DIETI UNINA UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II
DIPARTIMENTO DI INGEGNERIA ELETTRICA E DELLE TECNOLOGIE DELL'INFORMAZIONE
Corso di Laurea in Ingegneria Informatica

Corso di Ingegneria del Software

UML
Advanced Structure Diagrams

Sommario

- Relazioni di dipendenza
- Package e Diagrammi di Package
- Interfacce e porte in UML
- Classificatori strutturati
- Componenti e diagramma dei componenti
- Nodi e diagramma di deployment

Riferimenti

J.Arlow, I. Neustadt: UML 2 e Unified Process, 2° edizione, McGraw-Hill. **Capp.** 9, 11, 16, 17, 22.

OMG, *Unified Modeling Language: Superstructure, version 2.0*, www.omg.org.

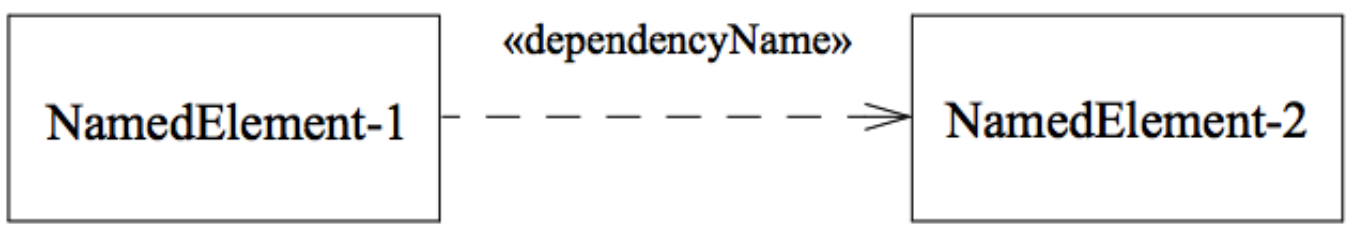
Dipendenza

Una dipendenza indica una relazione fornitore/cliente tra elementi del modello, in cui la modifica del fornitore può avere impatto sugli elementi del cliente

- Esistono tre tipi base di dipendenze:

1. di Uso
2. di Astrazione
3. di Permesso

Il tipo di dipendenza risulta di solito chiaro dal contesto anche senza indicare lo stereotipo, che spesso si omette



Dipendenze di uso

Il cliente A utilizza alcuni dei servizi resi disponibili dal fornitore B per implementare il proprio comportamento

«use»	Il cliente fa un qualche uso del fornitore. È la relazione d'uso più generica e comune, spesso implicita.
«call»	È una dipendenza tra operazioni (o tra operazioni di classificatori): l'operazione cliente invoca l'operaz. fornitore.
«parameter»	Il fornitore è un parametro dell'operazione del cliente.
«send»	Il cliente è un'operazione che invia il fornitore (che deve essere un segnale) a un destinatario non specificato.
«instantiate»	Il cliente può creare istanze del fornitore.

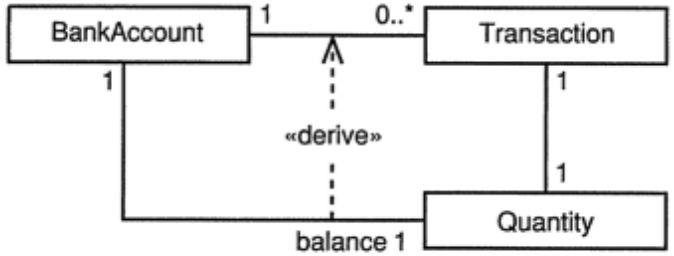
Dipendenze di astrazione

Modellano dipendenze tra elementi che sono ad un diverso livello di astrazione

«trace»	Il fornitore e il cliente rappresentano lo stesso concetto ma appartengono a modelli diversi (es: di analisi e di progettazione).
«substitute»	Inidica la capacità del cliente di sostituire il fornitore in runtime non basandosi sulla specializzazione. Indica cioé una condivisione di interfacce e contract tra il fornitore e il cliente.
«refine»	Ha la stessa semantica di «trace» ma si può utilizzare per lo stesso modello.
«derive»	Indica esplicitamente che un elemento può essere derivato in qualche altro modo da un altro.

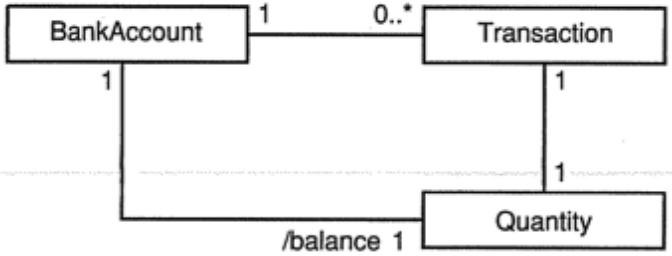
Dipendenze di astrazione

In UML gli elementi si possono denotare come **derivati** anteponendo una barra (“/”) al loro nome



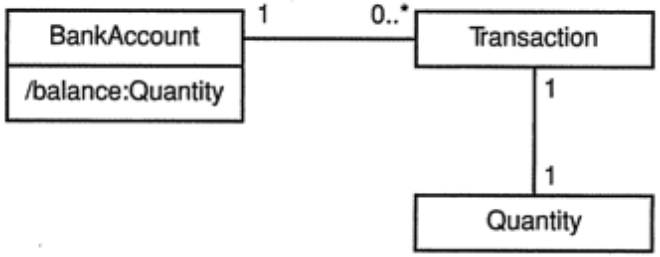
The BankAccount class has a derived association to Quantity where Quantity plays the role of the balance of the BankAccount

This model emphasizes that the balance is derived from the BankAccount's collection of Transactions



In this case a slash is used on the role name to indicate that the relationship between BankAccount and Quantity is derived

This is less explicit as it does not show what the balance is derived from



Here the balance is shown as a derived attribute – this is indicated by the slash that prefixes the attribute name

This is the most concise expression of the dependency

Dipendenze di permesso

Concernono la capacità di un elemento di accedere a un altro elemento

«access»	Dipendenza di permesso tra package, discussa in seguito.
«import»	Dipendenza di permesso tra package, discussa in seguito.
«permit»	Questo tipo di dipendenza consente una violazione controllata della incapsulamento. L'elemento cliente ha accesso all'elemento fornitore, qualunque sia la visibilità dichiarata per ques'ultimo.

Package

Un Package è un meccanismo generalizzato per organizzare e raggruppare **logicamente** elementi (compresi altri package) e diagrammi

- I package facilitano la gestione di sistemi complessi composti da numerose parti
  - Forniscono uno spazio di nomi incapsulato al cui interno i nomi devono essere **univoci**
  - Permettono di raggruppare componenti omogenei del sistema (ad es. classi) che manifestano un elevato livello di coesione interna
  - Consentono di creare confini semantici interni al modello, ad esempio creando diversi package che supportano i diversi insiemi di funzionalità del sistema

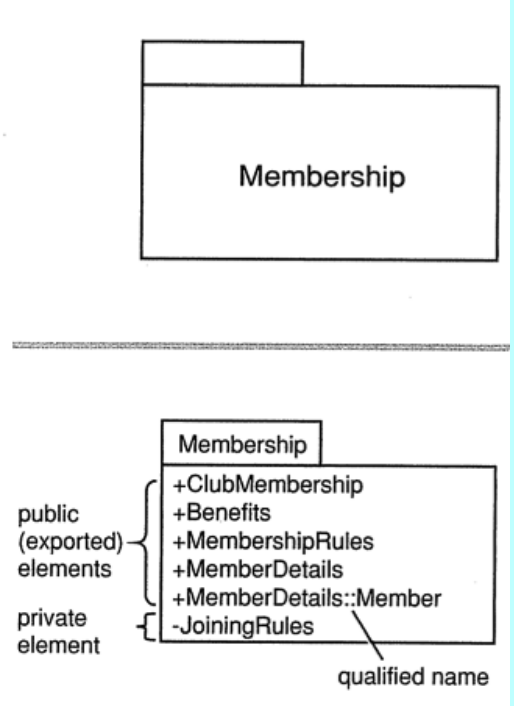
Package

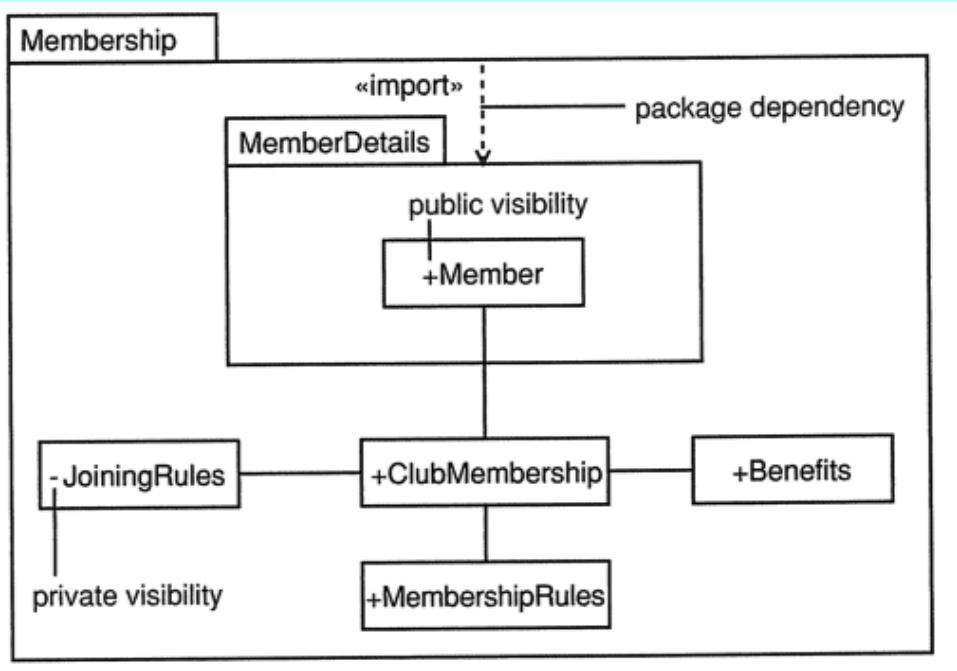
- I package possono formare una struttura ad albero perché ogni elemento del modello appartiene ad un unico package di livello superiore
  - UML identifica il package di livello più alto (che esiste implicitamente) con lo stereotipo «topLevel»
- Gli elementi contenuti in un package possono avere una visibilità che indica se sono visibili o meno ai clienti del package
  - La visibilità consente di limitare il grado d'interdipendenza tra package

