Ingegneria del Software
13b. Viste

Dipartimento di Informatica
Università di Pisa
A.A. 2014/15

viste in AS

- Una vista è la rappresentazione di un insieme di elementi del sistema e delle loro proprietà e relazioni
- Una vista espone differenti attributi di qualità in gradi differenti
  - una vista a livelli espone la portabilità
  - una vista di “deployment” espone le prestazioni e l’affidabilità
- Ciascuna vista astrae da informazioni irrilevanti per quel punto di vista

documentare una AS

- Documentazione di ciascuna vista rilevante
  - visione complessiva, spesso grafica
  - catalogo degli elementi
  - la specifica delle interfacce e del comportamento degli elementi
  - razionale delle scelte
- Documentazione globale
  - introduzione e guida alla documentazione
  - relazioni tra le viste
  - razionale e vincoli globali
- La rilevanza dipende dagli obiettivi

tipo di vista (viewtype)

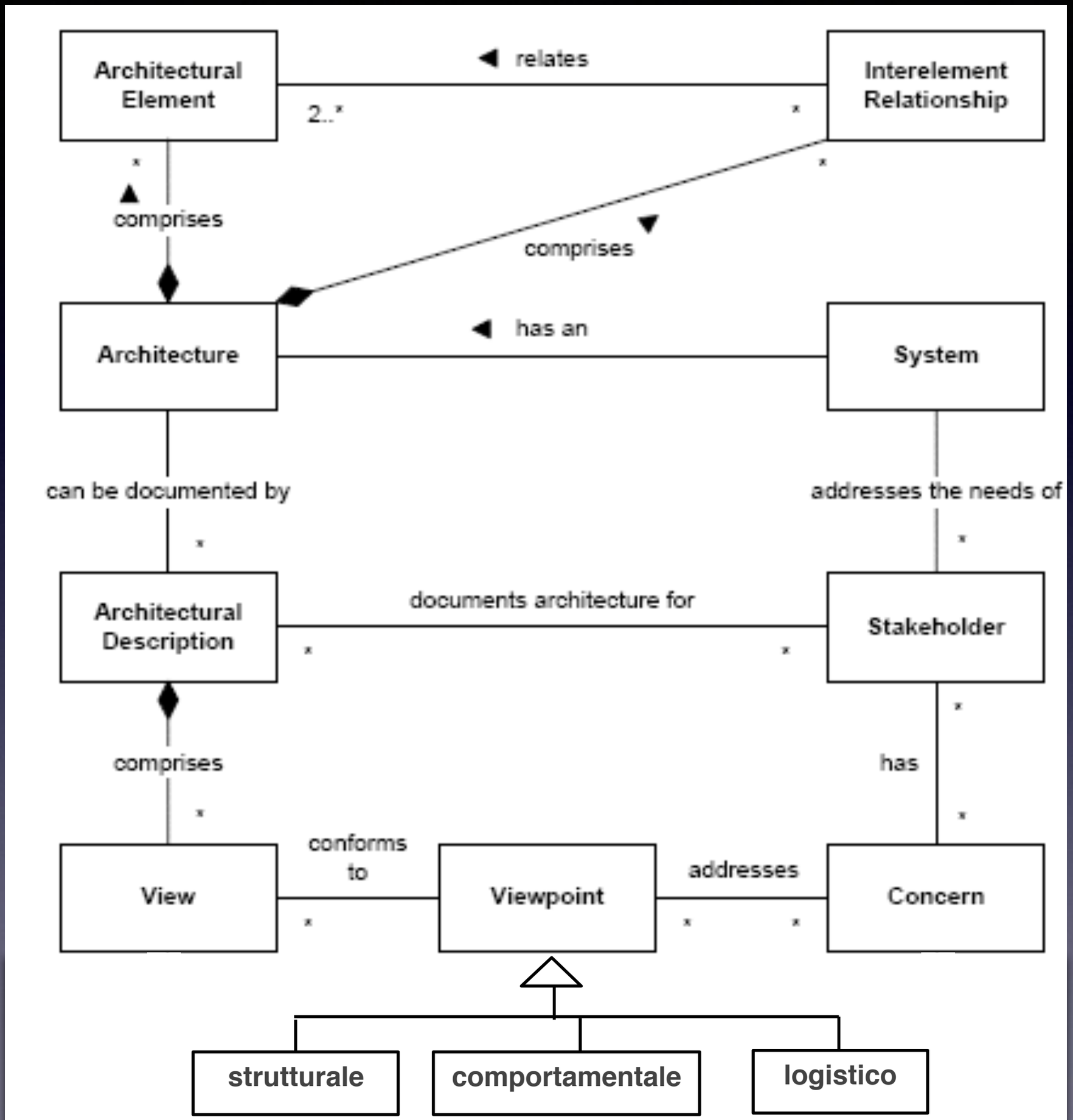
- Un tipo di vista definisce i tipi degli elementi e delle relazioni usati per descrivere l'AS da un particolare punto di vista
- Tre punti di vista simultanei sul sistema
  - la struttura come insieme di unità di realizzazione - strutturale
  - la struttura come insieme di unità con comportamenti e interazioni, a run-time - comportamentale (component-and-connector, C&C)
  - le relazioni con strutture diverse dal software nel contesto del sistema - logistico (problemi di allocazione)

stili (o schemi)

- Uno stile è una proprietà di una architettura
- Uno stile caratterizza una famiglia di architetture che soddisfano i vincoli
  - stile a livelli: vincola quale modulo può usare quale altro
  - stile client-server: vincola le interazioni tra componenti
- Documentati in una guida di stile
- Di uso comune

organizzazione

- La definizione di ciascun tipo di vista e di ciascuna vista è strutturata come segue
  - elementi e relazioni
  - loro proprietà
  - usi (utilità della vista)
  - notazione (costrutti UML utilizzati)


