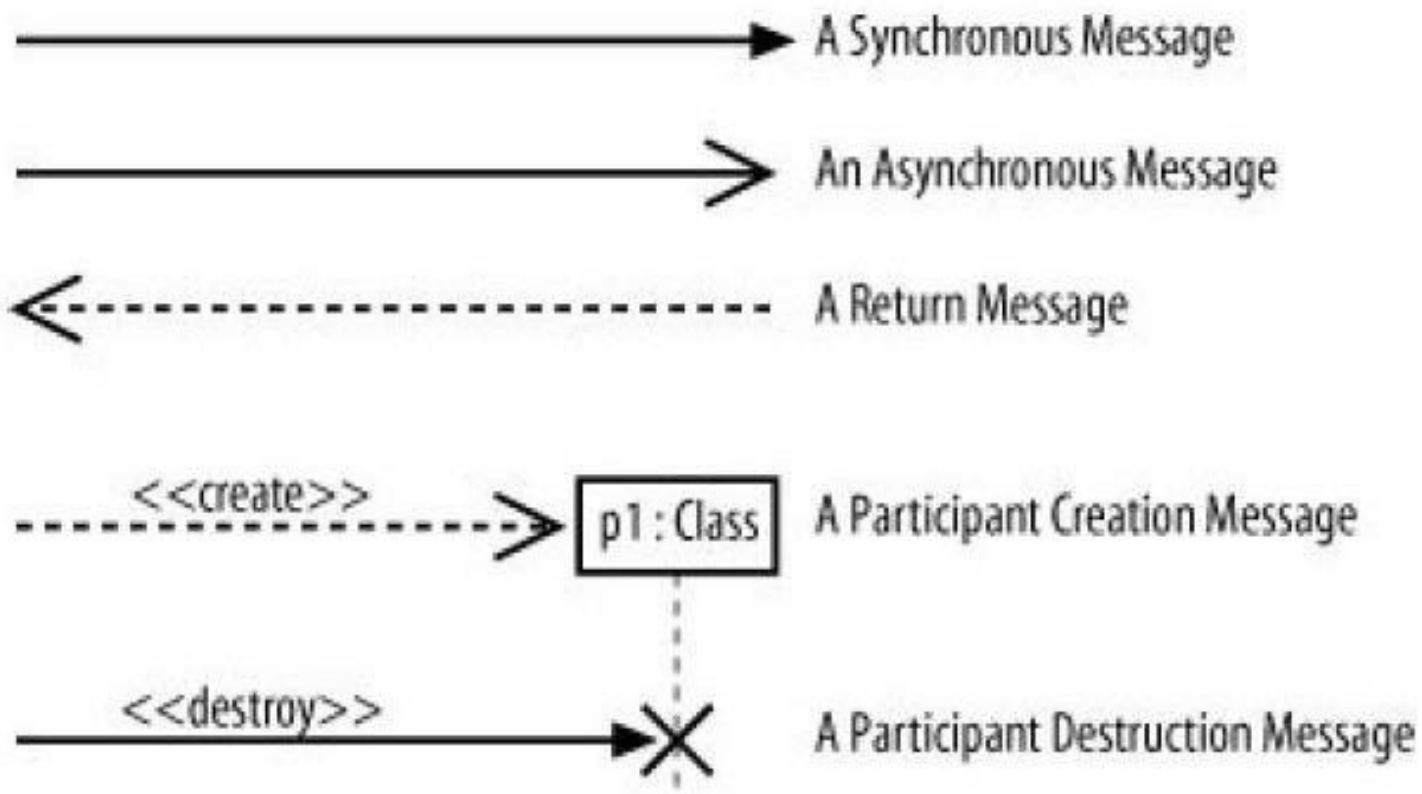
ESEMPIO

- Creazione di un Utente da parte di un Client



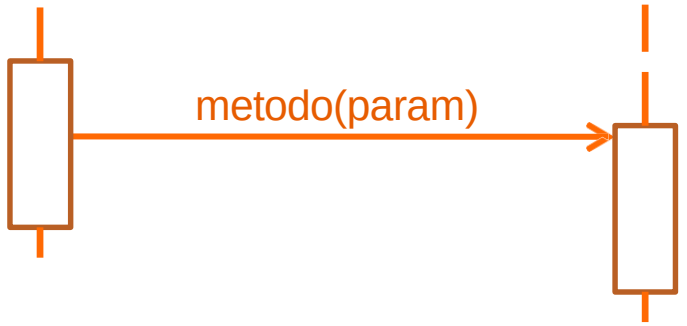
MESSAGGI /SEGNALI / chiamate di METODI

- Tipologie di messaggi o segnali o **chiamate di metodi** (da sx verso dx) o **ritorni** (da dx verso sx)

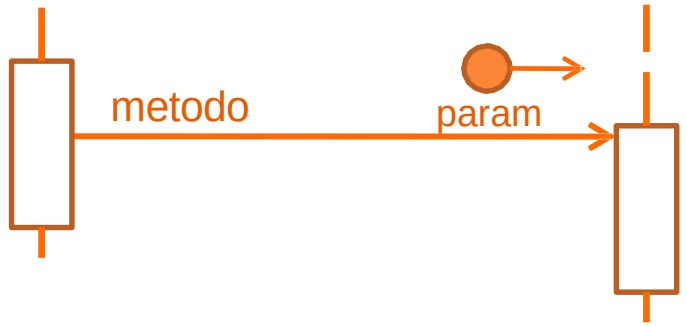


MESSAGGI /SEGNALI / chiamate di METODI

- Passaggio di dati
  - Nessuna tecnica di modellazione standard!!!
  - Metodo classico

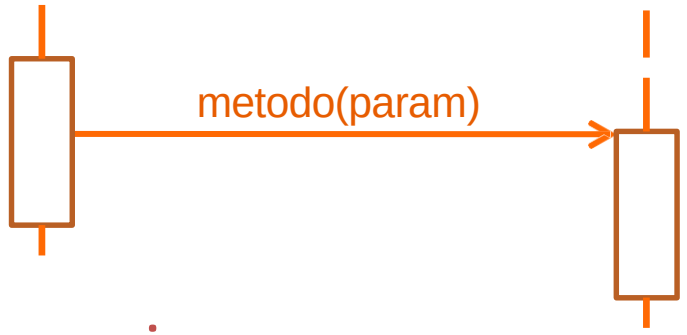


  - Girini dei dati (*data tadpoles*)