Package

Ogni package deve avere un nome univoco che lo distingue dagli altri nel package in cui è annidato:

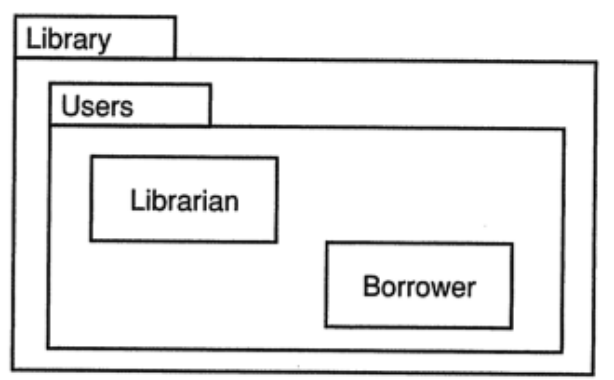
- Per nome (semplice) del package intendiamo la stringa testuale che è associata al package
- Per nome qualificato intendiamo il nome del package con anteposti tutti i nomi dei package separati da **(::)** in cui il package è annidato

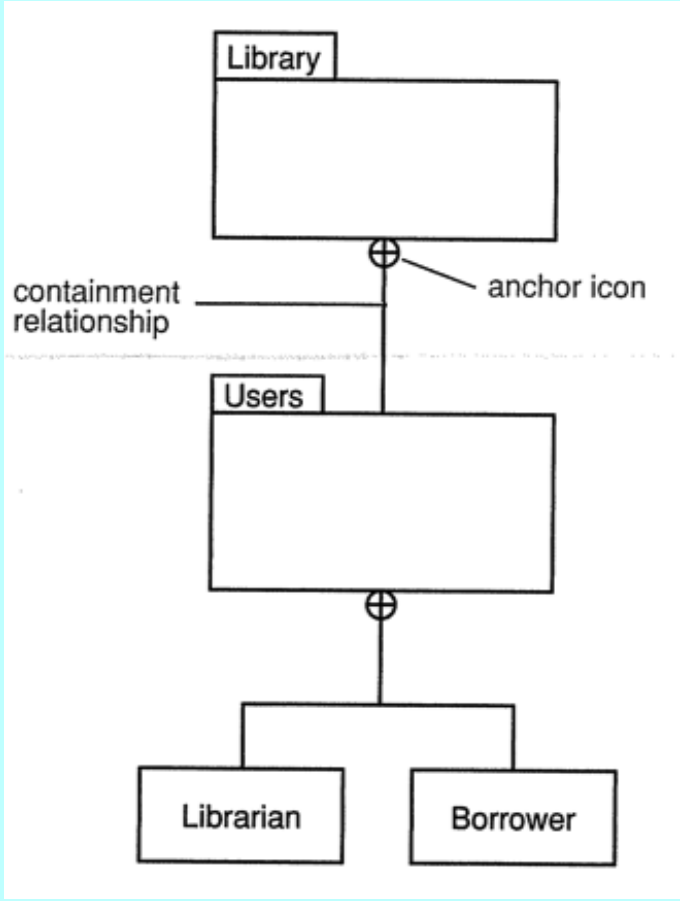




Package annidati

- L'annidamento tra package può essere rappresentato anche tramite la relazione di contenimento
- I package annidati hanno accesso allo spazio dei nomi dei package che li contengono usando nomi non qualificati
- I package contenitori per accedere ai package contenuti devono usare il nome qualificato





Alta Coesione

- È preferibile raggruppare in package gli elementi che hanno alta coesione e adempiono a (poche) responsabilità ben definite
- La coesione interna al package può essere misurata in *class category relational cohesion*:
  - CCRC = NumberOfLinks/NumberOfClasses
- *Range* nominale: 150% - 350% (oltre 350% la complessità aumenta troppo)

Stereotipi dei package

- Lo standard UML fornisce due stereotipi per adattare la semantica dei package a situazioni particolari

«framework»	Un package che contiene elementi del modello che specificano un'architettura riutilizzabile.
«modelLibrary»	Un package che contiene elementi che sono progettati per essere riutilizzati da altri package.

Dipendenze tra package

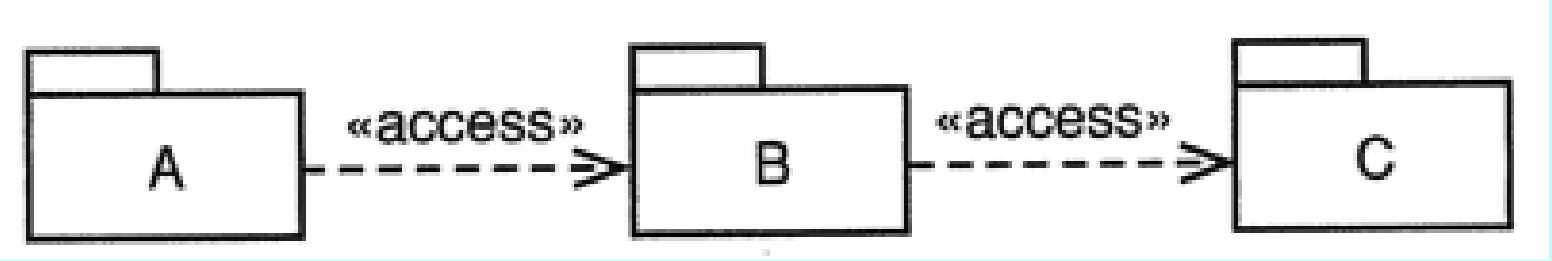
- Oltre alle dipendenze «use» e «trace», che possono essere evidenziate in un diagramma dei package, esistono alcune dipendenze limitate ai package

«import» e «access»	Gli elementi pubblici dello spazio dei nomi del package fornitore vengono aggiunti come elementi pubblici (per «import»)/privati (per «access») allo spazio dei nomi del package cliente. Gli elementi del package cliente possono accedere a tutti gli elementi pubblici del package fornitore usando nomi non qualificati.
«merge»	Gli elementi pubblici del package fornitore vengono incorporati negli elementi del package cliente. Questa dipendenza viene usata solo nella meta-modellazione. Non la si dovrebbe incontrare nella progettazione e nell'analisi OO ordinaria.

Dipendenze tra package

Se il package A dipende da B e B dipende da C può esistere una relazione implicita tra A e C

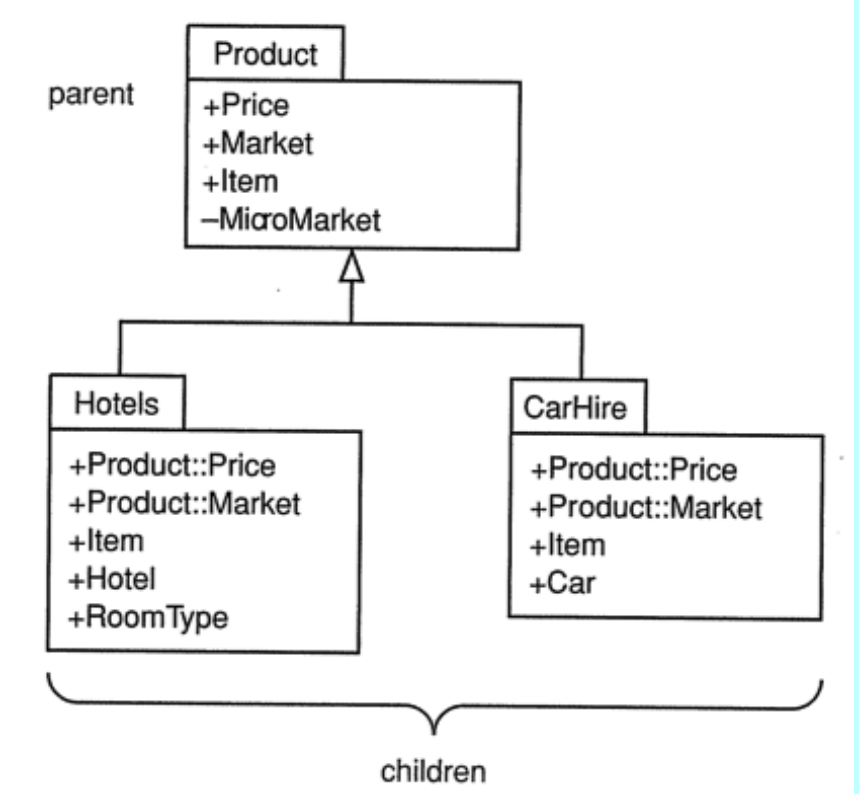
- Se la dipendenza tra B e C è di **«import»** allora esiste per transitività una dipendenza implicita di A da C
  - con «import» gli elementi pubblici di C diventano elementi pubblici B e A può dipendere tramite B da C
- Se la dipendenza tra B e C è di **«access»** allora **non** esiste una dipendenza implicita di A da C
  - con «access» gli elementi pubblici di C diventano elementi privati di B e A può dipendere solamente dall'interfaccia di B



Generalizzazione tra package

Tra package può esistere la relazione di generalizzazione.
I package figli:

- sono più specializzati ed ereditano gli elementi pubblici e protetti del loro package genitore.
- possono aggiungere nuovi elementi e possono ridefinire elementi del genitore fornendone un'implementazione alternativa con lo stesso nome
- vale il principio di sostituibilità con i genitori



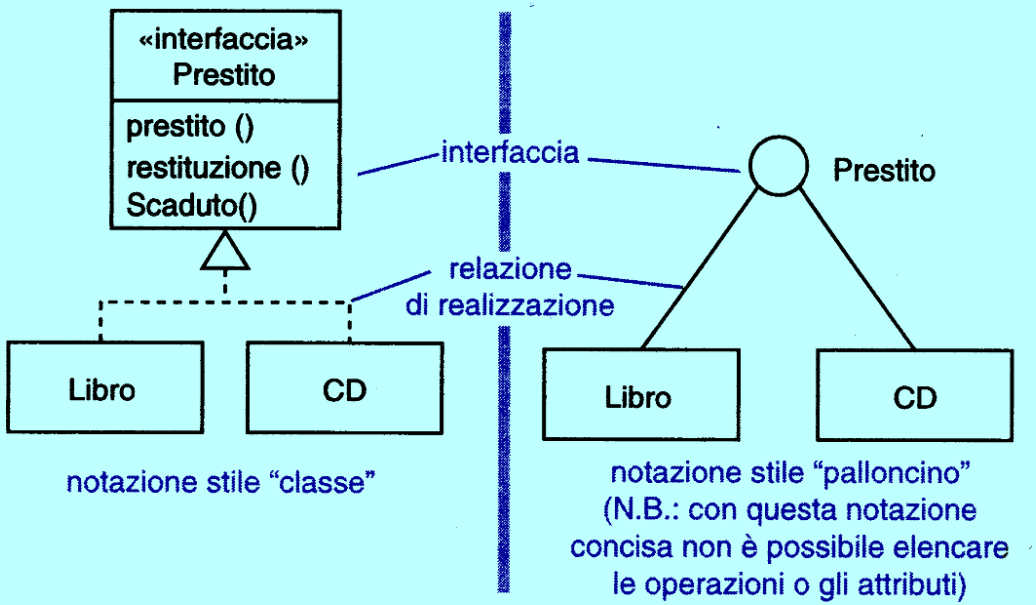
Interfacce

Un'interfaccia è un insieme di funzionalità pubbliche identificate da un nome.

