UML

In questo documento sono esposti ed illustrati i vari diagrammi definiti da UML.

I diagrammi presentati sono:

* Diagrammi di casi d’uso;
* Diagrammi di classe, degli oggetti e di package;
* Diagrammi di interazione: sequenza e collaborazione;
* Diagrammi fisici: componenti e deployment;
* Diagrammi di stato;
* Diagrammi di attività.

**Diagramma di casi d’uso.**

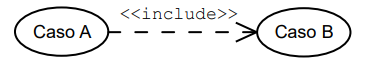
Lo scopo dei diagrammi di casi d’uso `e descrivere ad alto livello l’interazione fra il sistema e uno o piu` attori che richiedono un servizio.

Ogni caso d’uso `e una raccolta di una o piu` situazioni (scenari) logicamente raggruppabili che possono svilupparsi a partire da un operazione iniziale.

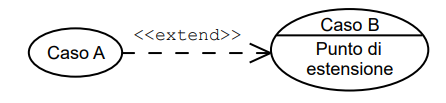
Ogni attore `e presente nel diagramma come un omino stilizzato che pu`o interagire col sistema per vie diverse, generando cos`ı casi d’uso differenti.

I casi d’uso sono rappresentati da forme ovali, e sono collegati agli attori da linee rette

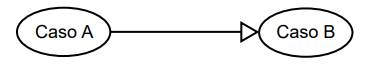
inclusione: indica se un caso d’uso ne include logicamente un altro. E` rappresentata da una freccia tratteggiata, marcata con <> e diretta verso il caso incluso:



estensione: indica una variazione precisa allo sviluppo normale di un caso d’uso. E` rappresentata da una freccia tratteggiata, marcata con 2 <> e diretta verso il caso che viene esteso, nel quale vengono specificati i punti di estensione



generalizzazione: indica se un caso estende informalmente il comportamento di un altro.



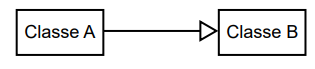
**Diagrammi di classe, di package e degli oggetti**

I diagrammi di classe sono il nucleo portante di UML. Il loro compito `e mostrare le relazioni fra le classi che lo sviluppatore definisce per il proprio progetto.

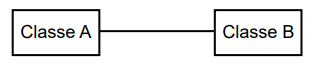
Ogni classe `e rappresentata da un rettangolo diviso al piu` in tre parti, per il nome, gli attributi e i metodi. Le regole di visibilit`a sono + (pubblico), # (protetto) e – (privato), e vanno indicate prima del nome di un elemento di classe.

E` possibile definire un’ampia gamma di relazioni statiche e dinamiche nella struttura delle classi, in particolare:

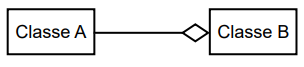
**generalizzazione**: quando una classe `e sottoclasse di un’altra, la relazione viene indicata da una freccia con punta a triangolo bianco (la frecci punta sulla classe archetipa):



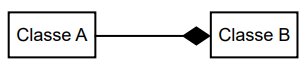
**associazione**: due classi sono associate se nel sistema sono in relazione fra di loro. Il legame viene espresso da una linea retta che congiunge le classi associate, Ai due capi della linea possono anche essere indicate delle molteplicit`a diverse dalla “uno a uno” di base, tramite un numero (molteplicit`a esatta), un asterisco (m. infinita) o un intervallo (m. opzionale, ad es., 4 0..2). Le rette di congiunzione possono diventare frecce per indicare la direzione del legame.



aggregazione: due classi sono aggregate se una rappresenta logicamente una parte dell’altra. Il legame `e segnalato da una retta con un piccolo rombo bianco dal lato della classe includente, Le opzioni di molteplicit`a sono le stesse del punto precedente.