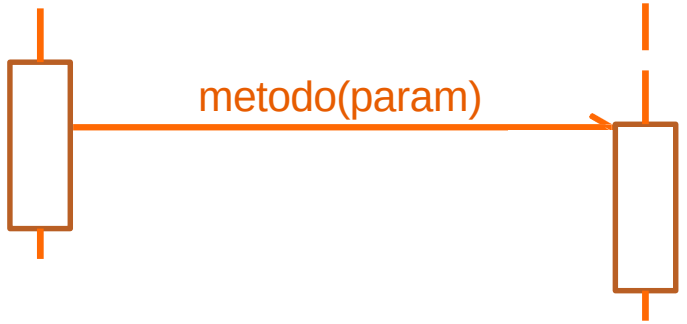


MESSAGGI /SEGNALI / chiamate di METODI

- Messaggi sincroni
  - Il chiamante rimane in attesa della risposta



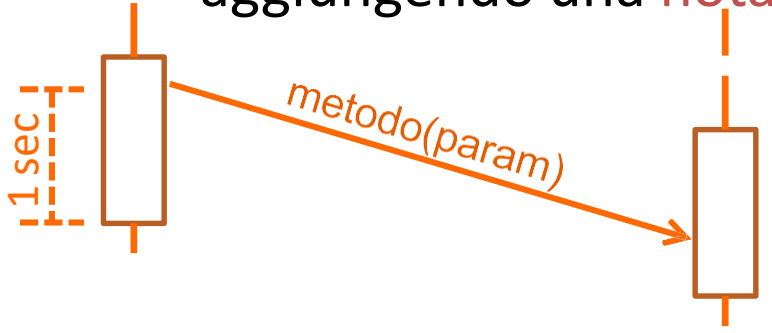
- Messaggi asincroni
  - Il chiamante non rimane in attesa della risposta

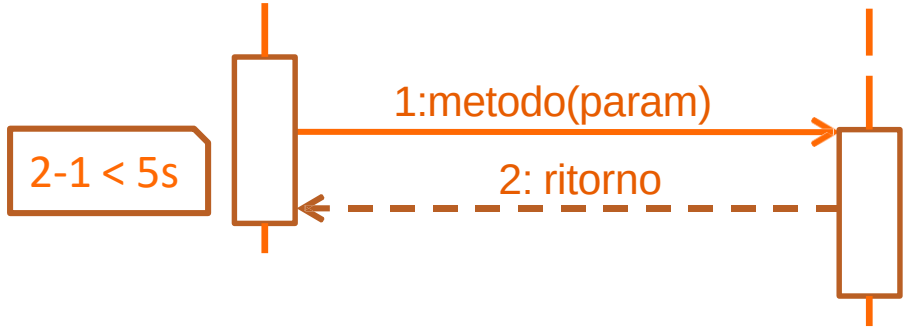


MESSAGGI /SEGNALI / chiamate di METODI

- Messaggi ritorno
  - Da utilizzare solo se necessario per chiarezza

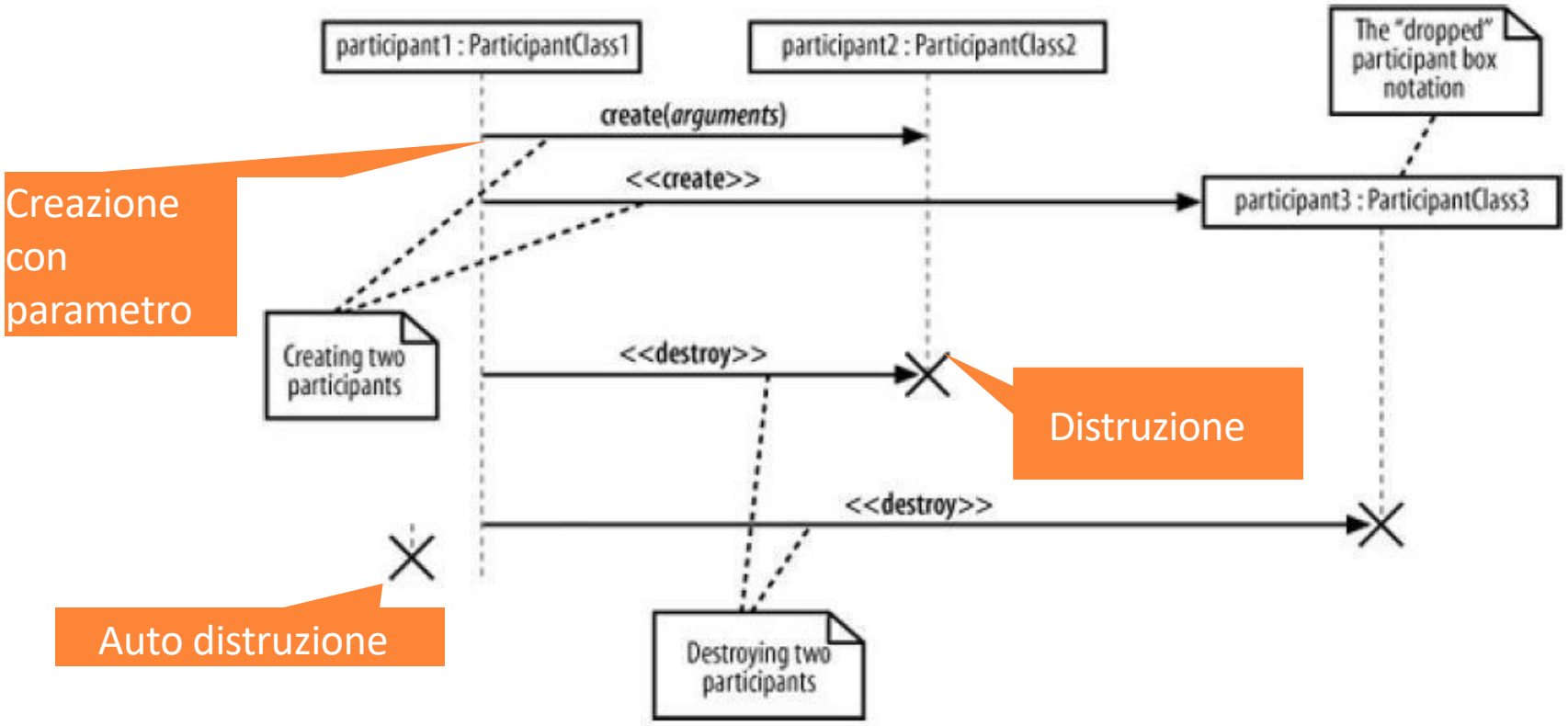
- Tempo
  - Solitamente il tempo di trasmissione è trascurabile
  - Se non lo è si può annotare la durata o etichettare i messaggi aggiungendo una nota