- permette di separare le *specifiche* di una funzionalità (l'interfaccia, il "che cosa") dall'implementazione (il "come" la realizzazione deve avere luogo)
- definisce un contratto che può essere realizzato da zero o più classificatori
- *un'interfaccia non può essere istanziata*.

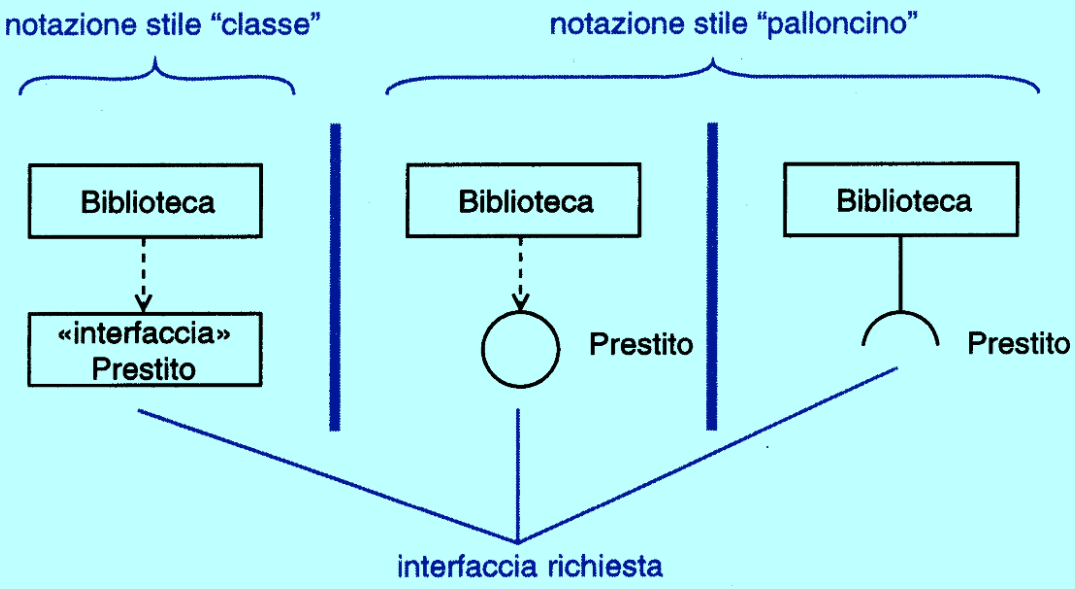
Interfacce fornite

- Le interfacce realizzate da un classificatore sono note come interfacce fornite
- La relazione di realizzazione è quella che intercorre tra una specifica e ciò che realizza la specifica (classi, package, componenti)



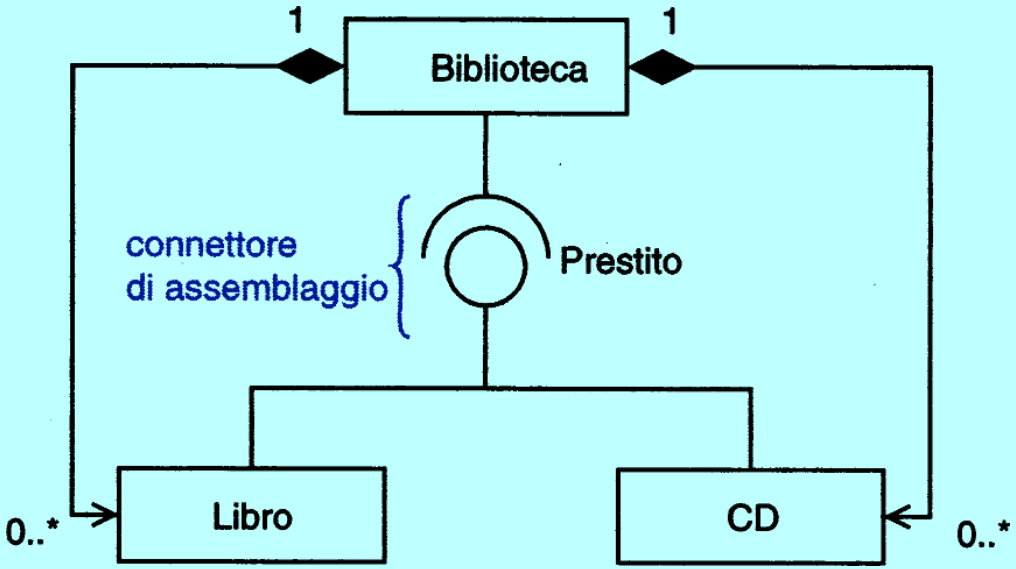
Interfacce richieste

- Le interfacce richieste sono quelle necessarie ad un classificatore per il suo funzionamento
- La notazione grafica a palloncino (*lollipop*) *non* elenca gli attributi e le operazioni di una interfaccia



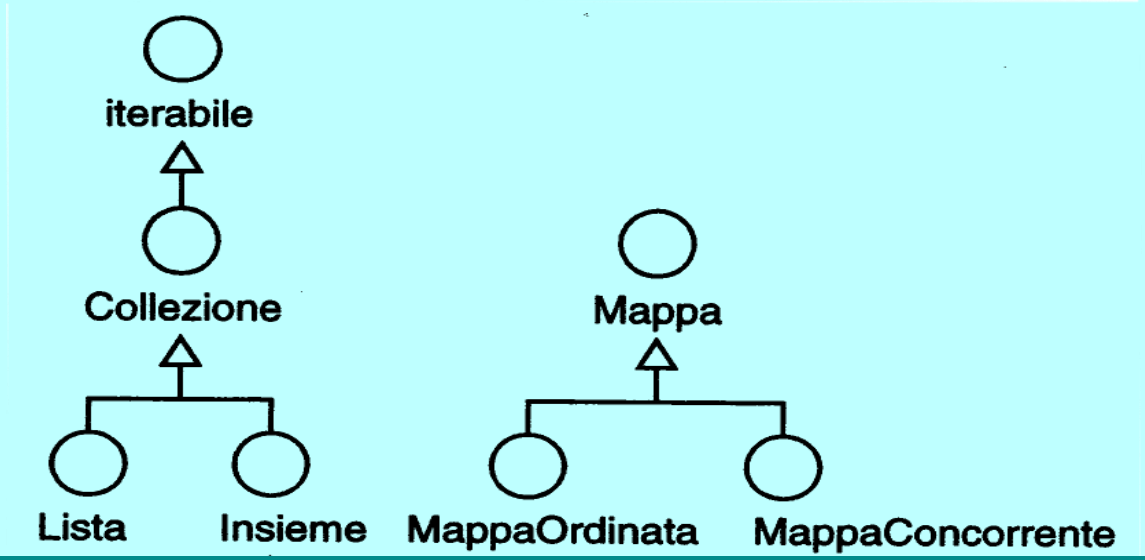
Composizione di interfacce

- Il connettore di assemblaggio permette di indicare chi fornisce o utilizza l'insieme dei servizi richiesti e offerti dai classificatori



Realizzazione di interfacce ed ereditarietà

- La semantica della realizzazione di interfaccia è "realizza il contratto specificato da", quella di ereditarietà è "è un"
- Poiché il principio di sostituibilità viene applicato sia all'ereditarietà sia alla realizzazione di interfaccia, entrambi i tipi di relazione possono generare polimorfismo
- Anche tra le interfacce possono esistere relazioni di ereditarietà



Realizzazione di interfacce ed ereditarietà

Ci sono due benefici nel manipolare oggetti in termini di interfacce [Design Patterns, GoF]:

1. I client non badano allo specifico tipo degli oggetti che usano, finché gli oggetti rispondono all'interfaccia che richiedono
2. I client rimangono ignari delle classi che implementano le interfacce. Essi devono conoscere solamente le interfacce.