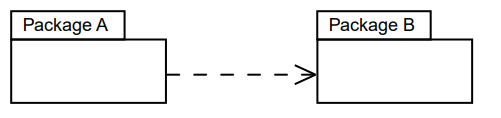
composizione: come per l’aggregazione, ma la classe parte `e vincolata ad un’unica classe contenitore. La notazione prevede un rombo nero al posto di quello bianco:



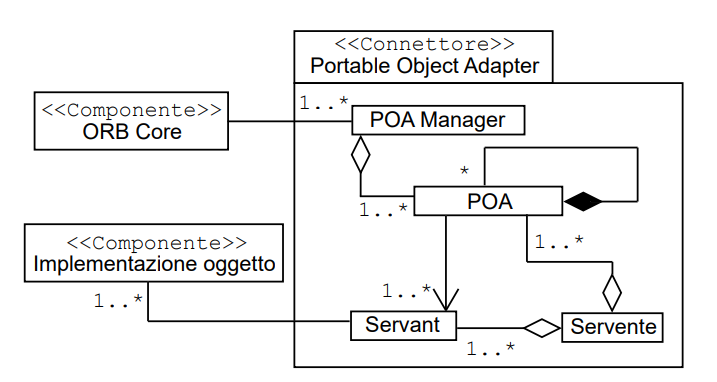
stereotipazione: UML permette di definire nuovi costrutti di modellazione che vengono chiamati stereotipi. Quando una classe ne implementa uno (ad esempio, un’interfaccia), questo deve essere indicato fra e assieme al nome della classe stessa.

Un package java è una raccolta di classi che formano un pacchetto autonomo.

. I package sono indicati come cartelle e **le relazioni di dipendenza vengono rappresentate da frecce tratteggiate**:



Gli oggetti vengono indicati con la notazione nome:classe

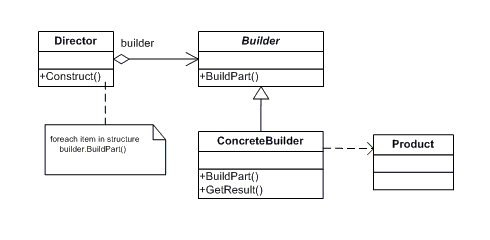


Il diagramma di classe in Figura 3 espande una parte della Figura 2, illustrando la composizione interna del Connettore Servente. Tale Connettore `e in realt`a un package contenente quattro classi diverse, POA Manager, POA, Servant e Servente. Le classi sono indicate solo con i nomi, senza attributi e metodi. Possiamo notare numerose relazioni fra le classi; in particolare, **ogni POA `e incluso in almeno un POA Manager e un Servente** (aggregazioni), **ha associato almeno un Servant** (associazione) e pu`o essere legato ad altri POA (composizione).

La freccia continua indica associazione (----🡪)

Gli oggetti vengono indicati con la notazione nome:classe (anche parziale, solo nome o classe), e possono eventualmente riportare dei valori distintivi di attributo. Fra gli oggetti sono generalmente presenti dei collegamenti tramite linee rette, che derivano dalle relazioni di associazione, composizione o aggregazione definite fra le classi originali

Facciamo una possibile analisi qui:



Director include builder (rombetto) ed è associate a bulder (freccia), concrete builder eredita da bulder (freccia triangolo), e product dipende da concrete builder (freccia tratteggiata dipendenza…)

**Diagrammi di sequenza**

Un diagramma di sequenza mostra lo scambio di messaggi fra tutti gli oggetti che vengono via via coinvolti in una attivit`a iniziata da un attore.

Il diagramma si sviluppa dall’alto verso il basso seguendo un’ipotetica linea temporale.

Gli oggetti (istanze di classi) che partecipano all’esecuzione vengono tutti indicati nella parte alta del diagramma con un rettangolo che ne riporta il nome e/o il tipo

La vita di ogni oggetto viene rappresentata da una linea tratteggiata verticale: su di essa possono essere tracciati stretti rettangoli piu` o meno alti, ad indicare i periodi di effettiva attivit`a dell’oggetto

Gli oggetti possono essere creati o distrutti da altri oggetti in corso di esecuzione.

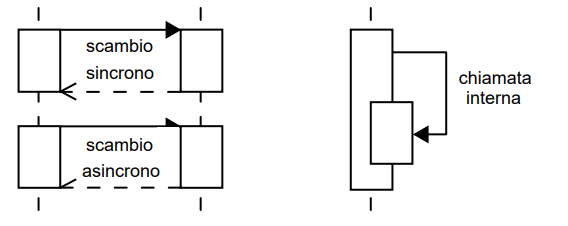
La distruzione di un oggetto viene invece indicata da una grande X sulla linea di vita:

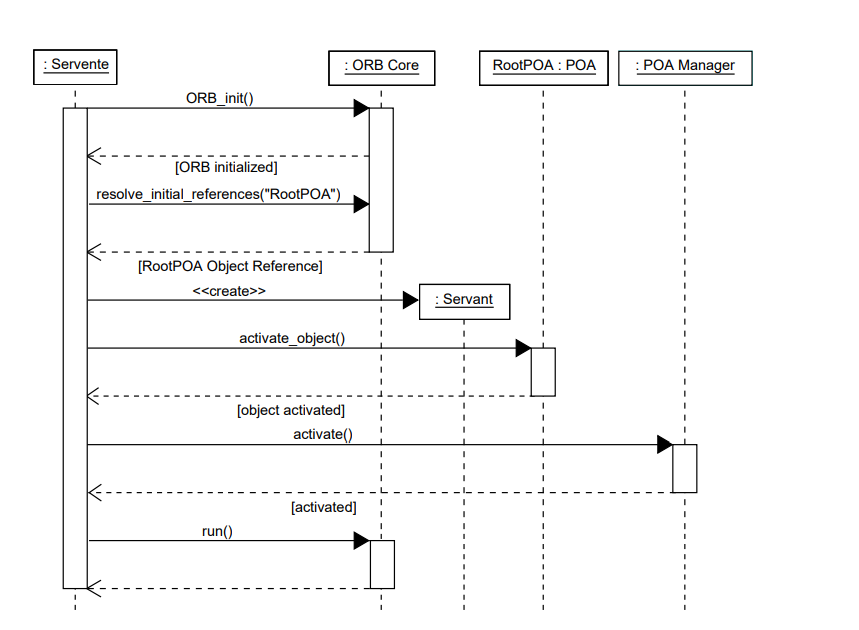
Una scrittura fra parentesi quadre indica un test condizionale il cui risultato determina l’invio del messaggio.

Le frecce dirette a punta piena indicano messaggi sincroni, mentre quelle tratteggiate sono risposte di ritorno.

Quando hanno solo mezza punta rappresentano invece i messaggi asincroni.

Sono possibili anche frecce che ritornano nell’oggetto da cui partono, per indicare chiamate a funzioni interne:

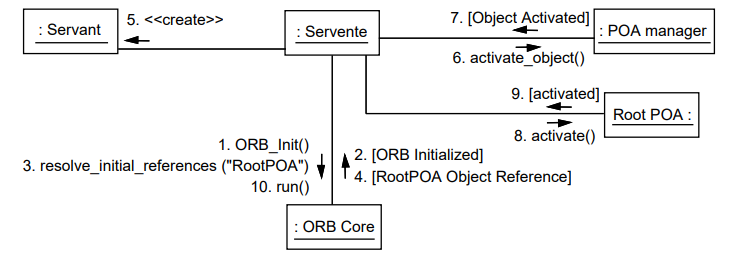




viene presentato il diagramma di sequenza che espande il primo caso d’uso definito in Figura 1. L’oggetto Servente si occupa di iniziare uno scambio di messaggi che porta all’attivazione di tutte le componenti necessarie al funzionamento del sistema CORBA sul lato servente. Sono chiaramente visibili i messaggi di invio marcati con chiamate di funzioni, e quelli di ritorno con le conferme; tutti i messaggi sono sincroni. Il terzo messaggio inviato da Servente porta alla creazione dell’oggetto Servant, che da quel momento in poi diventa parte del diagramma.

**Diagrammi di collaborazione.**

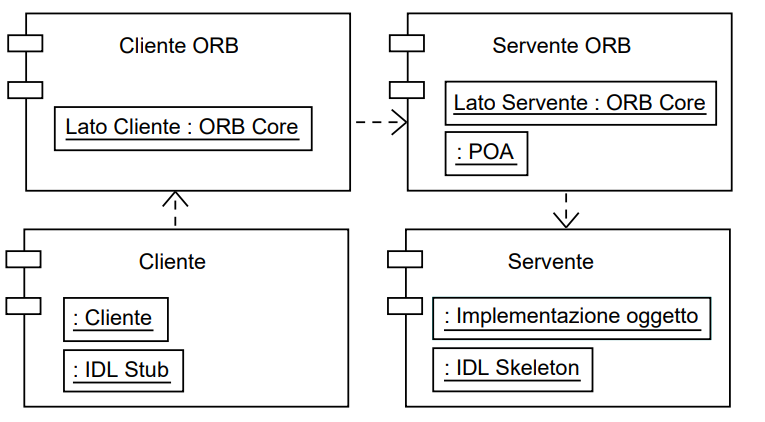
I diagrammi di collaborazione analizzano le stesse situazioni dei diagrammi di sequenza, evidenziando maggiormente i legami fra le varie componenti.



