



# UML2

## Diagrammi di Interazione

Andrea Polini

Ingegneria del Software  
Corso di Laurea in Informatica

# Comportamento dinamico delle classi

Dopo aver definito un modello dei casi d'uso e realizzato un modello di analisi abbiamo bisogno di rispondere alle seguenti domande:

- Come le classi devono interagire per realizzare il comportamento definito da un caso d'uso?
- Quali messaggi devono essere scambiati?

Bisogna considerare che:

- In un contesto di sviluppo iterativo il lavoro condotto porta tipicamente a **modifiche a manufatti precedentemente definiti**
- È importante mantenere i **manufatti allineati e coerenti**
- no dettagli eccessivi quali parametri specifici delle operazioni e loro tipo

# Realizzazione dei casi d'uso

Realizzazione dei casi d'uso è attività che coinvolge:

- Diagramma delle classi di analisi
- Diagrammi di Interazione
- Requisiti Speciali
- Raffinamento dei casi d'uso

# Realizzazione dei casi d'uso

I principali diagrammi utilizzati nella **concretizzazione** di un caso d'uso sono i **diagramma di interazione**. Due entità fondamentali costituiscono questo tipo di diagrammi:

- linee di vita
- messaggi

# Linee di vita

Servono a rappresentare un elemento di una classe all'interno di un'interazione. Consta di:

- Nome
- Tipo
- Selettore

# Messaggi

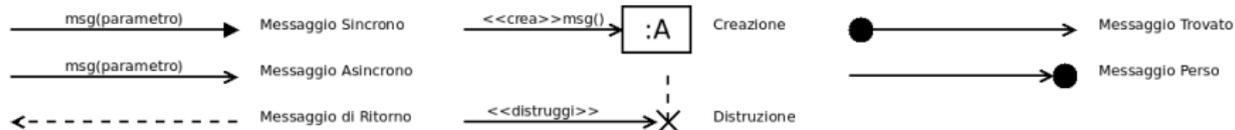
Rappresentano tipo di interazione tra due linee di vita. Una comunicazione si può risolvere in:

- chiamata di un'operazione
- creazione/distruzione di un'istanza
- invio di un segnale

La ricezione di un messaggio attiva il **focus di controllo** per la linea di vita che riceve il messaggio stesso. Il focus risulterà dunque annidato.

# Tipologie di messaggi

UML 2 fornisce la possibilità di specificare le seguenti tipologie di messaggi:



# Diagrammi di interazione

- Diagrammi di sequenza
- Diagrammi di Comunicazione
- Diagrammi di Interazione Generale
- Diagrammi di Temporizzazione

# Diagrammi di Sequenza

Sono la forma di diagramma di interazione più usata nelle fasi dell'analisi e realizzazione dei casi d'uso.

- gli oggetti che interagiscono vengono rappresentati da rettangoli con “coda”
- messaggi vengono rappresentati tra le linee di vita
- il **focus** viene rappresentato da un rettangolo sottile sulla linea di vita
- è possibile rappresentare messaggi annidati
- è possibile rappresentare **invarianti di stato**
- è possibile rappresentare vincoli di durata