Architectural Element
relates
2..*
*
Interelement Relationship
*
*
comprises
comprises
Architecture
has an
System
can be documented by
*
addresses the needs of
*
Architectural Description
documents architecture for
*
*
Stakeholder
comprises
*
has
*
View
conforms to
*
Viewpoint
addresses
*
*
Concern
strutturale
comportamentale
logistico

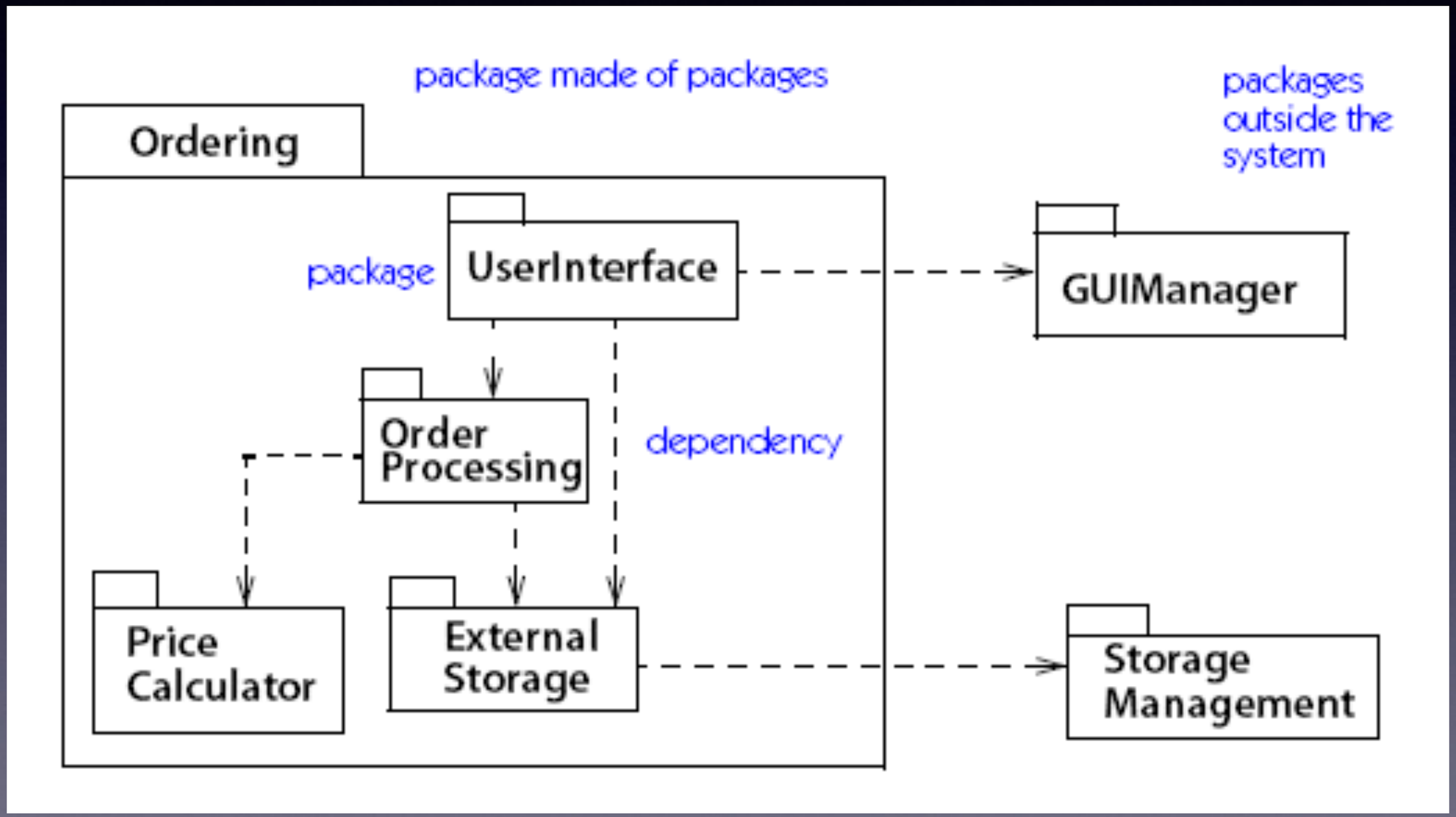
viste strutturali

- Elementi
  - modulo: unità di software che realizza un insieme coerente di responsabilità
    - classe, package (collezione di classi, un livello)
- Relazioni
  - parte di, eredita da, dipende da, può usare
- Proprietà
  - responsabilità, visibilità, autore,
  - informazioni sulla realizzazione
    - codice associato, informazioni sul test e gestionali, vincoli sulla realizzazione…

viste strutturali, 2

- Usi: costruzione
  - la vista può fornire lo schema del codice se directory e file sorgente hanno struttura corrispondente
- Usi: analisi
  - tracciabilità dei requisiti
  - analisi d'impatto per valutare eventuali modifiche
- Usi: comunicazione
  - se la vista è gerarchica, offre una presentazione top-down della suddivisione delle responsabilità nel sistema ai novizi
- Non usi: analisi dinamiche
  - cfr. viste comportamentali e logistiche
- Notazione: classi e package + per relazioni