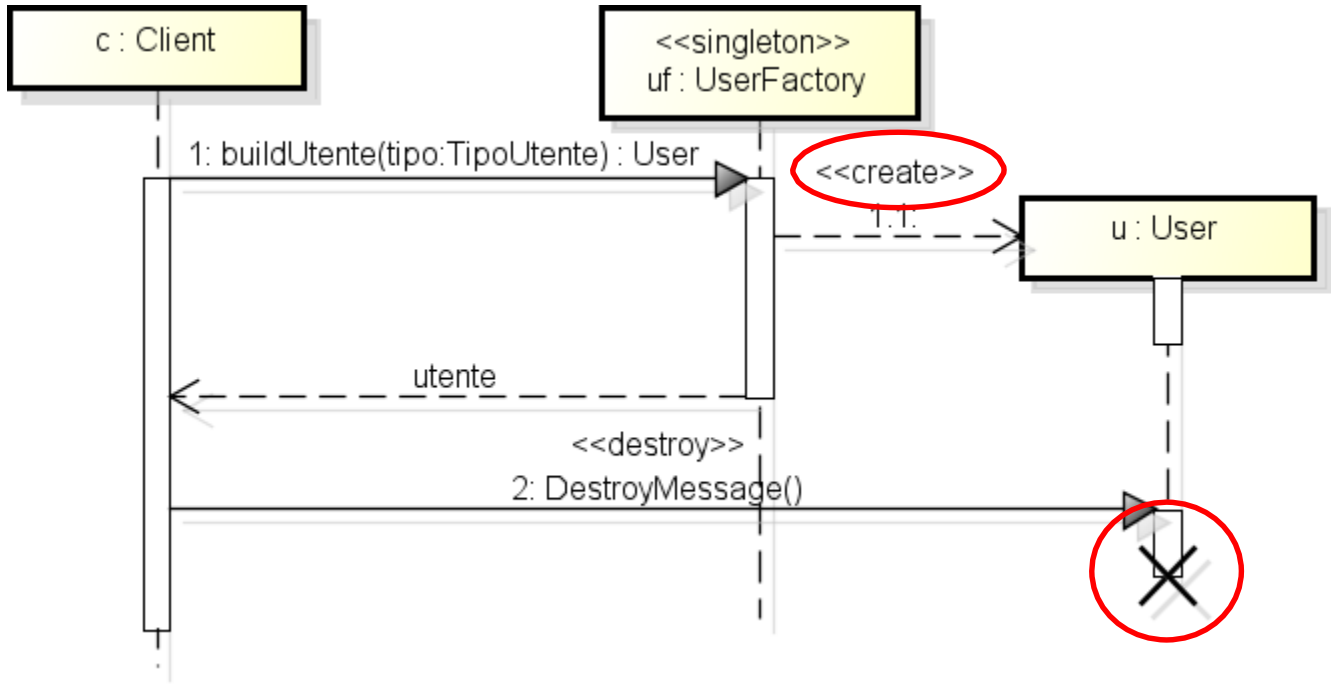


MESSAGGI /SEGNALI / chiamate di METODI

- Creazione partecipanti
  - segnale con lo **stereotipo <<new>>** o **<<create>>**
- Distruzione segnale con lo **stereotipo <<destroy>>**



ESEMPIO


c : Client
<<singleton>>
uf : UserFactory
1: buildUtente(tipo:TipoUtente) : User
<<create>>
1.1:
u : User
utente
<<destroy>>
2: DestroyMessage()

SOMMARIO

- Introduzione
- Partecipanti e messaggi
- Concetti avanzati
- Diagrammi di comunicazione/collaborazione

CICLI E CONDIZIONI

- **Condizioni** solitamente limitate a uno scenario con **condizioni guardia** o un ramo **else**
- **Cicli** solitamente limitati a **iterazioni indicate** con asterisco * e/o **condizioni guardia**

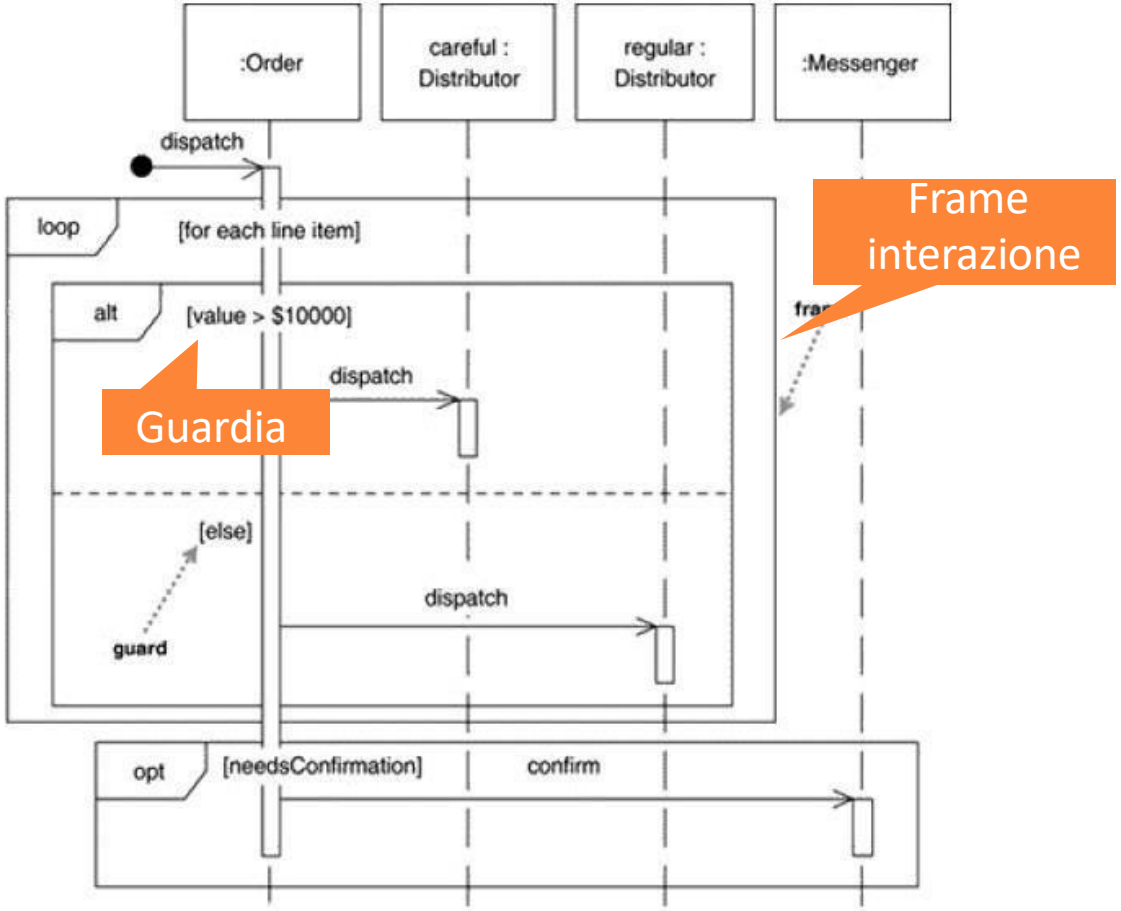
- **La specifica di dettaglio è lasciata all'implementatore**