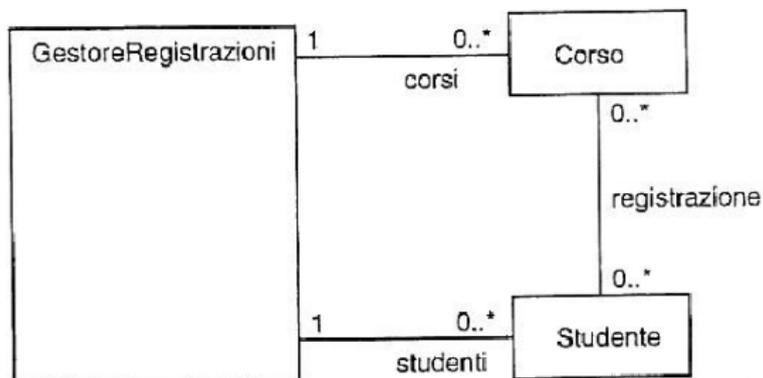
# System Sequence Diagrams

Ristorante 2.0

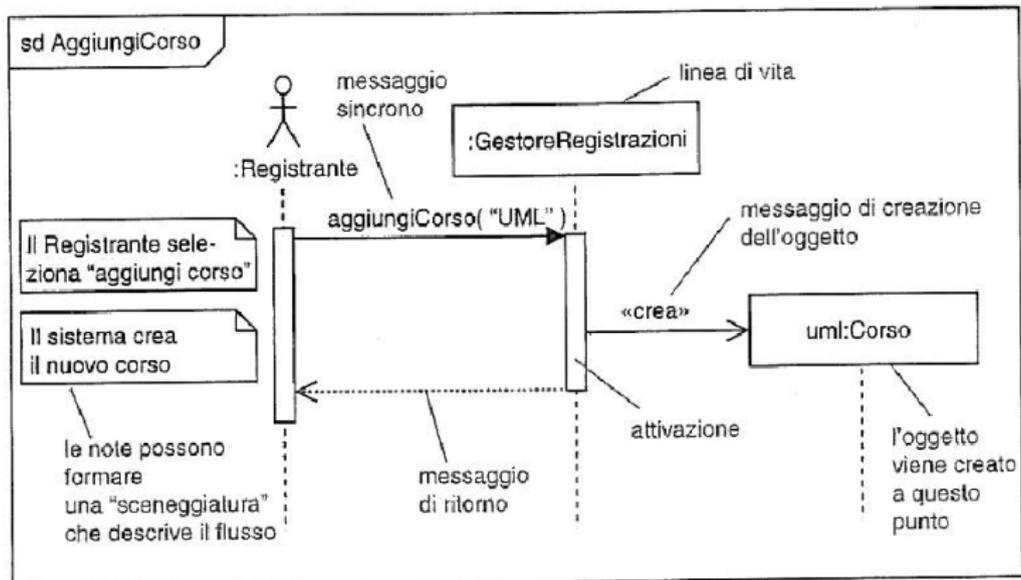
# Esempio - Linee di vita, messaggi ed attivazioni

Caso d'uso: AggiungiCorso
ID:8
Breve descrizione: Aggiunta di dettagli di un nuovo corso al sistema
Attori primari: Registrante
Attori secondari: Nessuno
Precondizioni: 1. Il Registrante si è collegato al sistema
Sequenza degli eventi principale: 1. Il Registrante seleziona "aggiungi corso"      · 2. Il Registrante inserisce il nome del nuovo corso 3. Il sistema crea il nuovo corso
Postcondizioni: 1. Un nuovo corso è stato aggiunto al sistema
Sequenza degli eventi alternativa: CorsoEsiste

# Esempio -Linee di vita, messaggi ed attivazioni



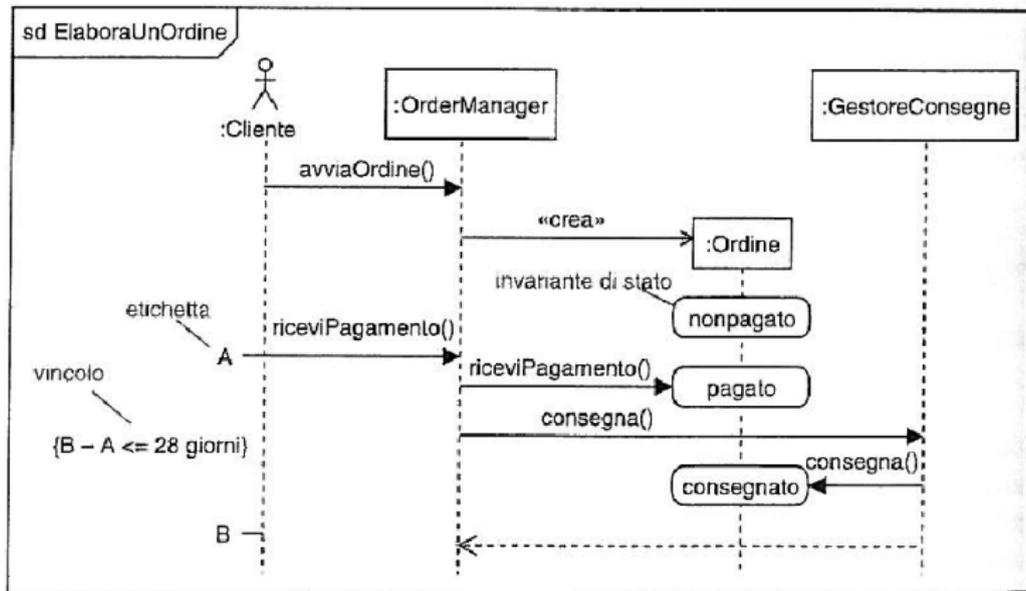
# Esempio - Linee di vita, messaggi ed attivazioni



# Esempio invarianti di stato e vincoli di durata

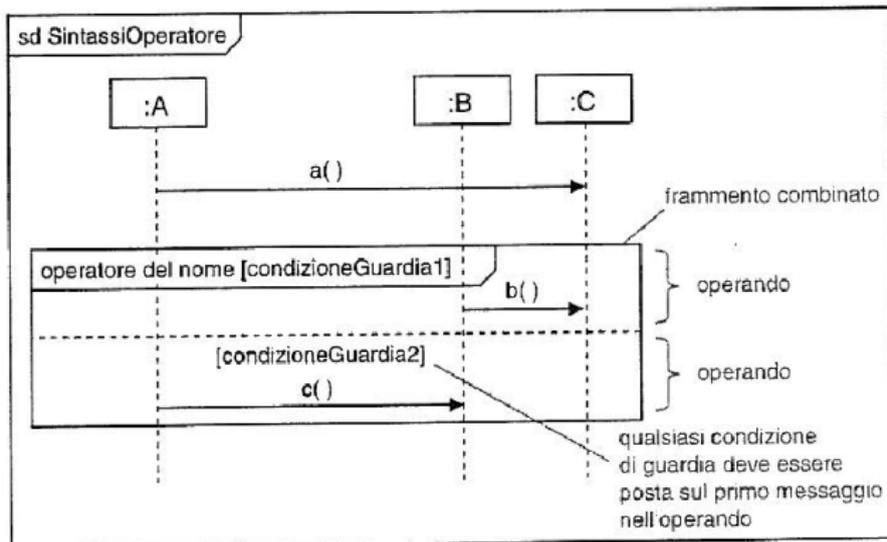
Caso d'uso:ElaboraUnOrdine
ID:5
Breve descrizione Il Cliente avvia un ordine che viene poi pagato e consegnato
Attori primari Cliente
Attori secondari Nessuno.
Precondizioni Nessuna.
Sequenza degli eventi principale 1. Il caso d'uso inizia quando l'attore Cliente crea un nuovo ordine. 2. Il Cliente paga l'ordine a saldo. 3. I prodotti vengono consegnati al Cliente entro 28 giorni dalla data del pagamento finale.
Postcondizioni 1. L'ordine è stato pagato. 2. I prodotti sono stati consegnati entro 28 giorni dal pagamento finale
Sequenza degli eventi alternativa PagamentoInEccesso OrdineAnnullato ProdottiNonConsegnati ProdottiConsegnatiInRitardo PagamentoParziale