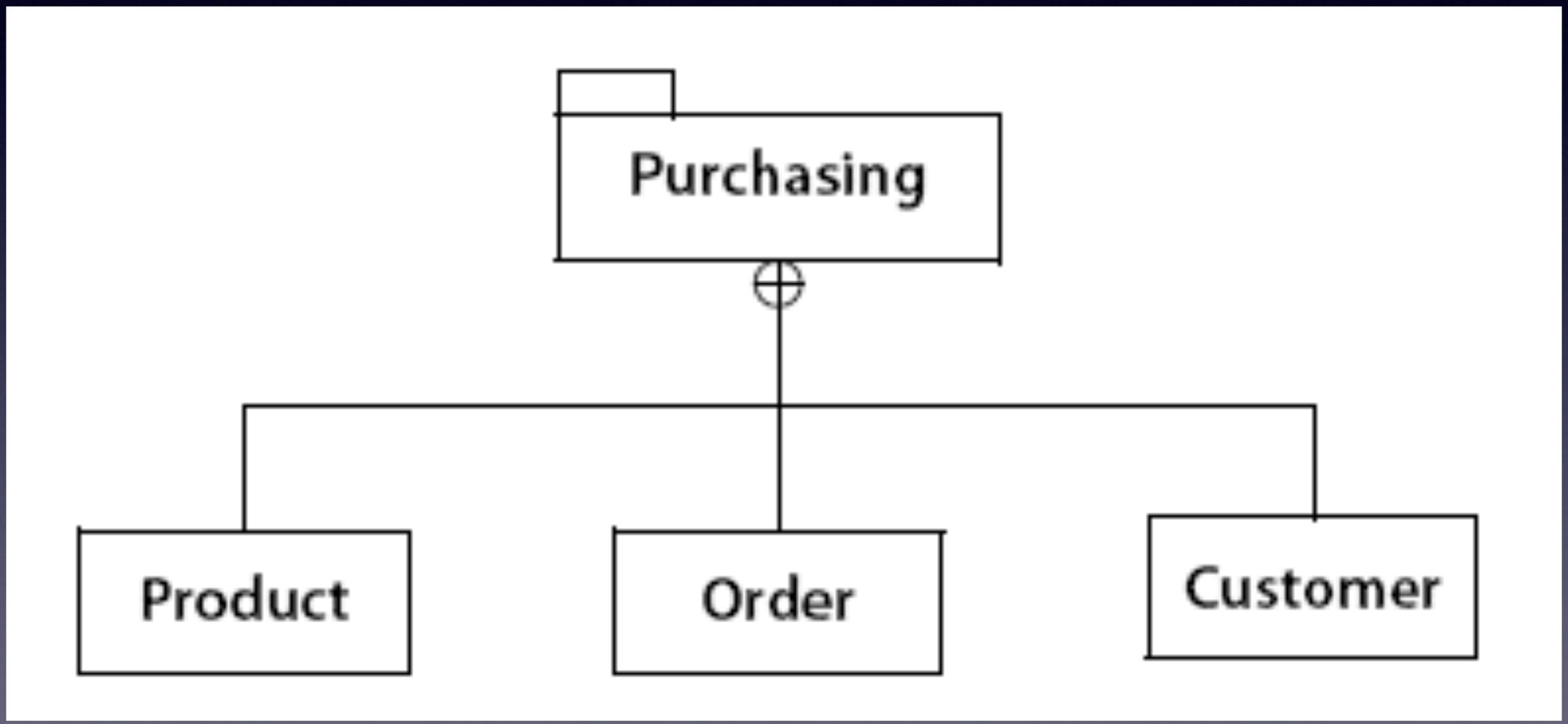
notazione: package



vista strutturale di decomposizione

- Relazione “parte di”
  - moduli e sottomoduli
  - applica l’approccio divide-and-conquer
- Usi
  - apprendimento del sistema
  - punto di partenza per l’allocazione del lavoro
- Criteri
  - incapsulamento per modificabilità
  - supporto alle scelte costruisci/compra
  - moduli comuni in linee di prodotto

notazione esterna (contenimento)