CICLI E CONDIZIONI

- Frame di interazione (UML 2)

Pseudocodice

```
procedura spedizione  foreach
  (elementoLinea)
    if (prodotto.valore > 10K)
      raccomandata.spedizione
    else
      normale.spedizione end if
  end for
  if (neccesitaConferma)
    messenger.conferma
  end if
end procedura
```



CICLI E CONDIZIONI

- Frame di interazione

Operatore	Significato
alt	Frammenti multipli in alternativa; verrà eseguito solo quello per cui è verificata la condizione.
opt	Opzionale; il frammento viene eseguito solo se la condizione specificata è verificata. Equivalente a alt con solo una freccia.
par	Parallelo; ogni frammento è eseguito in parallelo.
loop	Ciclo; il frammento può essere eseguito più volte, la base dell'iterazione è indicata dalla guardia.
region	Regione critica; il frammento può essere eseguito da un solo thread alla volta.
neg	Negativo; il frammento mostra un'interazione non valida.
ref	Riferimento; si riferisce ad un'interazione definita in un altro diagramma
sd	Sequence diagram; utilizzato per racchiudere un intero diagramma di Sequenza come chiamata innestata.

MODELLAZIONE

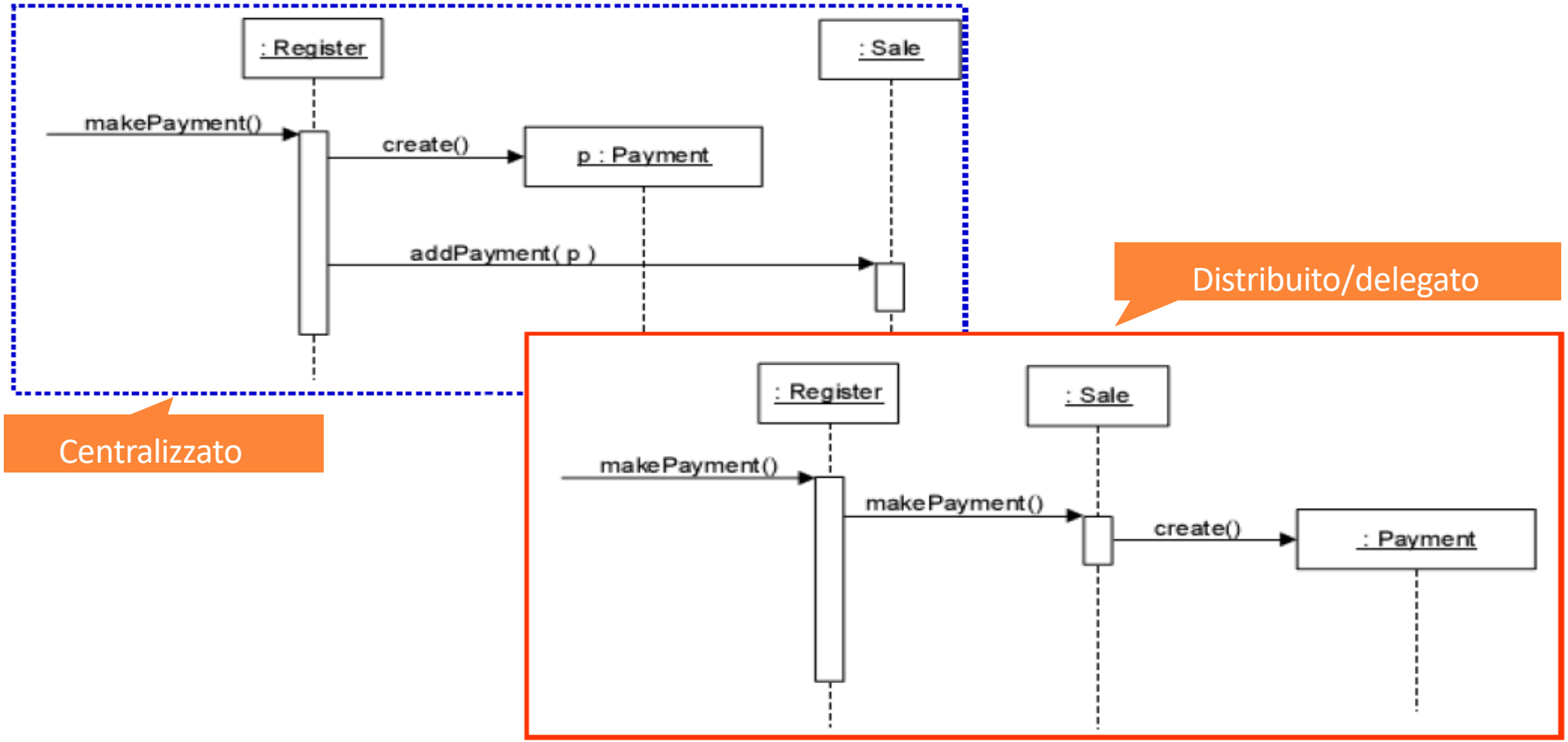
- Ottimi per modellare le collaborazioni fra oggetti
  - Non la logica di controllo, non il codice

- Inadeguatezza a modellare cicli e condizioni …
  - Meglio i **diagrammi di attività activity diagrams**
    - … o pseudocodice allegato(non standard UML) …
    - -

- Controllo centralizzato VS Distribuito
  - Centralizzato
    - **Unico partecipante che governa l'elaborazione**
  - Distribuito
    - **Suddivisione dei compiti dei partecipanti**