# Esempio invarianti di stato e vincoli di durata



# Frammenti combinati

È possibile rappresentare sequenze complesse attraverso l'uso di frammenti combinati. Un frammento combinato ha un **operatore**, uno o più **operandi** e zero o più **condizioni di guardia**. La sintassi per tali costrutti è esemplificata da:



# Frammenti combinati - Tipologie Operatori

- `opt`: sequenza opzionale se condizione vera
- `alt`: sequenze alternative. Eseguita operando con guardia vera
- `loop`: ciclo (*prossima slide*)
- `break`: uscita da un operando
- `ref`: riferimento ad altro diagramma
- `par`: parallelismo
- `critical`: esecuzione atomica
- `seq`: sequenzializzazione debole
- `strict`: sequenzializzazione forte
- `neg`: iterazioni non valide
- `ignore`: elenca messaggi omessi
- `consider`: solo messaggi inclusi
- `assert`: unico comportamento accettabile in quel punto dell'esecuzione

# loop

La sintassi dei loop è data da: `loop min,max [condizione]` Il significato è:

*Esegui il loop un numero minimo di volte `min` continua l'esecuzione finché `condizione` è vera per un numero massimo di `max-min` volte.*

Tipici loop e loro rappresentazione:

- `while (true) {body}`
- `for i=n to m {body}`
- `while (espressioneBooleana) {body}`
- `repeat {body} while (espressioneBooleana)`
- `forEach` oggetto della collezione {body}
- `forEach` oggetto della classe {body}

# loop

La sintassi dei loop è data da: `loop min,max [condizione]` Il significato è:

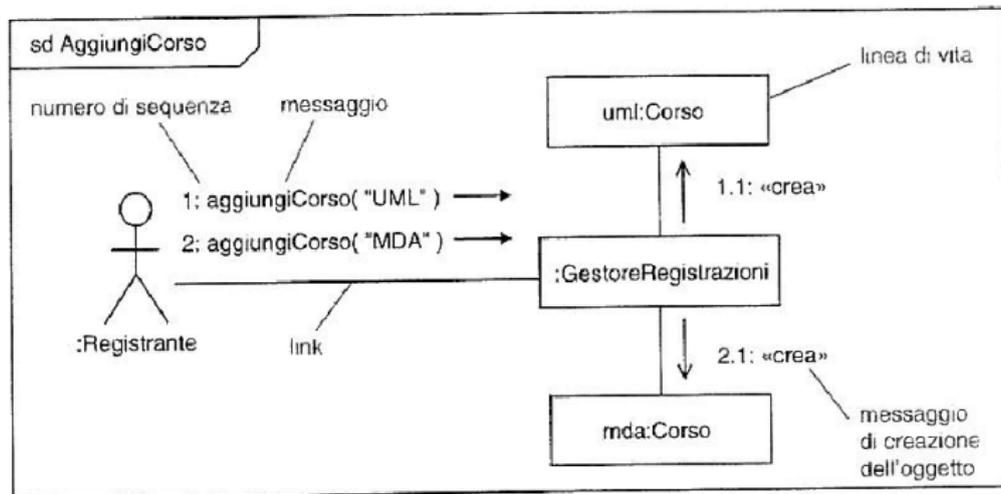
*Esegui il loop un numero minimo di volte `min` continua l'esecuzione finché `condizione` è vera per un numero massimo di `max-min` volte.*

Tipici loop e loro rappresentazione:

- **while** (true) {body}
- **for** i=n **to** m {body}
- **while** (espressioneBooleana) {body}
- **repeat** {body} **while** (espressioneBooleana)
- **forEach** oggetto della collezione {body}
- **forEach** oggetto della classe {body}

# Diagrammi di comunicazione

Si focalizzano sugli aspetti strutturali dell'interazione mostrando come le linee di vita sono collegate tra loro



Il potere espressivo è inferiore a quello dei diagrammi di sequenza è possibile comunque esprimere **iterazioni e ramificazioni**.