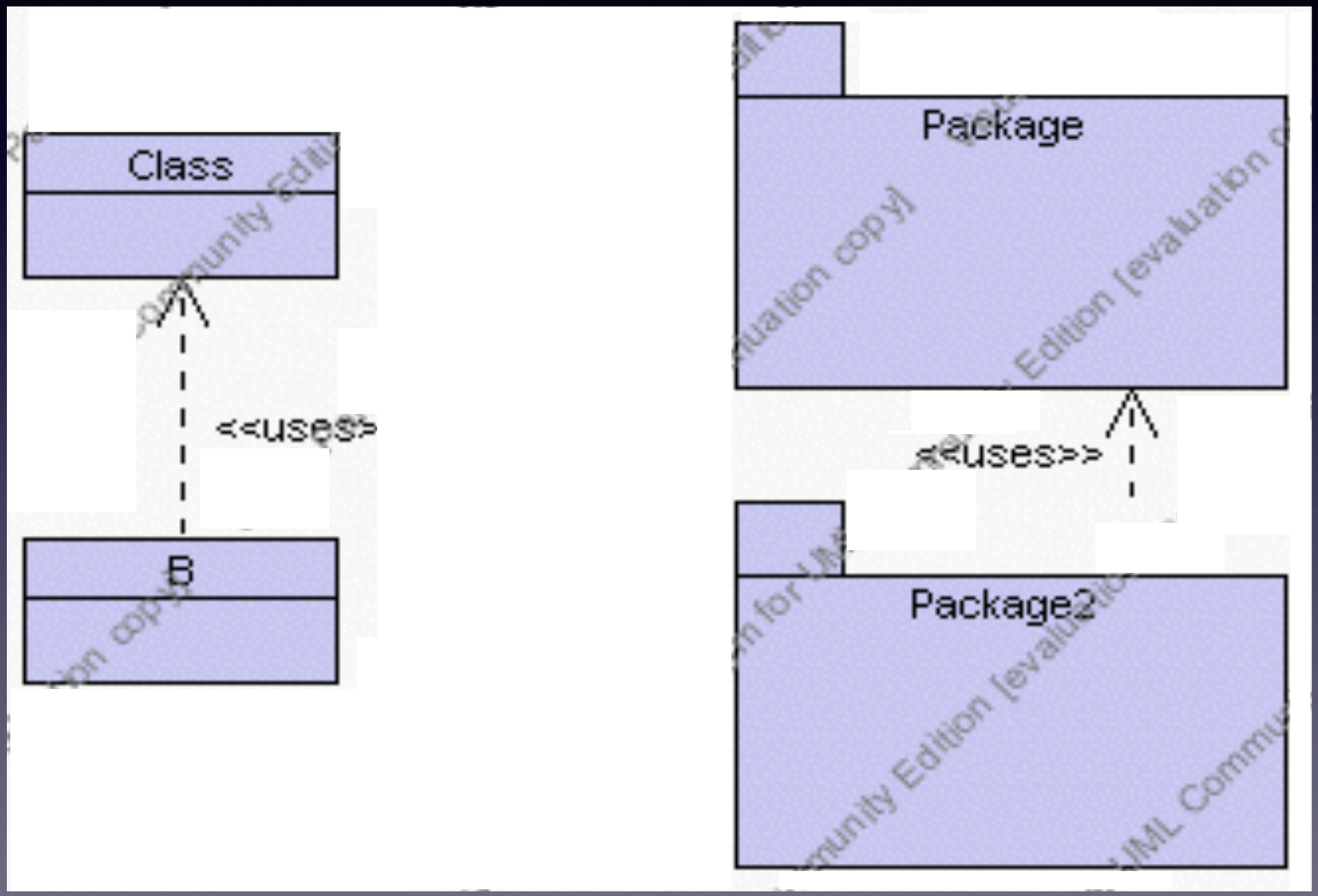


vista strutturale d’uso

- Relazione “usa”
  - il modulo A usa il modulo B se dipende dalla presenza di B (funzionante correttamente) per soddisfare i suoi requisiti
  - abilita lo sviluppo incrementale e il rilascio di sottosistemi utili
  - cicli permessi ma pericolosi
- Usi
  - pianificazione di sviluppo incrementale, di estensioni e di ritagli del sistema del testing incrementale
  - analisi d’impatto

NB. un modulo che segnala un errore funziona correttamente indipendentemente da cosa fa il modulo che riceve la segnalazione [sorta di comunicazione asincrona: invocazione non è uso]

notazione



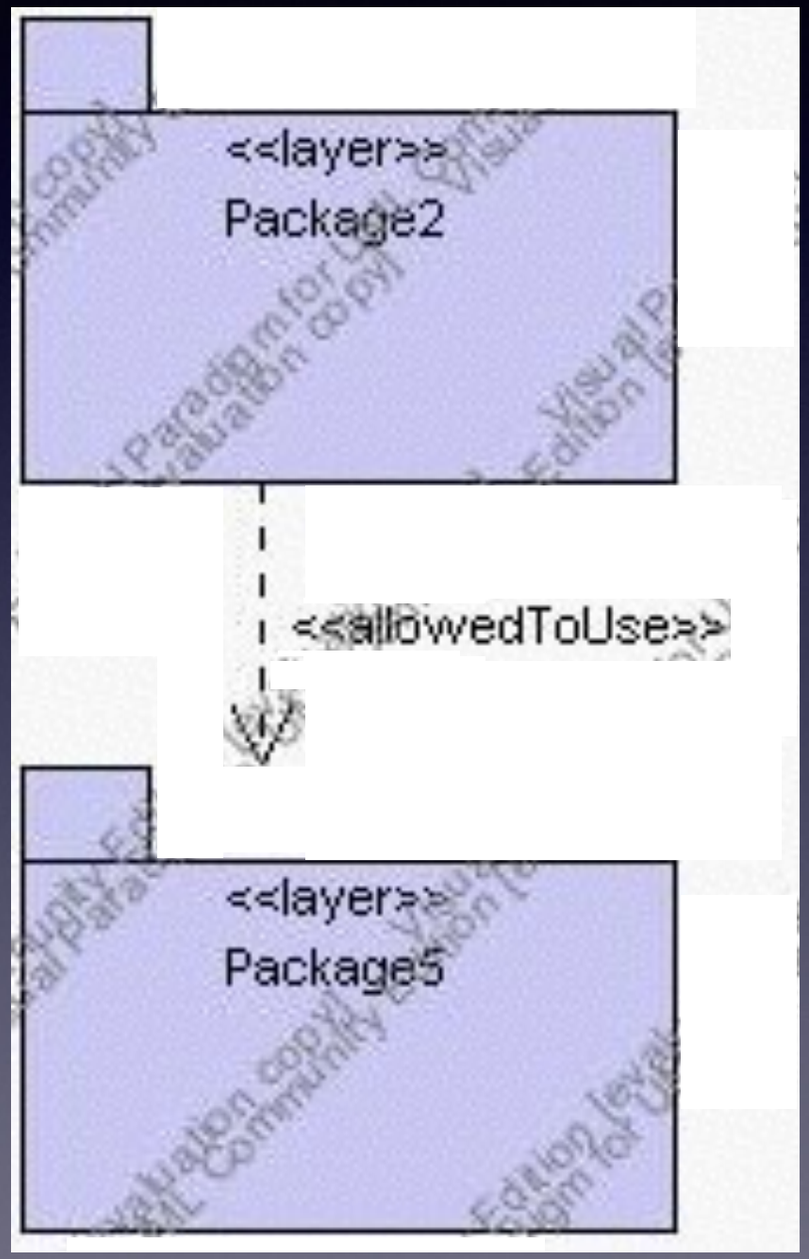
esempi

- User interface USA database
- A usa B che usa C e D, E usa B, B usa A
- Vedere esempio finale

vista a strati (macchine virtuali)