Le interfacce consentono l'uso

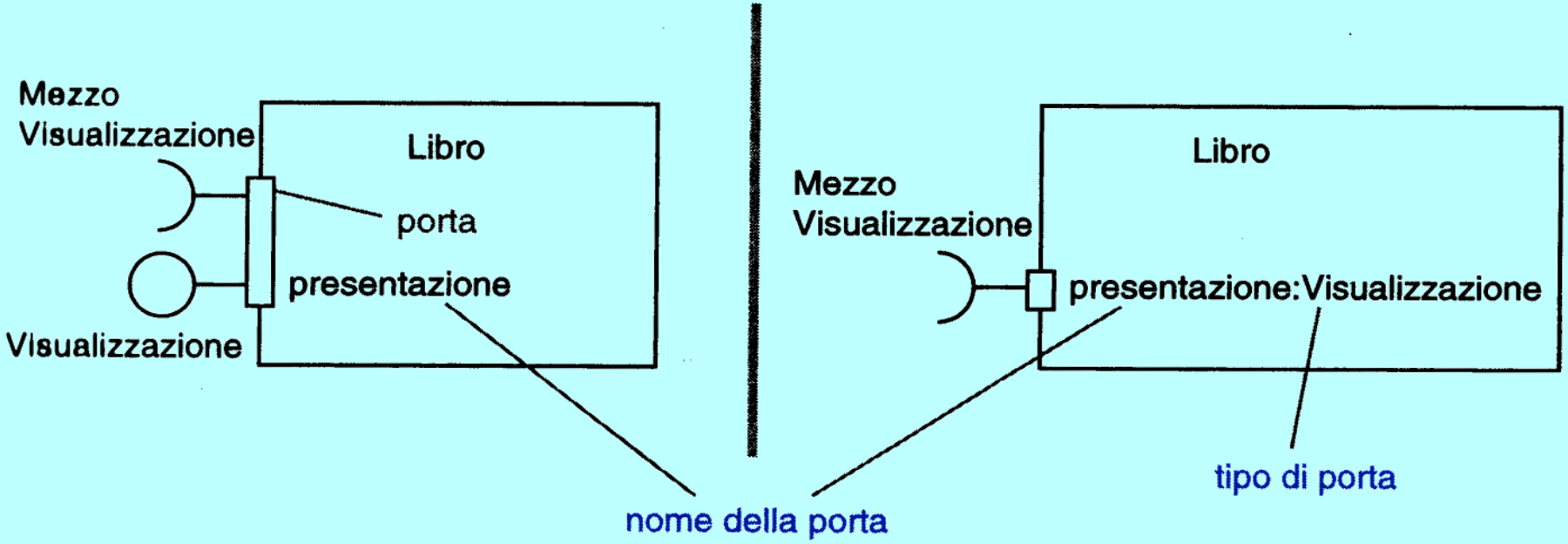
di un *qualunque numero* di diverse implementazioni,

a patto che ciascuna rispetti il contratto

Porte

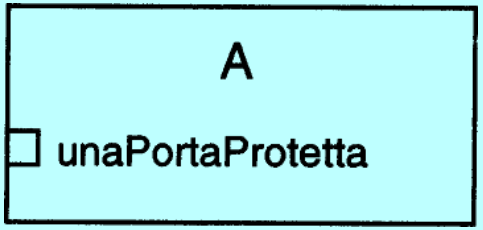
Una porta raggruppa un insieme semanticamente coeso di interfacce fornite e richieste.

- Forniscono un metodo utile per strutturare interfacce fornite e richieste di un classificatore



Porte

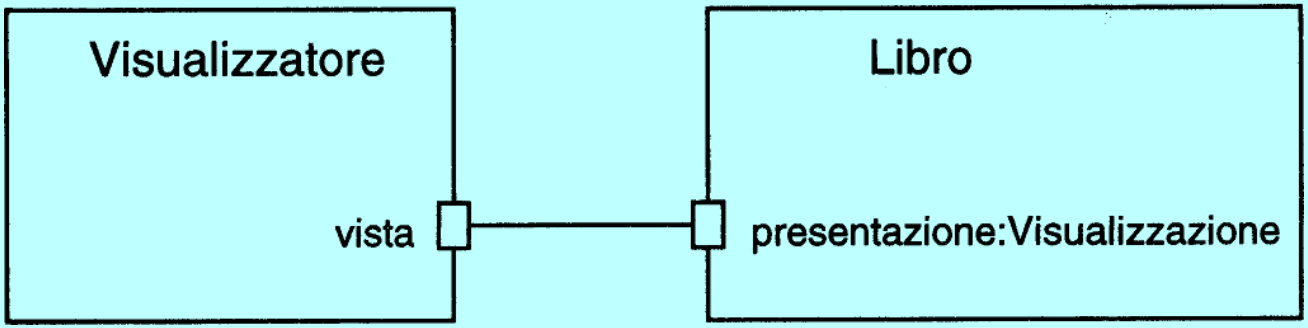
- Le porte possono avere una visibilità e una molteplicità:
  - quando la porta è disegnata sovrapposta al limite del classificatore è pubblica (le interfacce raggruppate sono pubbliche), altrimenti ha visibilità protetta (di default) o privata.



  - la molteplicità (specificata tra parentesi quadre [n] dopo il nome della porta e il nome del tipo) specifica il numero di istanze della porta che ciascuna istanza del classificatore avrà.

Porte

- Se le porte sono connesse le loro interfacce devono corrispondere



Classificatori strutturati

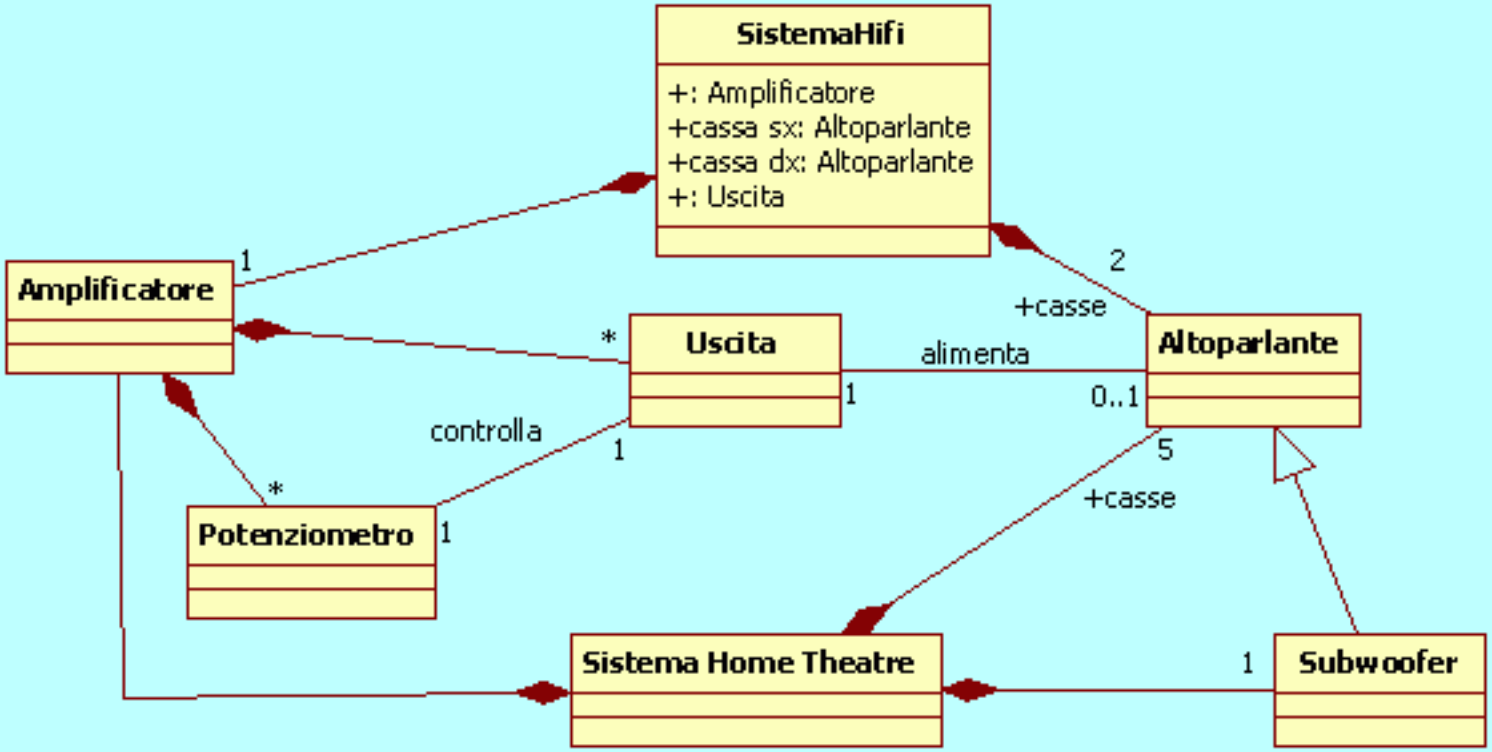
Un classificatore strutturato è un classificatore che ha una struttura interna

- La struttura interna è rappresentata mediante le **parti** e le loro unioni con gli altri elementi mediante **connettori**
  - Una **parte** è un **ruolo** che una o più istanze di un classificatore possono ricoprire nel contesto del classificatore strutturato
    - Le parti **non** rappresentano delle classi
  - Un **connettore** è una relazione tra le parti nel contesto del classificatore strutturato
    - I connettori indicano che le istanze che ricoprono i ruoli specificati dalle parti possono in qualche modo comunicare