MODELLAZIONE

- Controllo centralizzato VS Distribuito



Relazione con Diagramma di Classe ed ERRORI comuni

- Nel diagramma di SEQUENZA compaiono **oggetti** con nome (es. ***X: NomeClasse***) o oggetti **oggetti anonimi** (es. ***:NomeClasse***) tutti appartenenti a classi, mentre NON compaiono MAI CLASSI ISOLATE (es. ***NomeClasse***)

- Tutte le **classi** cui appartengono gli oggetti DEVONO essere **dichiarate nel diagramma di classe**

- Tutti i **metodi** (segnali, messaggi) **chiamati sugli oggetti** DEVONO essere **definiti nella classe dell'oggetto** su cui il metodo è chiamato (quello sulla punta della freccia) e NON sulla classe dell'oggetto chiamante (quello da cui parte la freccia)

SOMMARIO

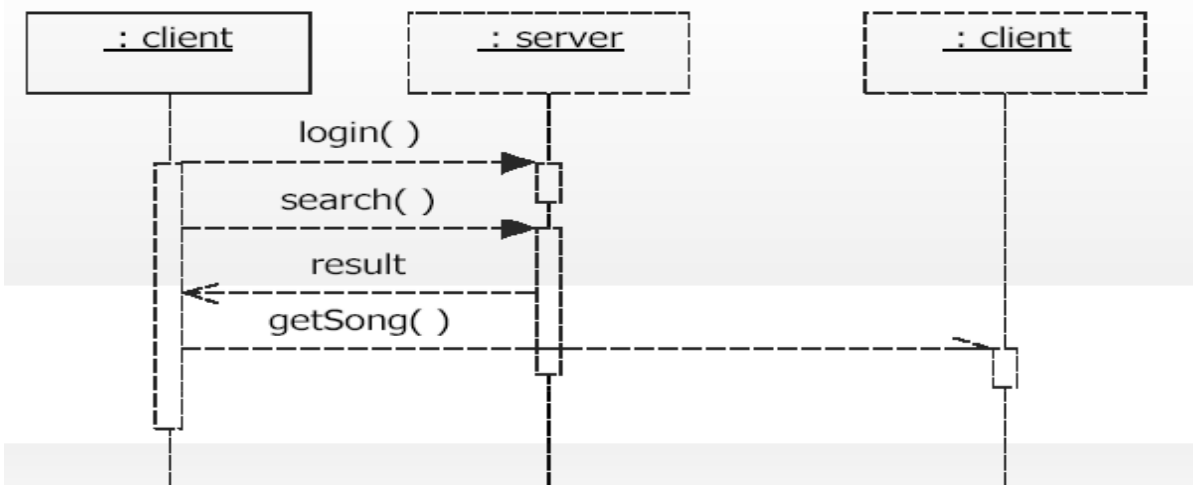
- Introduzione
- Partecipanti e messaggi
- Concetti avanzati
- Diagrammi di comunicazione/collaborazione

DIAGRAMMI DI COLLABORAZIONE/COMUNICAZIONE

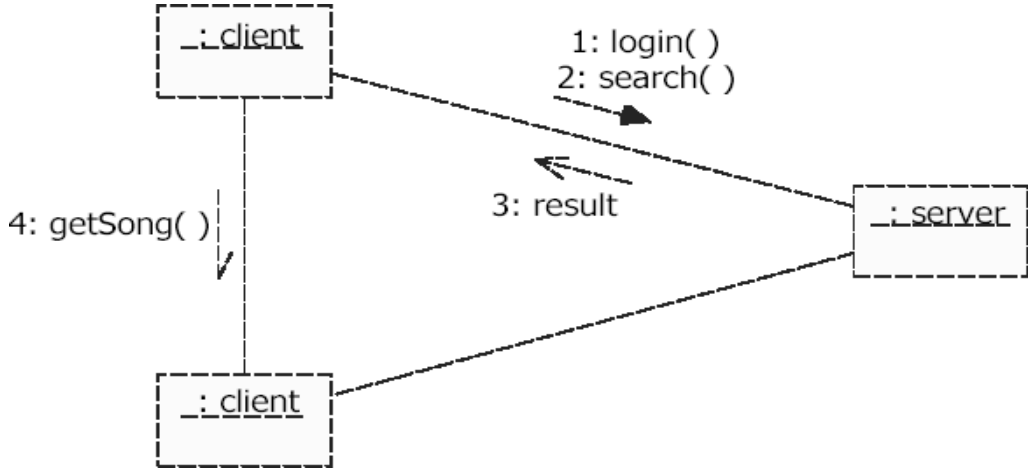
- I **diagrammi di collaborazione** o anche **diagrammi di comunicazione** mostrano una particolare sequenza di messaggi scambiata tra un certo numero di oggetti
  - esattamente come i diagrammi di sequenza
  - I diagrammi di sequenza sono utilizzati per modellare il flusso del controllo rispetto all'ordinamento temporale
  - sono migliori per mostrare la sequenza dei messaggi
  - non sono adatti per rappresentare costrutti condizionali ed iterativi complessi
- i diagrammi di **collaborazione/comunicazione** sono utilizzati per modellare l'organizzazione del flusso del controllo
  - mostrano i collegamenti tra gli oggetti considerando una particolare sequenza di messaggi alla volta
  - I diagrammi di Collaborazione sono **COMPLEMENTARI** ai diagrammi di Sequenza