- Elementi: strati
  - uno strato è un insieme coeso di moduli
    - a volte raggruppati in segmenti
  - offre un’interfaccia pubblica per i suoi servizi (macchina virtuale)
- Relazione: può usare
  - antisimmetrica (a meno di eccezioni)
  - transitiva (se esplic. detto)
- Usi
  - modificabilità e portabilità
  - controllo della complessità
  - analisi d’impatto

notazione



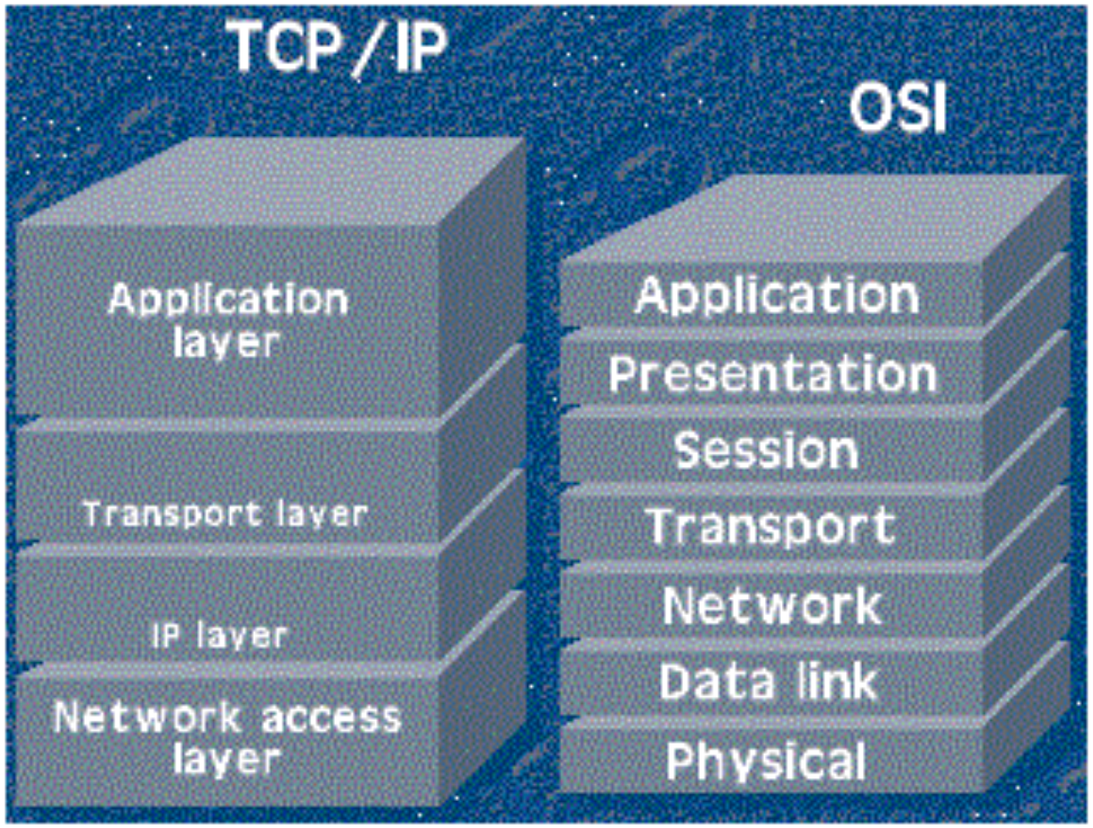
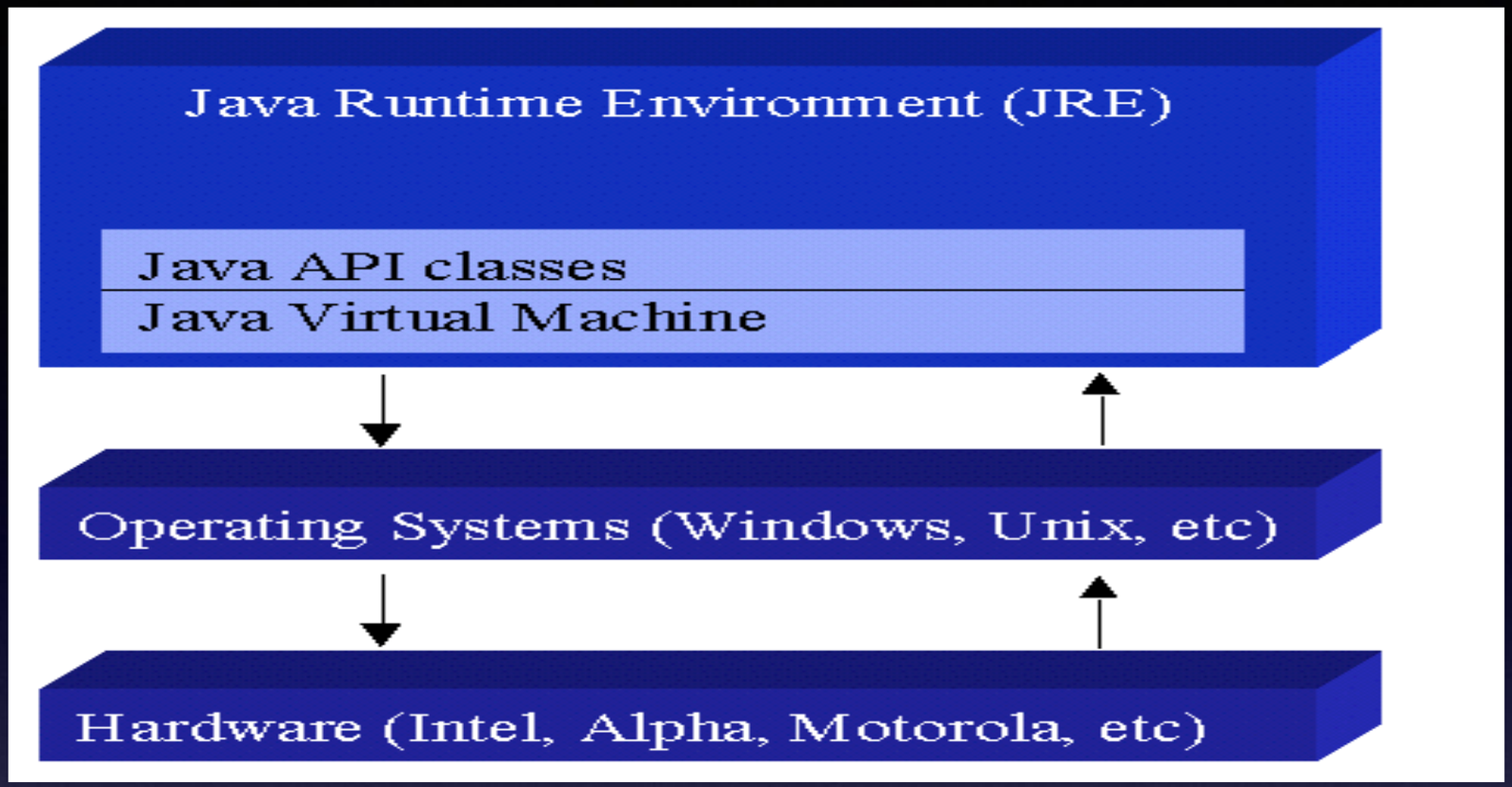
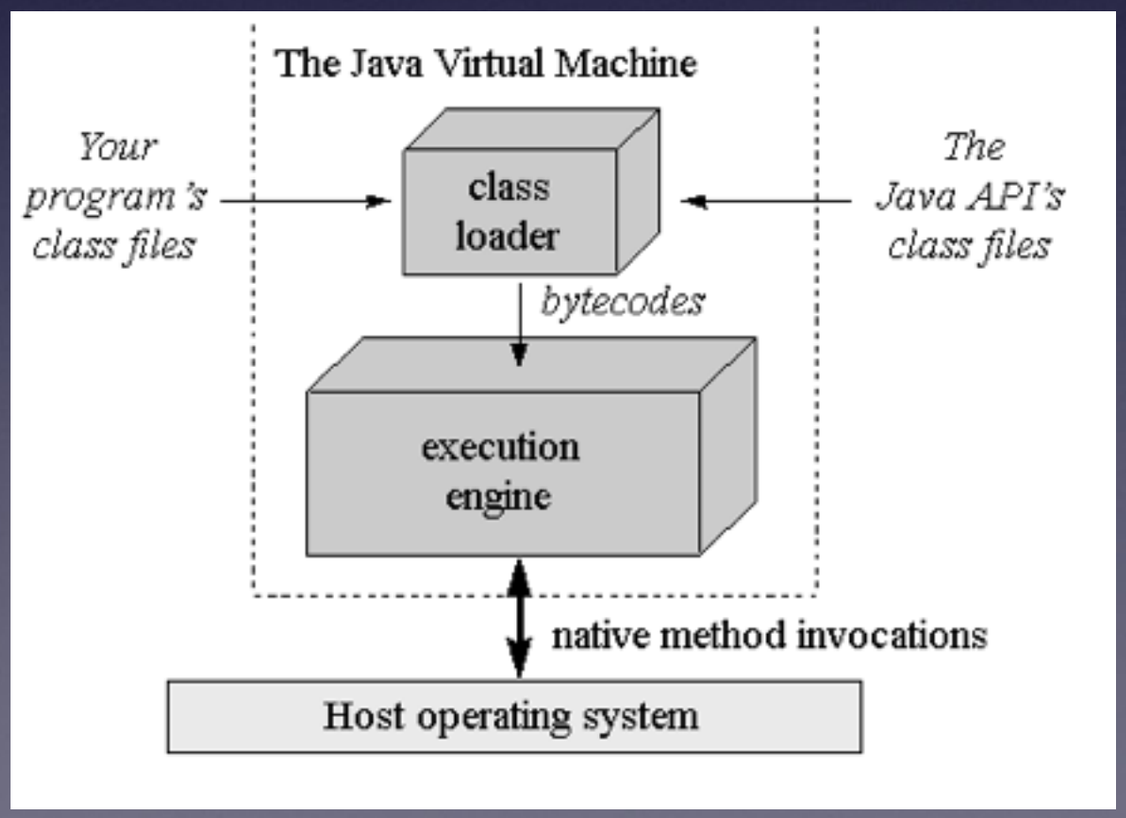


Fig. 1.2.3 - Strati OSI e strati dell'architettura TCP/IP

Telnet	FTP	SMTP	DNS	Application Layer (Layer OSI 5)
TCP			UDP	Transport Layer (Layer OSI 4)
IP ICMP ARP				InterNetwork Layer (Layer OSI 3)
X.25	Ethernet	ISDN	Token-Ring	Network Access (Layer OSI 2)