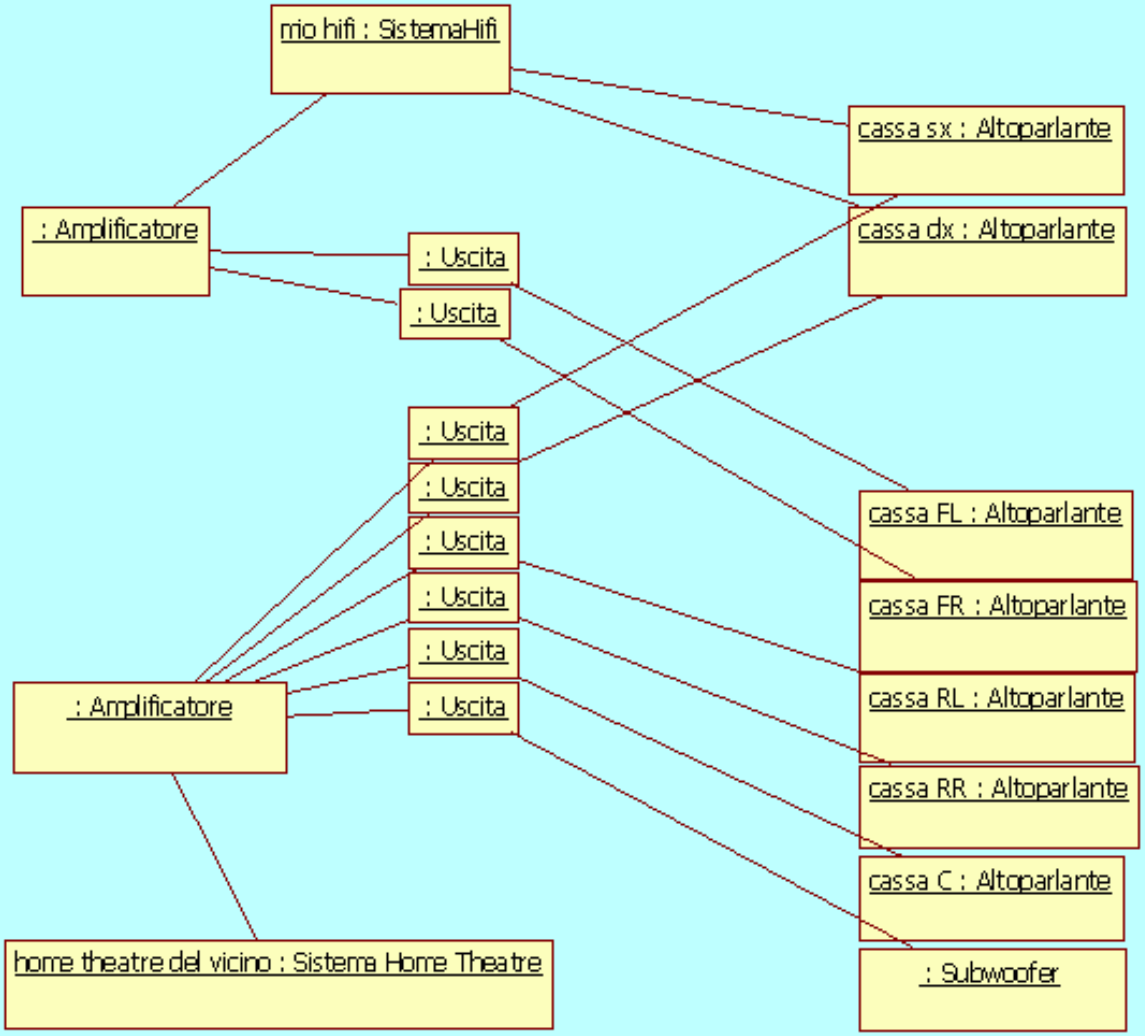
Classificatori strutturati

- Le classi strutturate e i componenti sono classificatori strutturati
- Tipicamente si evidenziano con i classificatori strutturati alcuni dettagli relativi all'implementazione interna o all'interfaccia esterna
  - L'interazione di un classificatore strutturato con il suo ambiente viene modellata dalle sue interfacce e dalle sue porte

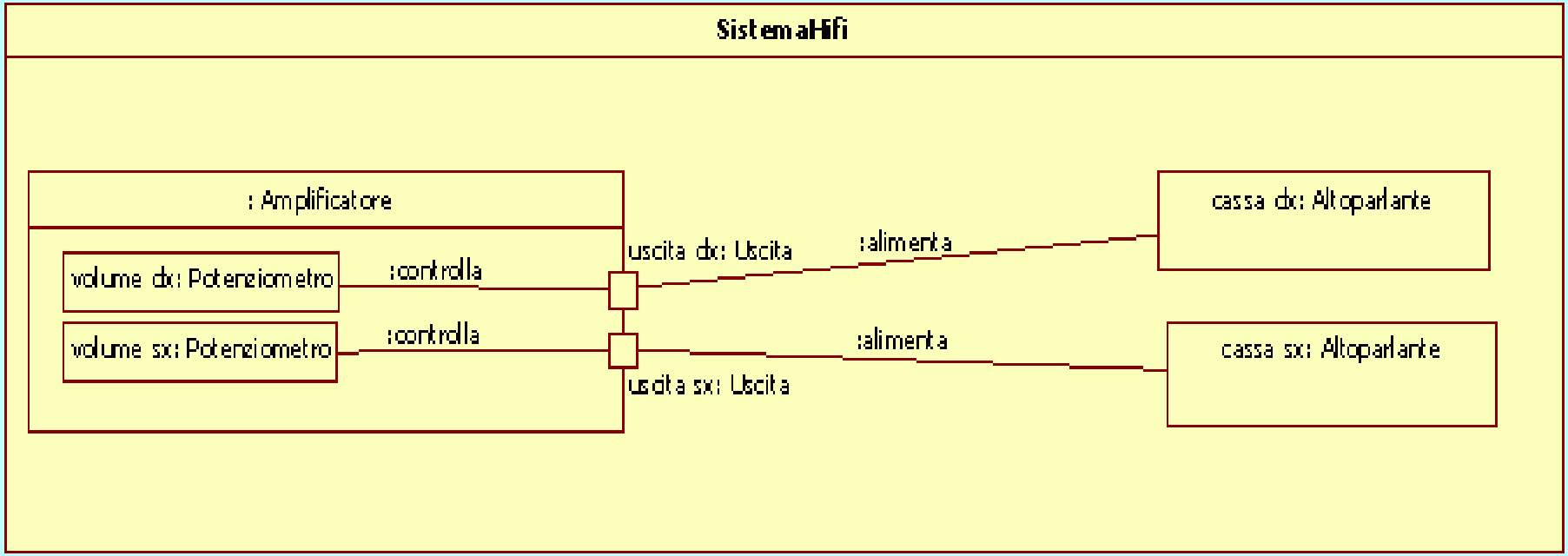
Classificatori strutturati: esempio (1/3)



Classificatori strutturati: esempio (2/3)

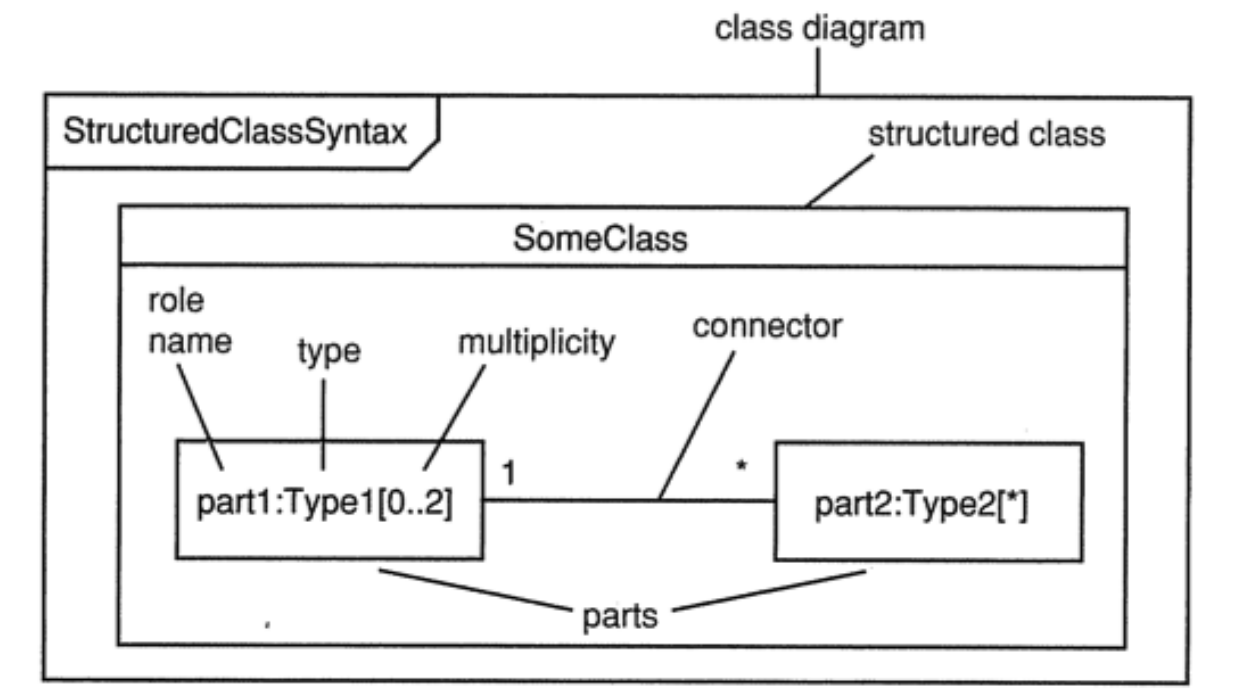


Classificatori strutturati: esempio (3/3)



Classe strutturata

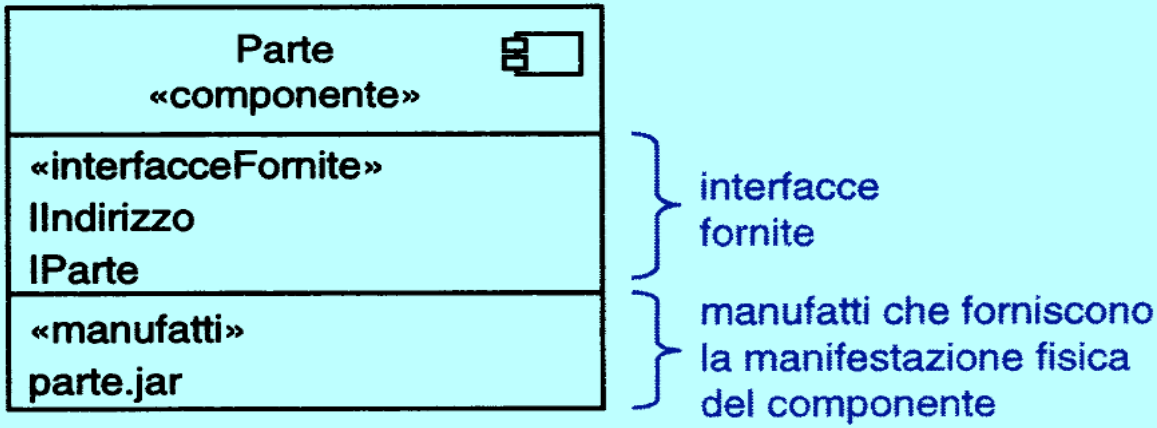
- Una classe strutturata ha una relazione implicita di **contenimento** di tutte le sue parti
  - Se la parte è disegnata con una linea tratteggiata, si specifica che la parte non è contenuta nella classe strutturata



Componenti

Un componente rappresenta una parte modulare di un sistema che incapsula i suoi contenuti e la cui manifestazione è sostituibile all'interno del suo ambiente

- Fungono da black-box il cui comportamento esterno è definito dalle sue interfacce fornite e richieste
- Possono essere resi manifesti da uno o più manufatti, cioè elementi del mondo fisico, come file di codice sorgenti o JAR



Componenti

I componenti:

- Sono classificatori strutturati e la la loro struttura interna può comprendere parti e connettori.
- Possono avere attributi e operazioni e possono partecipare a relazioni di associazione e di generalizzazione.
- Possono rappresentare qualcosa di instanziato a tempo d'esecuzione, oppure costrutti puramenti logici, instanziati indirettamente in virtù del fatto che le parti vengono instanziate
- La nozione di componente è alla base del component-based development (CBD)

Diagramma dei componenti

Un diagramma dei componenti può mostrare i componenti, le dipendenze tra di essi e il modo in cui i classificatori vengono assegnati ad altri componenti