ESEMPIO

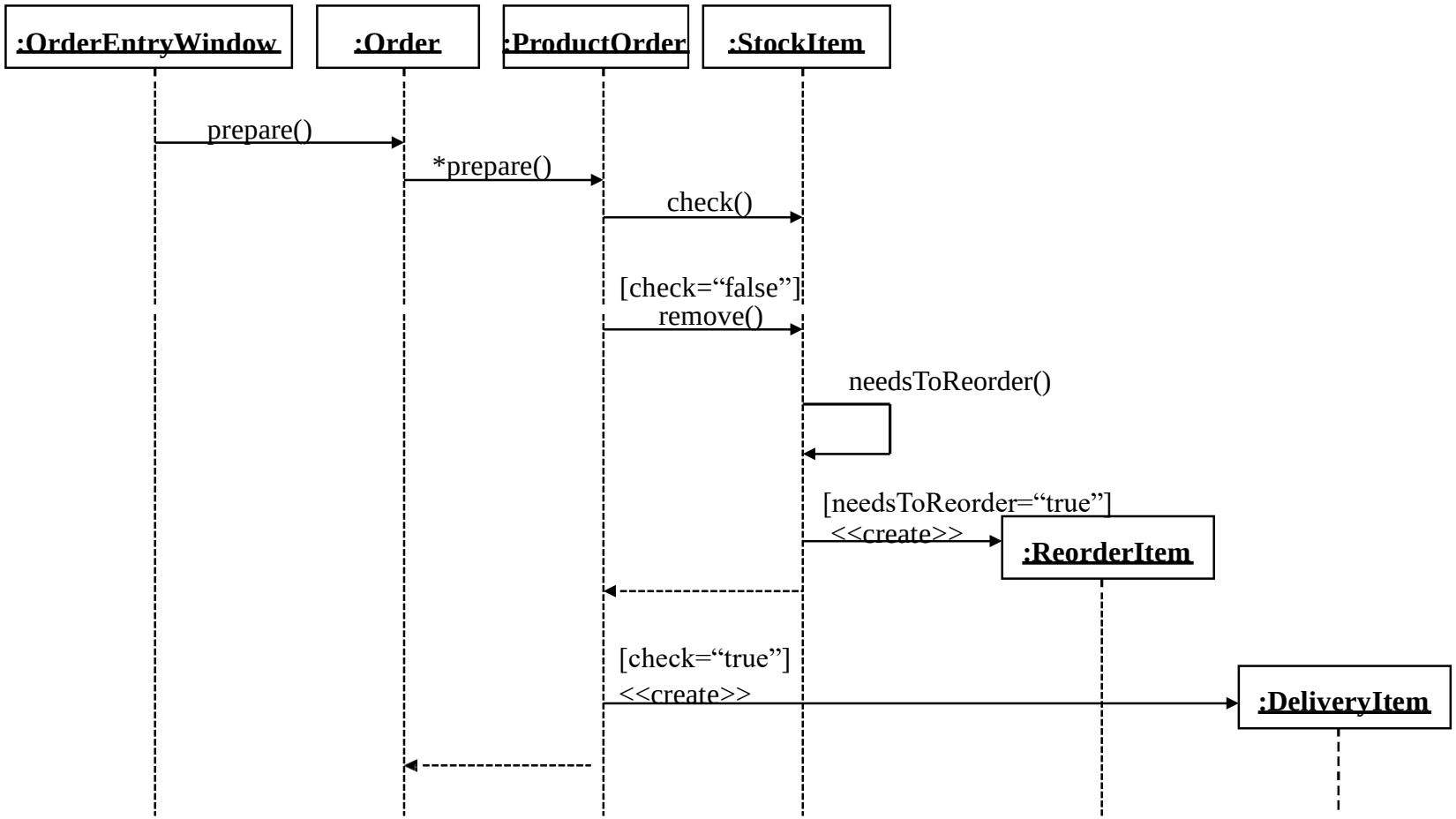
- Diagramma di **sequenza**



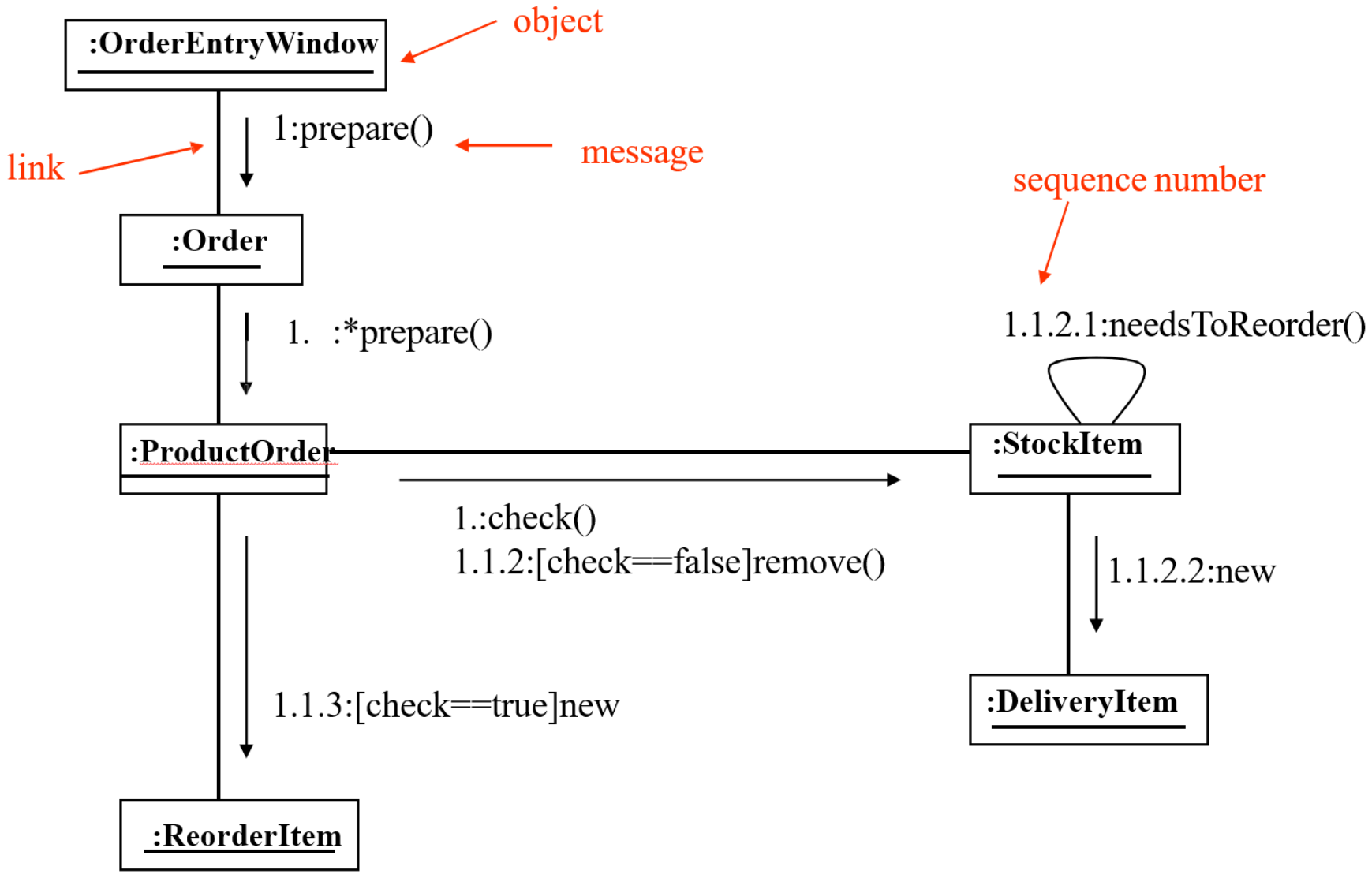
- Diagramma di **collaborazione** corrispondente