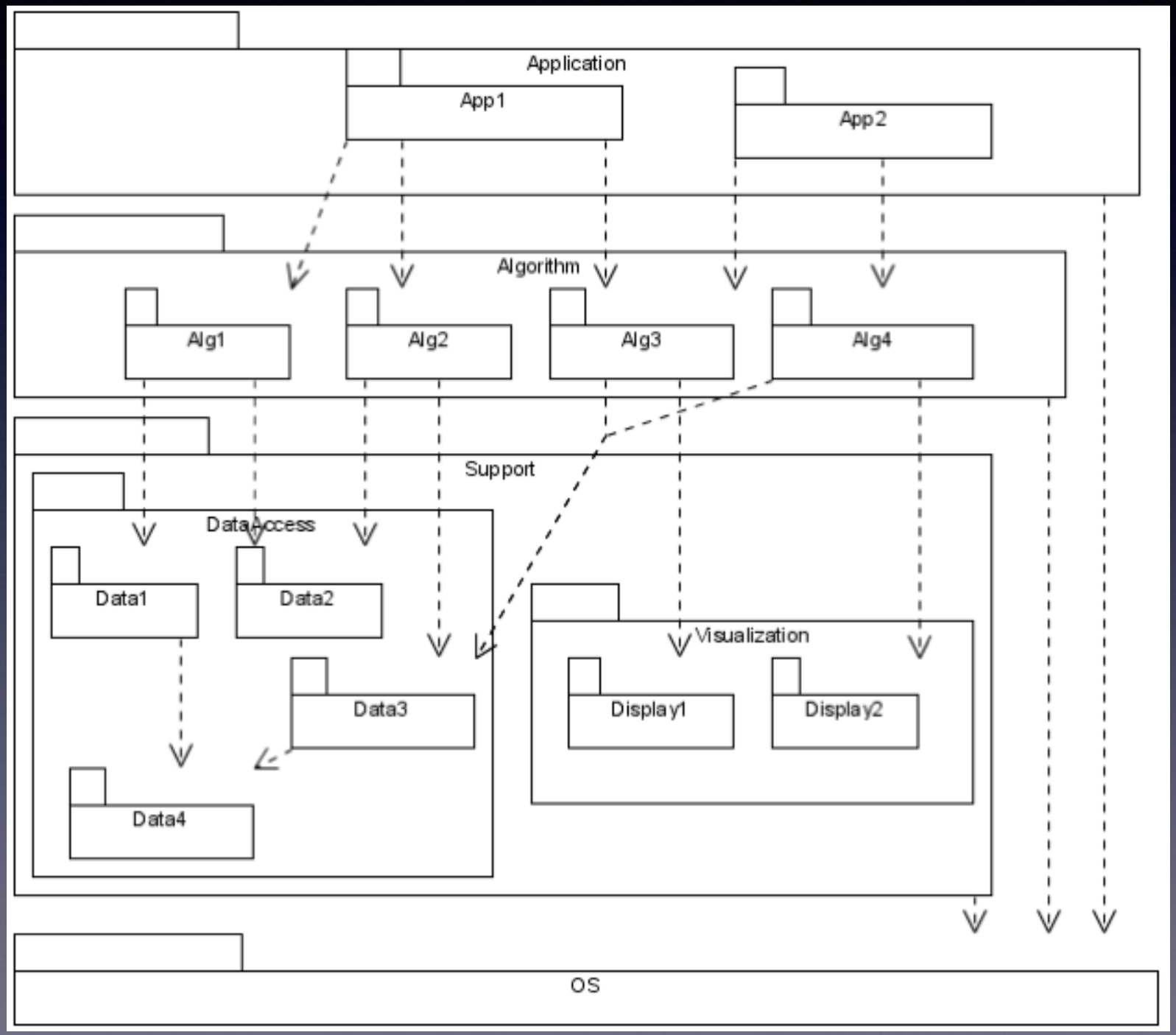
Fig. 1.2.3 - Strati OSI relativi all'architettura TCP/IP





Un esempio familiare di architettura a strati è l'ambiente di esecuzione di programmi Java (Java platform): la macchina virtuale Java (Java Virtual Machine o JVM) rappresenta un'astrazione del sistema operativo e costituisce uno strato tra questo e l'applicazione nel bytecode generato dalla compilazione del codice sorgente

vista ibrida



vista strutturale di generalizzazione

- Vista strutturale in cui le relazioni sono unicamente di tipo “eredita da”
- Notazione: generalizzazione
  - La generalizzazione tra package è di fatto un abuso di notazione, con il significato che alcune delle classi contenute nel package A ereditano da alcune classi del package B
- Utile per descrivere AS ottenute per istanziazione di un framework
  - un framework è definito da un insieme di classi astratte e dalle relazioni tra esse. Istanziare un framework significa fornire un'implementazione delle classi astratte
  - “Nella produzione del software, il framework è una struttura di supporto su cui un software può essere organizzato e progettato. […] Il termine framework può essere tradotto come intelaiatura o struttura, che è appunto la sua funzione, a sottolineare che al programmatore rimane solo da creare il contenuto vero e proprio dell’applicazione” [wikipedia]

framework

- Un framework può essere visto come (e in alcuni casi è) un sovra-insieme o un’aggiunta alle librerie di run-time di un linguaggio

  - oltre alla libreria standard, C++ ha un framework non grafico, la Standard Template Library (STL); esistono numerosi framework che estendono o, in parte, sostituiscono la STL, tra cui l'Active Template Library (ATL) e le Microsoft Foundation Classes (MFC), Qt di Trolltech, e wxWidgets. Microsoft ha inoltre sviluppato un'implementazione del C++ (ora standard ECMA: C++/CLI) che si appoggia integralmente al Framework .NET