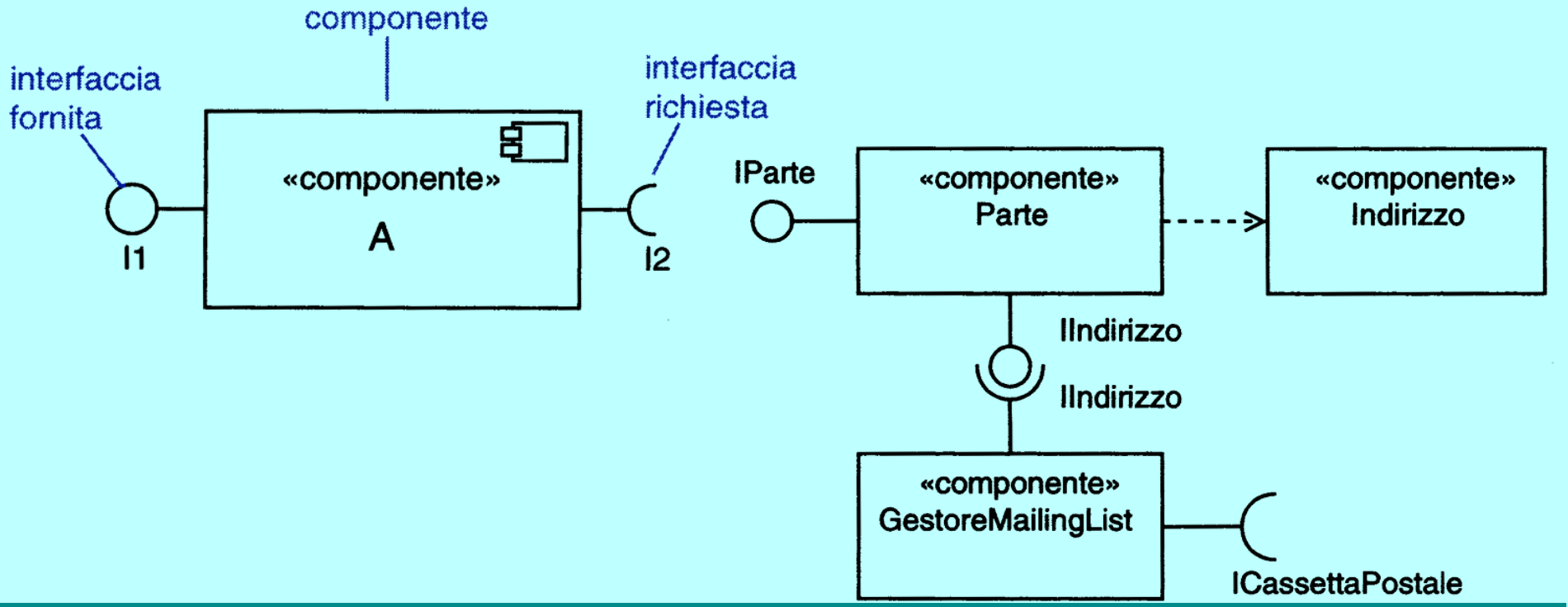


Diagramma dei componenti

- Essendo i componenti classificatori strutturati, è possibile mostrare le parti annidate all'interno del componente
  - Un componente di solito delega le responsabilità definite alle sue interfacce a una o più parti interne
- I componenti possono dipendere da altri componenti solo per mediazione di interfacce

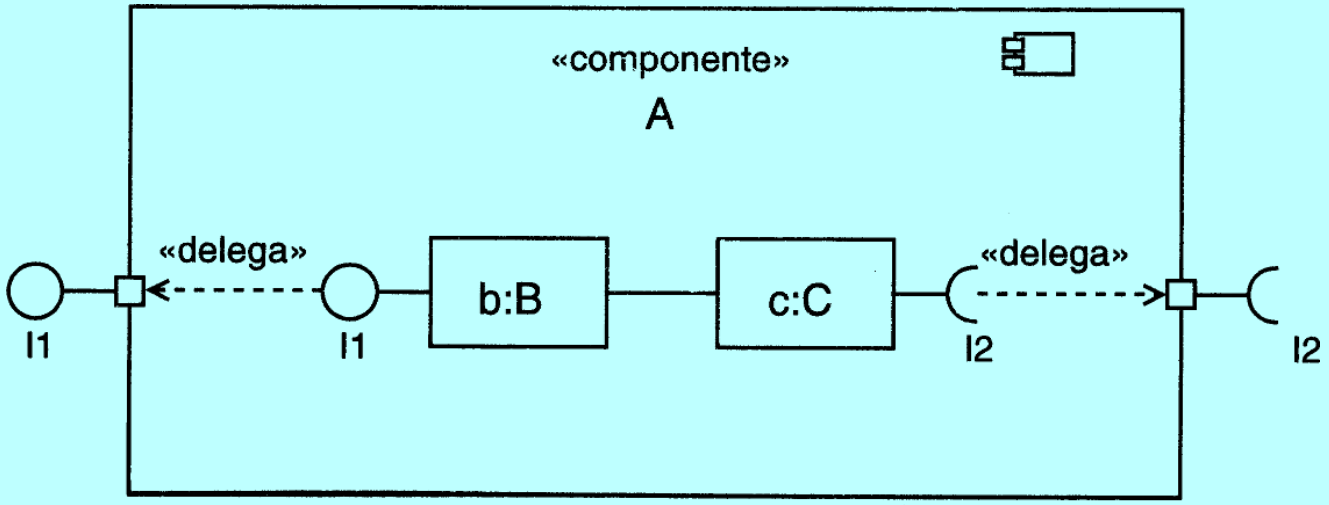
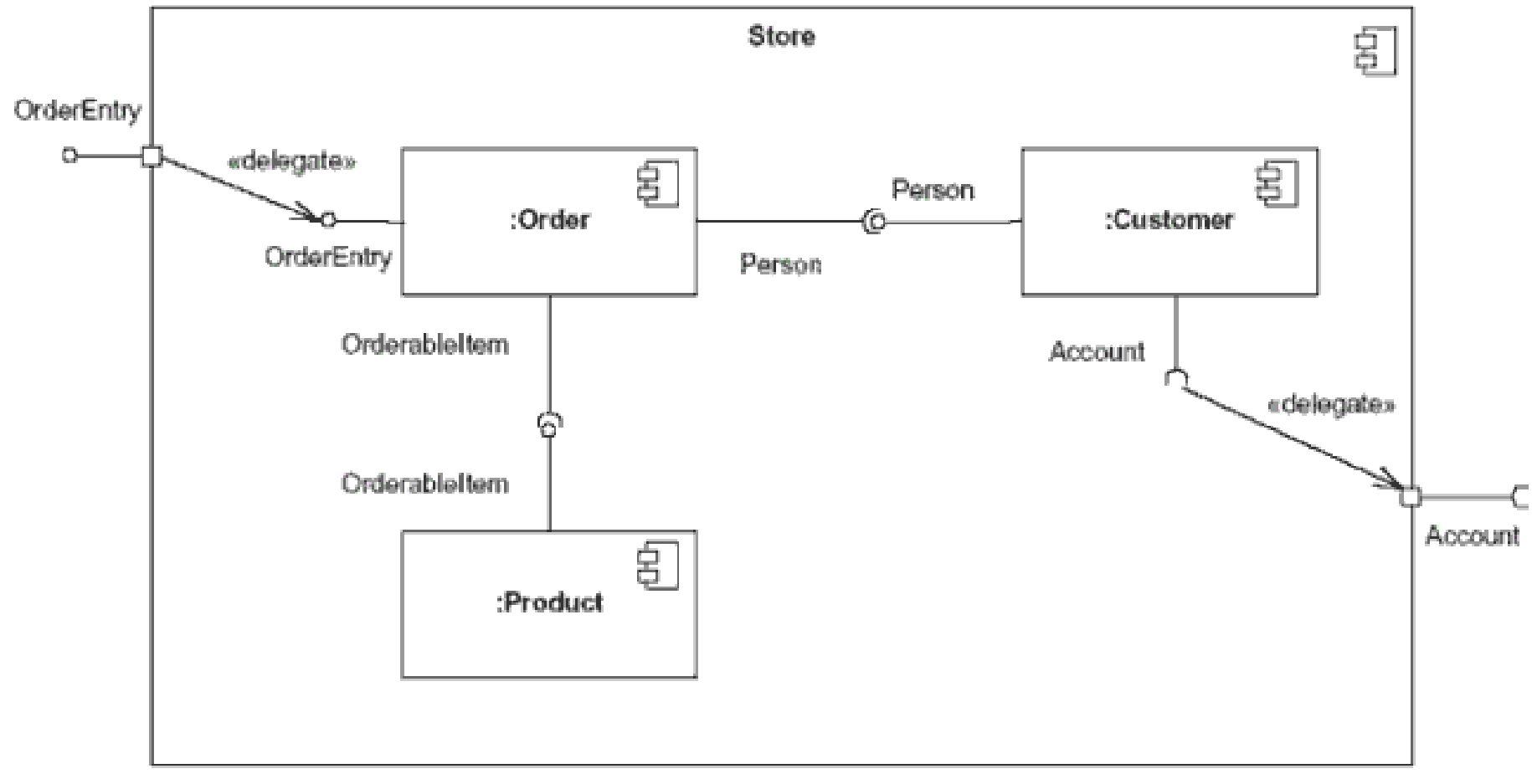


Diagramma dei componenti



Stereotipi dei componenti

- UML fornisce un insieme di stereotipi standard dei componenti

Stereotipo	Significato
«buildComponent»	Un componente che definisce un insieme di cose per scopi organizzativi o di sviluppo a livello di sistema.
«entity»	Un componente informativo persistente che rappresenta un concetto di business.
«implementation»	Una definizione di componente che non ha specifica; è un'implementazione per una «specifica» separata da cui dipende.
«specification»	Un classificatore che specifica un dominio di oggetti senza definire l'implementazione fisica di quegli oggetti; per esempio, un componente con stereotipo specifico ha solo interfacce fomite e richieste e nessun classificatore di realizzazione.
«process»	Un componente basato su transizioni.
«service»	Un componente funzionale senza stato che calcola un valore.
«subsystem»	Un'unità di scomposizione gerarchica per grandi sistemi.