ESEMPIO DI DIAGRAMMA DI SEQUENZA E …



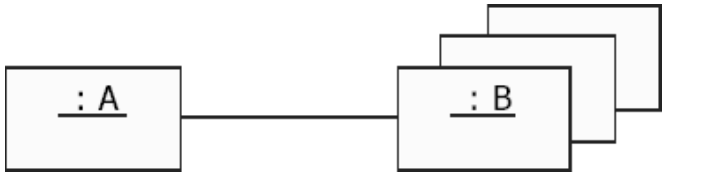
… IL SUO DIAGRAMMA DI COMUNICAZIONE



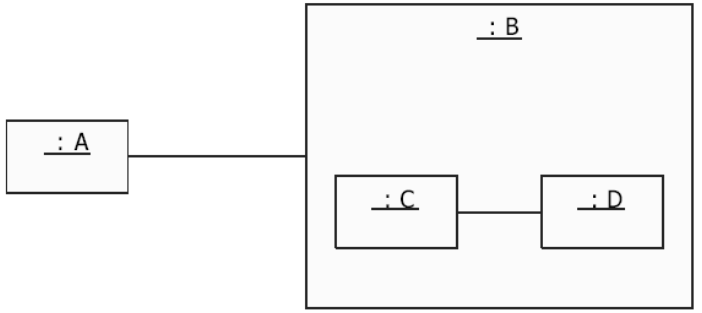
DIAGRAMMI DI COMUNICAZIONE/COLLABORAZIONE

- Componenti alcune convenzioni

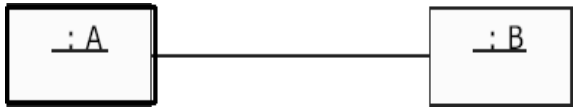
  - **Multi oggetto**
    - dialogo tra un oggetto di classe :A ed un di un insieme di oggetti di classe :B



  - **Oggetti composti (oggetto di classe :A** interagisce con oggetto compost di classe :B)

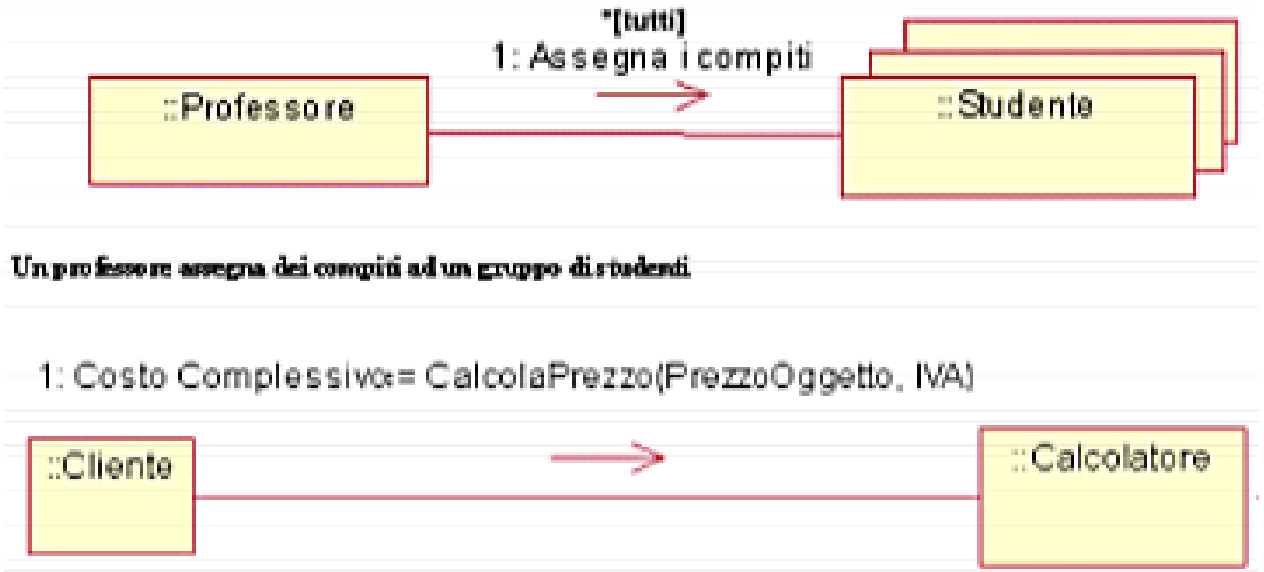


  - **Oggetti attivi**
    - in grado generare autonomamente messaggi e controllo del flusso dei messaggi, indicati **in grassetto**