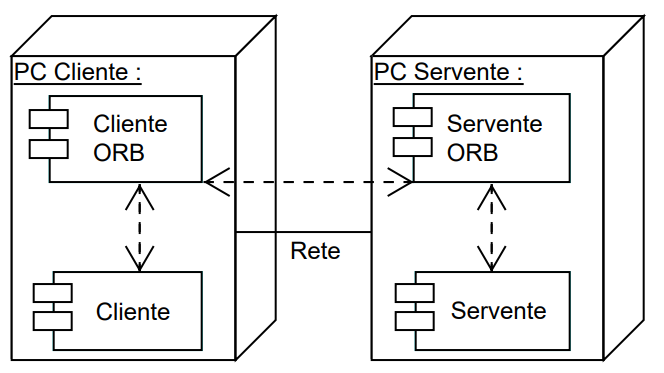
Diagrammi fisici.

I diagrammi fisici sono diagrammi di alto livello utilizzati per mostrare l’implementazione del software su una certa architettura hardware. Diventano essenziali per tutte quei progetti in cui il sistema software `e strettamente dipendente dall’architettura fisica dove dovr`a funzionare.

I diagrammi dei componenti individuano le relazioni presenti, a livello di codice, fra componenti software



I diagrammi di deployment (schieramento, distribuzione) presentano le varie risorse hardware coinvolte dall’esecuzione del software.



**Diagramma di stato.**

Questo tipo di diagramma illustra la vita di un oggetto software durante l’attivit`a del sistema, mostrando tutti i possibili stati in cui esso pu`o venirsi a trovare, e come avviene il passaggio fra gli stati.

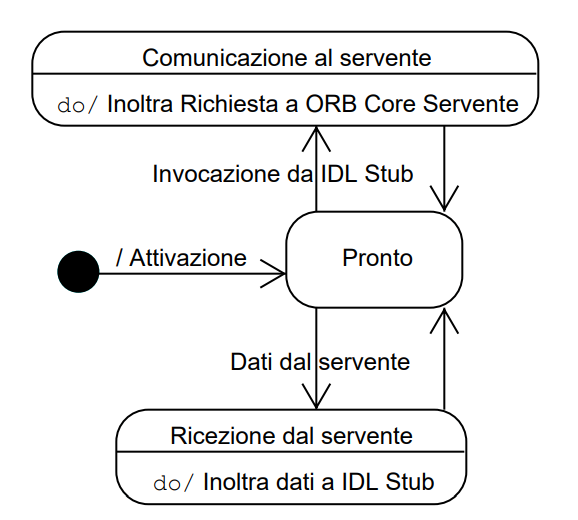
**Diagrammi di attività**

I diagrammi di attivit`a vengono utilizzati per illustrare le sequenze di operazioni che possono nascere da una richiesta o un intervento sul sistema.

Gli stati possono essere di due tipi: • semplice: lo stato `e caratterizzato solo da un nome. La transizione verso un altro stato `e determinata dal verificarsi di un certo evento, segnato sulla freccia di cambio di stato. Se sono definite transizioni verso piu` stati differenti, allora ogni freccia uscente viene marcata da eventi, test condizionali (fra parentesi quadre) ed eventuali operazioni (precedute da una barra, /); • con attivit`a: lo stato ha una operazione associata, il cui nome inizia con do/, che deve essere eseguita quando l’oggetto entra in tale stato, o ci

Sono utili per visualizzare a quali conseguenze e sviluppi pu`o portare una certa operazione. I diagrammi di attivit`a nascono come evoluzione dei classici diagrammi di flusso (flowchart), dai quali ereditano la possibilit`a di definire flussi che dipendono da test condizionali. L’attivit`a ha il suo punto di partenza in un piccolo cerchio nero, e termina in un altro cerchio nero ricerchiato. Le operazioni sono indicate da degli ovali e il flusso da frecce semplici.

I salti condizionali (branch) sono visualizzati come dei rombi,



A questa normale strutturazione, UML aggiunge un’importante novit`a: la possibilit`a di descrivere computazioni parallele. Questi flussi di operazioni si sviluppano contemporaneamente da un certo punto di diramazione (fork), indicato da uno spesso segmento nero orizzontale