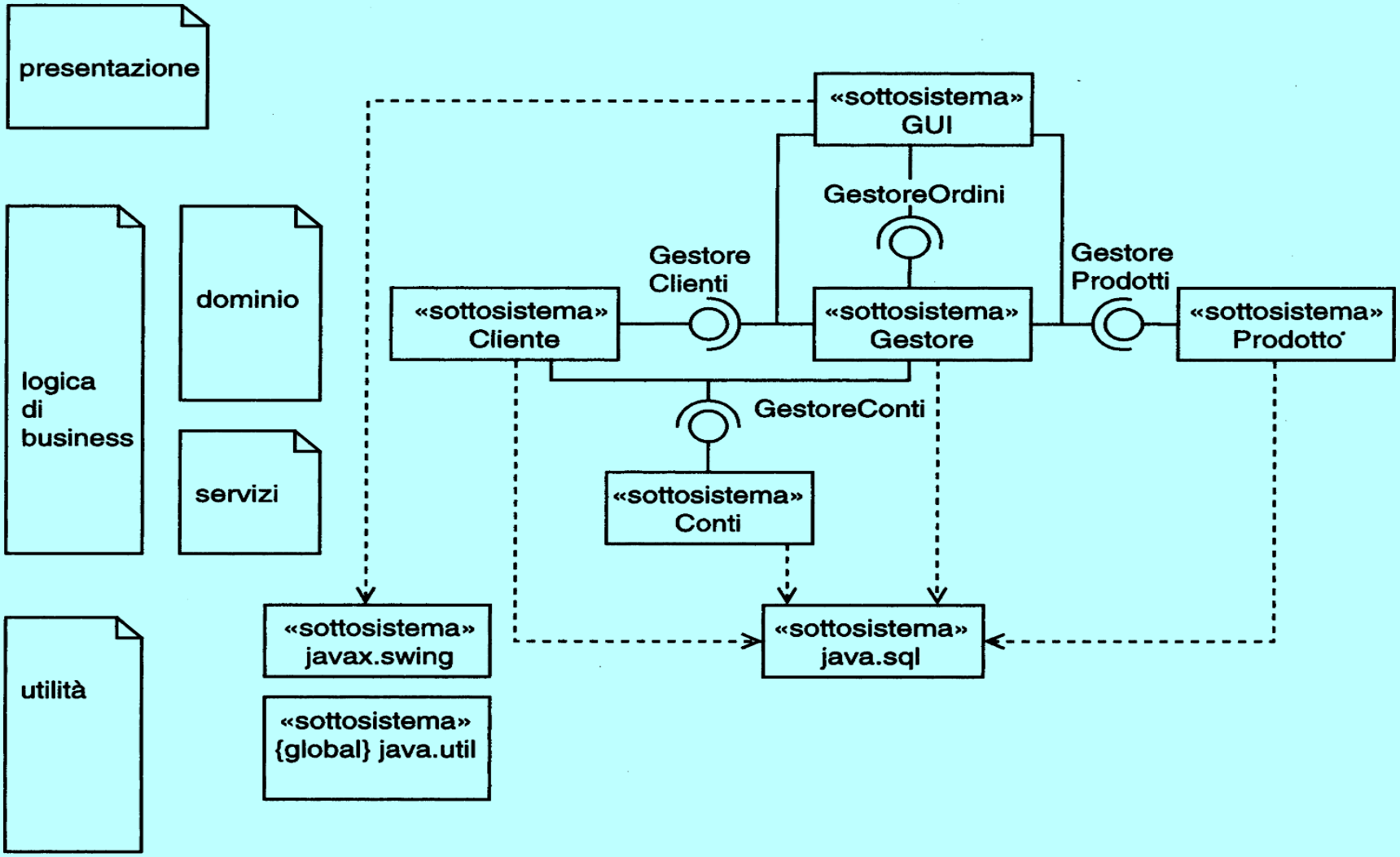
Sottosistemi

Un sottosistema è un componente che agisce come unità di scomposizione per un sistema più grande

- Non possono essere instanziati a tempo di esecuzione, a differenza dei loro contenuti

La decomposizione in sottosistemi permette di suddividere il problema di sviluppo in parti più piccole favorendo la flessibilità architetturale e avendo controllo sulle interdipendenze tra le parti di un sistema

Sottosistemi



Componenti e Package

I componenti (e i *«subsystem»*) sono simili ai package perché definiscono i confini per raggruppare elementi in strutture. Tuttavia:

- I package sono un meccanismo di raggruppamento logico che fornisce uno spazio di nomi per i suoi membri
  - I package sono usati per organizzare il modello
  - Un package esiste a tempo di sviluppo
- I componenti raggruppano gli elementi del modello in membri che avranno una istanziazione in esecuzione
  - I component sono utilizzati per organizzare la soluzione
  - Un componente esiste a runtime, ha responsabilità, comportamenti ed interfacce ben definite che devono funzionare a tempo di esecuzione

Componenti e Artifact

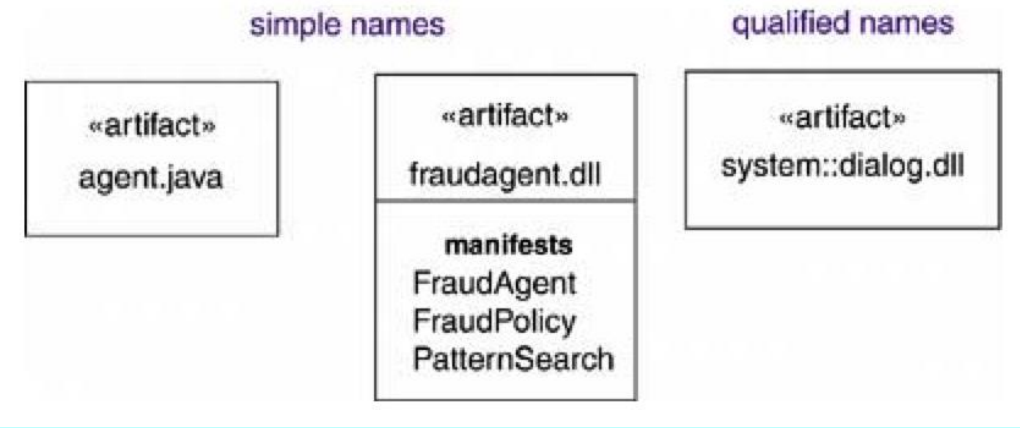
- Un componente è una unità a livello architetturale che può rappresentare:
  - Unità della logica applicativa (business software architecture)
  - Unità tecniche (technical architecture) del dominio della soluzione (per es. un componente *database*, un componente responsabile della sicurezza)
  - Entrambe
  - Non rappresenta unità fisiche

- Un componente può essere implementato da uno o più artifact (manufatti)
  - **ne costituisce una realizzazione fisica**

Artifact

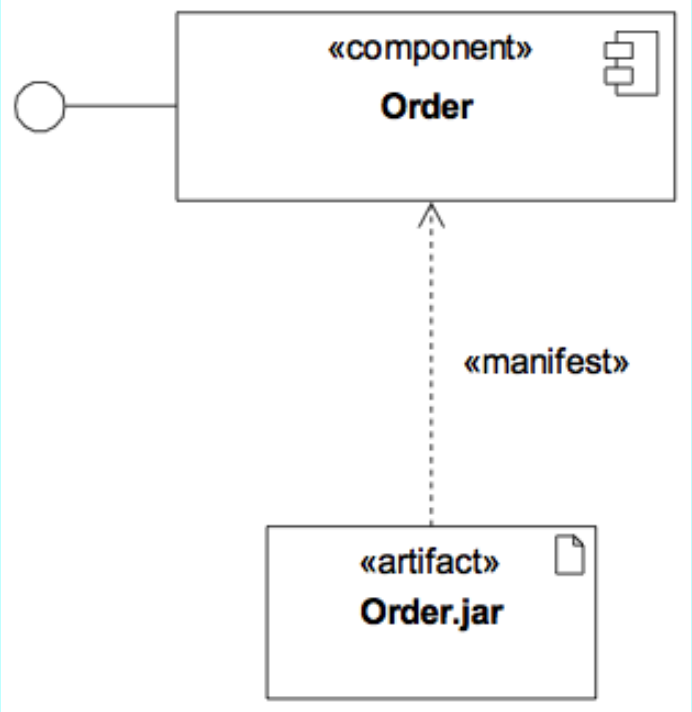
Un artifact (manufatto) rappresenta un'entità concreta del mondo reale (file, tabelle DB, documenti, modelli UML, …)

- Un manufatto può offrire una manifestazione fisica di qualsiasi tipo di elemento UML
- I manufatti possono dipendere da altri manufatti
- Un'istanza di manufatto rappresenta un'istanza specifica di un particolare manufatto
  - Essi hanno un nome che fa riferimento alla collocazione fisica dell'istanza



Artifact

- Un manufatto che è manifestazione di un elemento:
  - è connesso ad esso tramite contenimento fisico oppure con una relazione «manifest»



Stereotipi dei manufatti

- UML fornisce un insieme di stereotipi standard degli artifact

Stereotipo	Significato
«file»	Un file fisico.
«deployment spec»	Una specifica dei dati di deployment (per esempio un file manifest.xml).
«document»	Un file generico che contiene alcune informazioni.
«executable»	Un file eseguibile di programma.
«library»	Una libreria statica o dinamica quale un file JAR (Java ARchive) oppure DLL (Dynamic Link Library).
«script»	Uno script che può essere eseguito da un interprete.
«source»	Un file di codice sorgente che può essere compilato in un file eseguibile.

Diagramma di Deployment

Un diagramma di allocazione (deployment) descrive le scelte d'allocazione dei componenti software sviluppati (siano essi sottosistemi, driver) sulle unità fisiche di elaborazione

- Esso può essere utilizzato per una descrizione dell'architettura fisica e della distribuzione del sistema (hardware e software)
- Nella pratica non si modella un deployment esaustivo ma solamente gli elementi architetturalmente significativi
  - Un sistema di tipo cliente-servente, ad esempio, sarà fisicamente realizzato come una serie di moduli software riguardanti la parte server e le parti client; tali moduli dovranno essere chiaramente dislocati su unità d'elaborazione diverse.