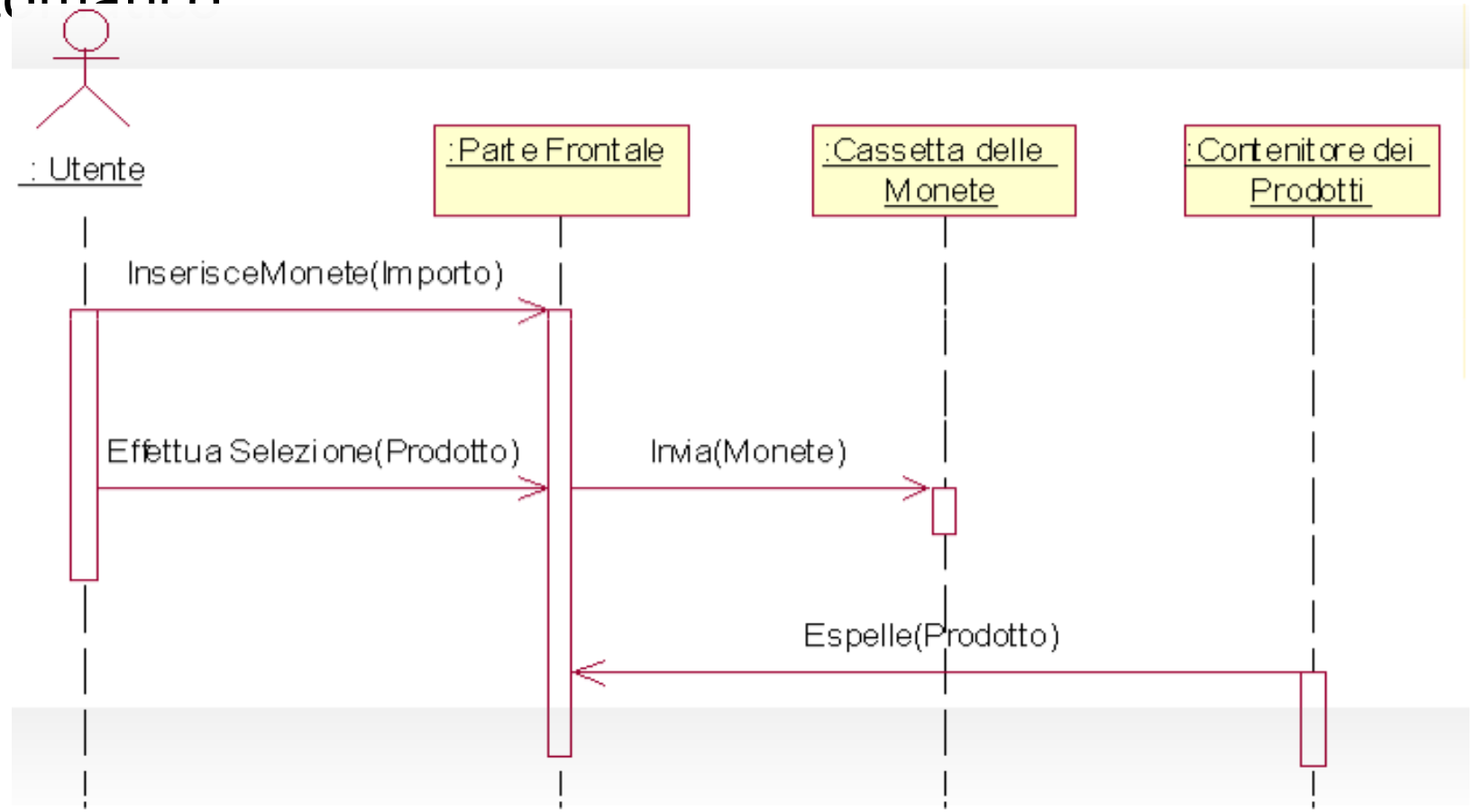
DIAGRAMMI DI COLLABORAZIONE/COMUNICAZIONE

- **etichette numerate** indicano l'ordine sequenziale e strutturato per livelli nei messaggi
- Stereotipi **<<create>>, <<destroy>>** e **[condizioni guardia]** analoghi ai diagrammi di sequenz
- **iteratore asterisco *** per multi messaggi
- **risultati di ritorno** restituiti **espressione assegnata con :=**
  l'espressione assegnata è detta firma del messaggio



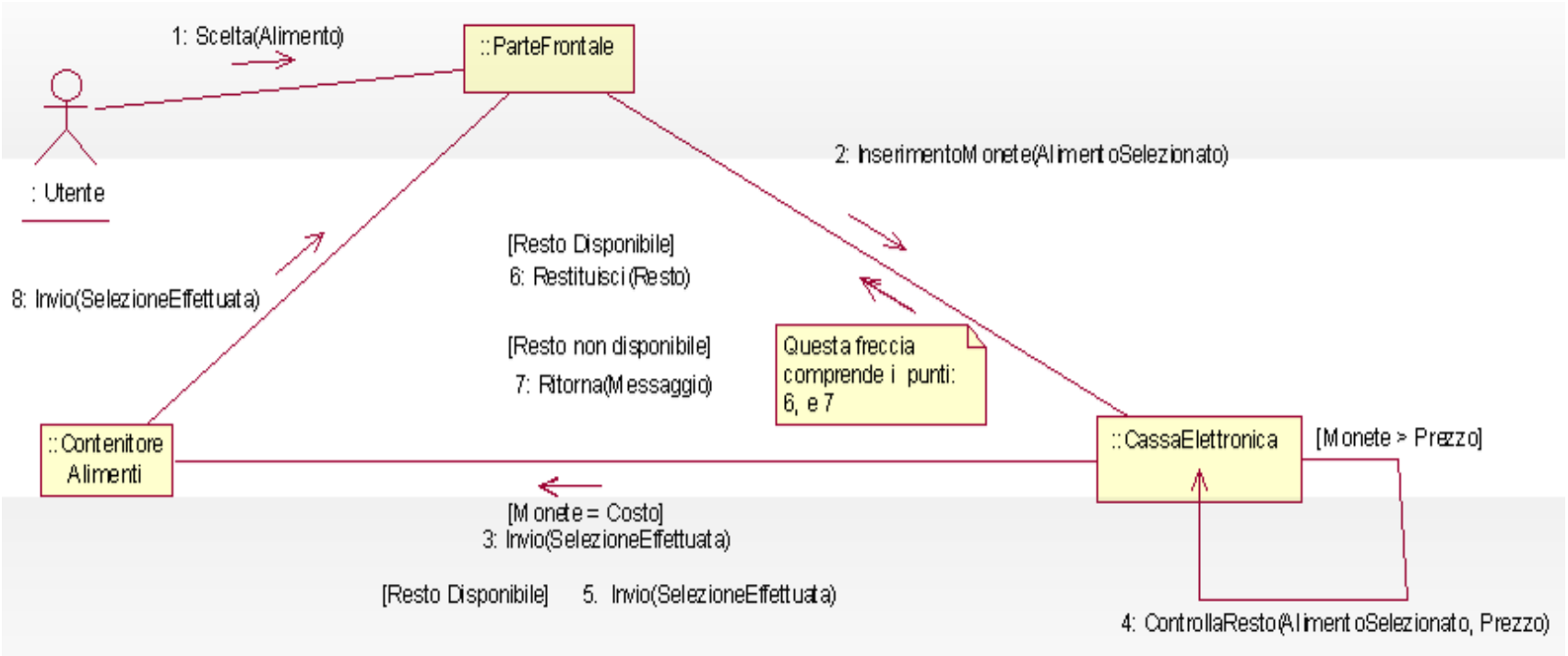
ESEMPIO

- Diagramma di sequenza: scenario principale uso distributore automatico



ESEMPIO

- Diagramma di comunicazione



RIFERIMENTI

- OMG Homepage
  - www.omg.org
- UML Homepage
  - www.uml.org
- UML Distilled, Martin Fowler, 2004, Pearson (Addison Wesley)
- Learning UML 2.0, Kim Hamilton, Russell Miles, O'Reilly, 2006