Diagramma di Deployment: Nodi

Un nodo rappresenta un tipo di risorsa computazionale (es.: PC IBM, smartphone) su cui i manufatti (artifact) possono essere dislocati per l'esecuzione

- Un nodo è un classificatore strutturato e può avere annidati altri nodi ed elementi
- Un'associazione tra nodi rappresenta un canale di comunicazione attraverso cui possono passare le informazioni

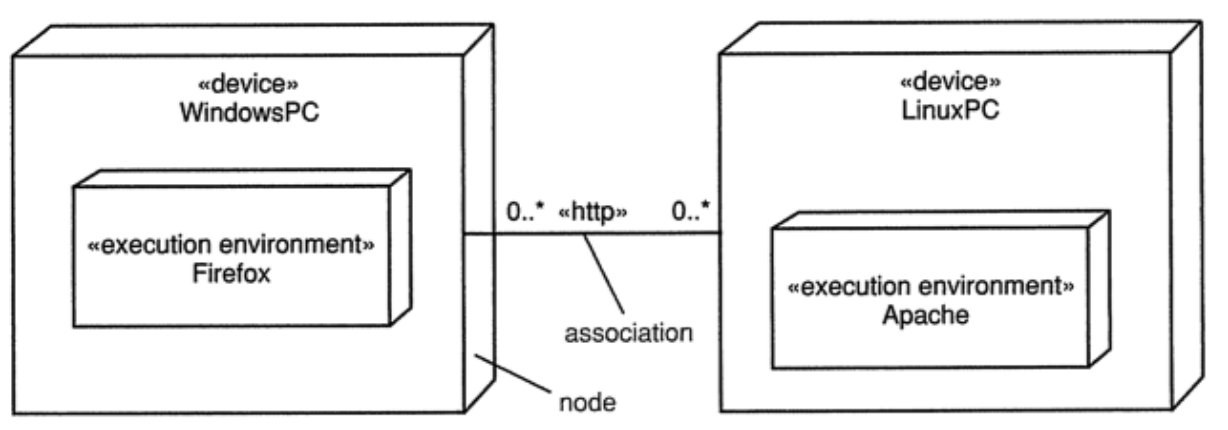
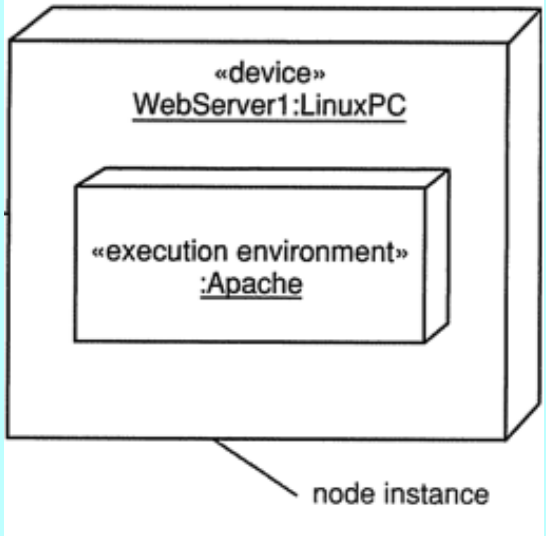


Diagramma di Deployment: Nodi

- Una istanza di nodo rappresenta un hardware specifico ed identificabile (es.: PC della cassa 3)
- Una istanza di manufatto è una specifica istanza di un tipo software (es.: versione 1.0.0 file system.exe)
- Il deployment è il processo di assegnazione di artifact a nodi o di istanze di artifact a istanze di nodi



Stereotipi dei Nodi

- UML fornisce un insieme di stereotipi standard dei Nodi

Stereotipo	Significato
«device» («periferica»)	Il nodo rappresenta un tipo di periferica fisica, per esempio un PC o un server Sun Fire
«execution environment»	Il nodo rappresenta un tipo di ambiente software di esecuzione quale un server web Apache o una Java Virtual Machine

Diagramma di Deployment

Il diagramma di deployment fa corrispondere l'architettura software creata nella progettazione a un'architettura del sistema fisico che lo esegue.

- Nei sistemi distribuiti modella la distribuzione del software sui nodi fisici.

Esistono due forme di diagramma di deployment

1. Forma di descrittore: contiene nodi, relazioni tra nodi, e manufatti.
2. Forma istanza: contiene istanze di nodo, relazioni tra istanze di nodo, e istanze di manufatto.

Diagramma di Deployment

I diagrammi di deployment in forma di descrittore:

- Sono utili per modellare un tipo di architettura fisica
- Possono essere prodotti in progettazione quando occorre decidere l'architettura hardware del sistema

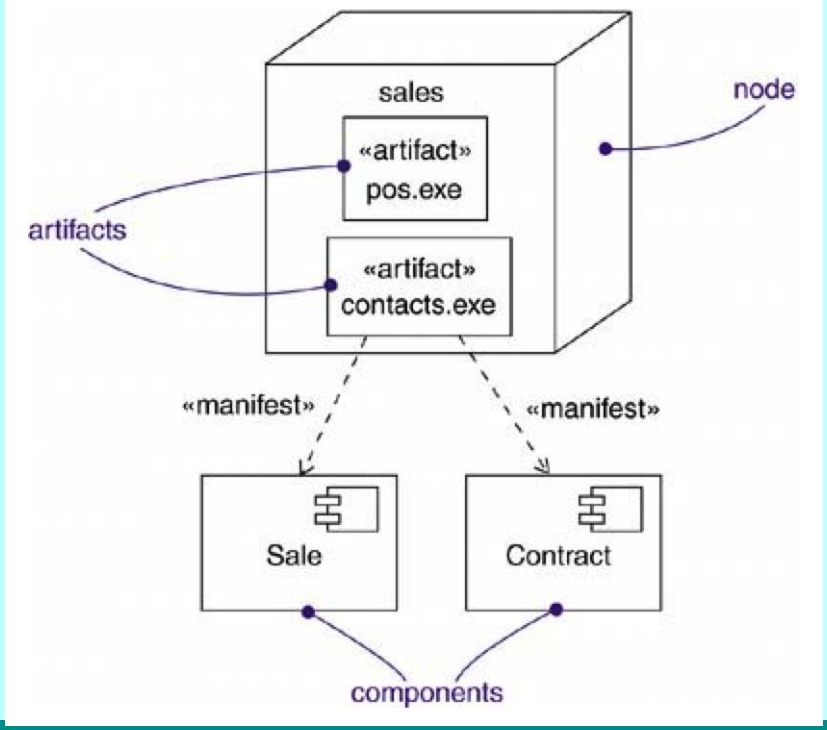
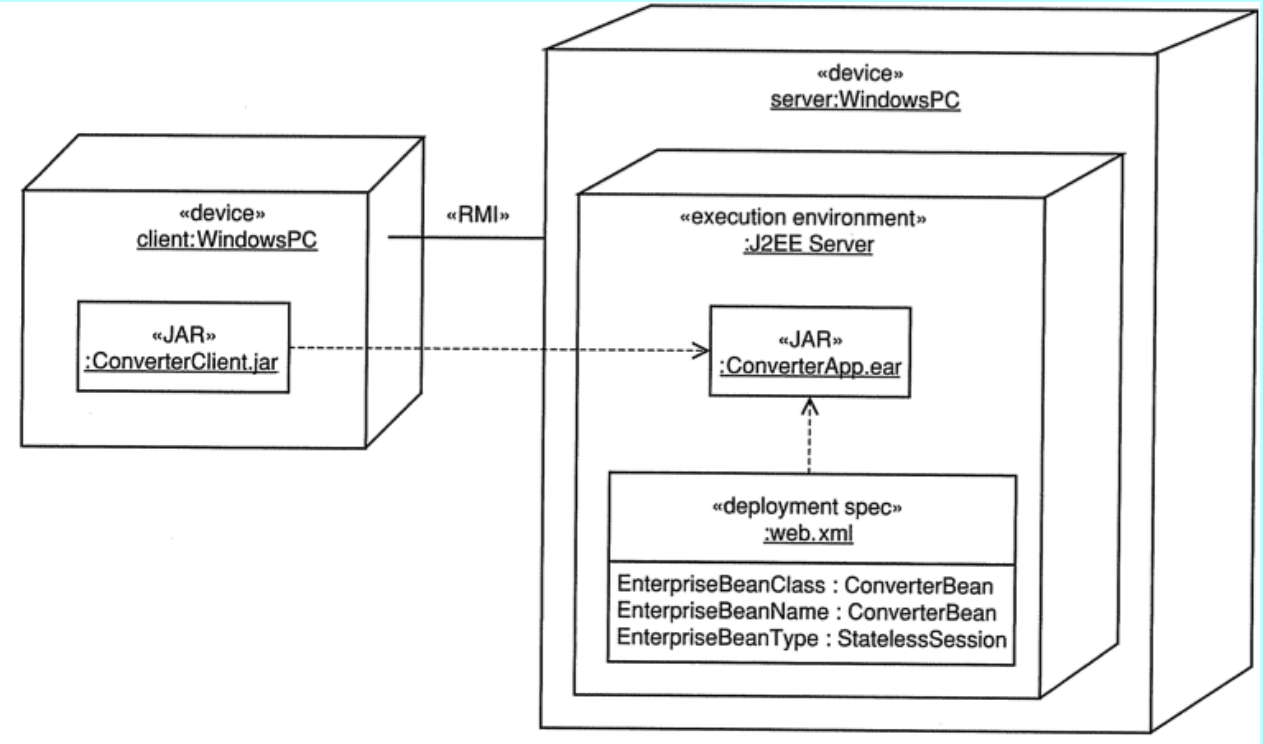


Diagramma di Deployment

I diagrammi di deployment in forma di istanza:

- Sono utili per modellare un deployment in un particolare sito
- mostra la configurazione del sistema a runtime: i componenti software, processi ed oggetti implicati nell'esecuzione.



Deployment Diagram

- I componenti che non esistono a tempo di esecuzione non compaiono in tale diagramma (potrebbero essere mostrati nel *component diagram*).
- I manufatti possono manifestare istanze di componenti ed essere allocati su nodi (ciò indica che i componenti vivono o girano sul nodo in questione).
- I componenti possono contenere oggetti (il che indica che l'oggetto è parte del componente)

Deployment Diagram

- I componenti possono inoltre essere collegati tra loro, il che significa che un componente usa i servizi di un altro componente.

- Si può anche rappresentare la migrazione di componenti fra nodi o di oggetti fra componenti.

Tali diagrammi possono essere utilizzati (ad un elevato livello d'astrazione) anche in fase di analisi per la specifica del sistema complessivo in termini dei vincoli fisici (hardware esistente) e/o naturali (tipologia dell'applicazione).