  - anche il C ha una propria libreria di run-time, la libreria standard del C, nota anche come libc (in ambiente Unix) o CRT (in ambiente Microsoft); tuttavia vi sono parecchi framework per il C, fra cui il GIMP Toolkit (GTK)

  - per C#, che non ha una propria libreria di run-time, il framework .NET svolge anche questa funzione. Lo stesso vale per VB .NET

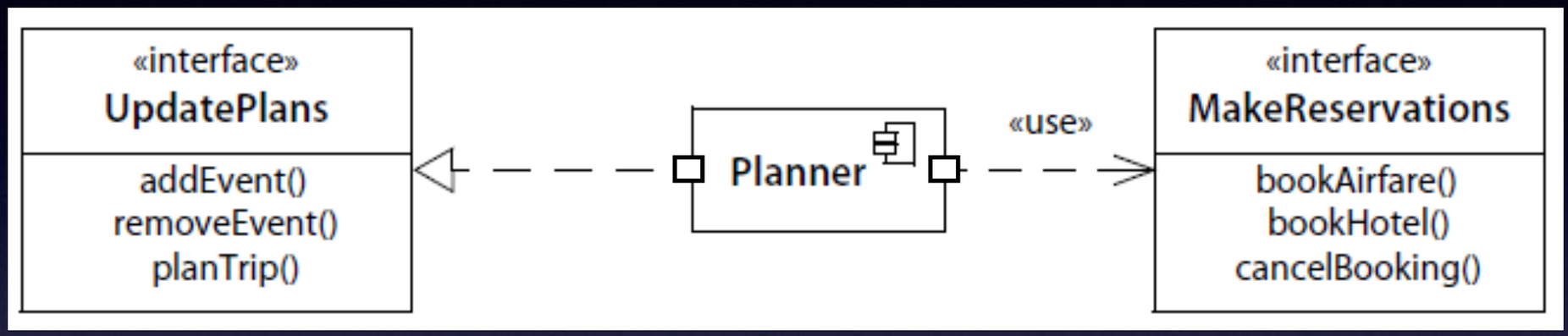
viste comportamentali

- Elementi
  - Componenti: entità presenti a tempo d’esecuzione
    - processi, oggetti, serventi, depositi di dati…
    - “**a** component represents a modular part of a system that encapsulates its contents and whose manifestation is replaceable within its environment”
  - Connettori: canali di interazione
    - protocolli, flussi d’informazione, accessi ai depositi…
- Proprietà
  - Componenti
    - nome, tipo (numero e tipo dei porti), altre informazioni specializzate (prestazioni, affidabilità…)
  - Connettori
    - nome, tipo (numero e tipo dei ruoli), protocollo, prestazioni…

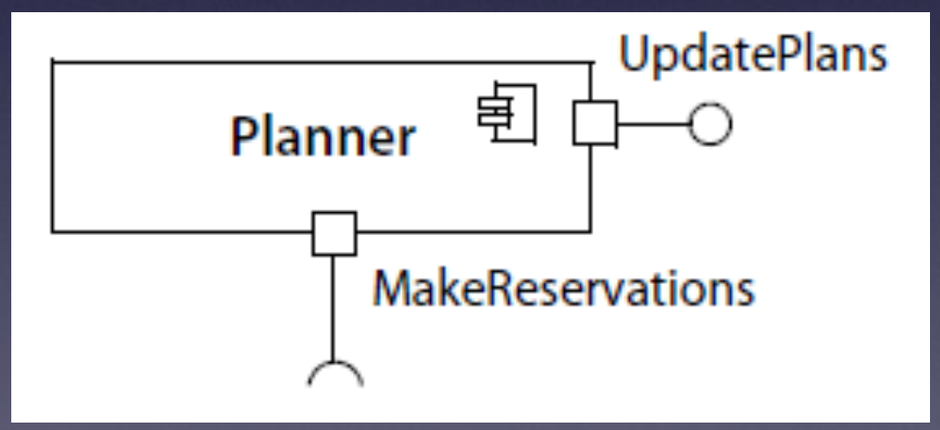
viste comportamentali

- Usi
  - analisi delle caratteristiche di qualità a tempo d'esecuzione
    - prestazioni, affidabilità, disponibilità, sicurezza…
  - comunicazione della struttura del sistema in esecuzione
    - flusso dei dati, dinamica, parallelismo, replicazioni…
- Relazioni
  - connessione di ruoli a porti

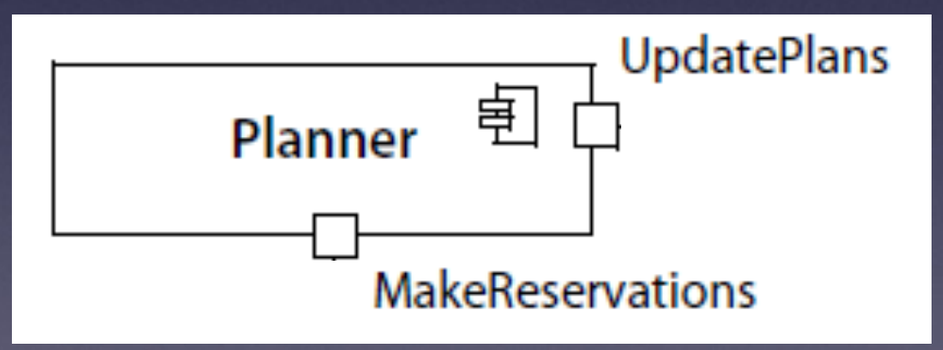
notazione



uso esplicito di interfacce



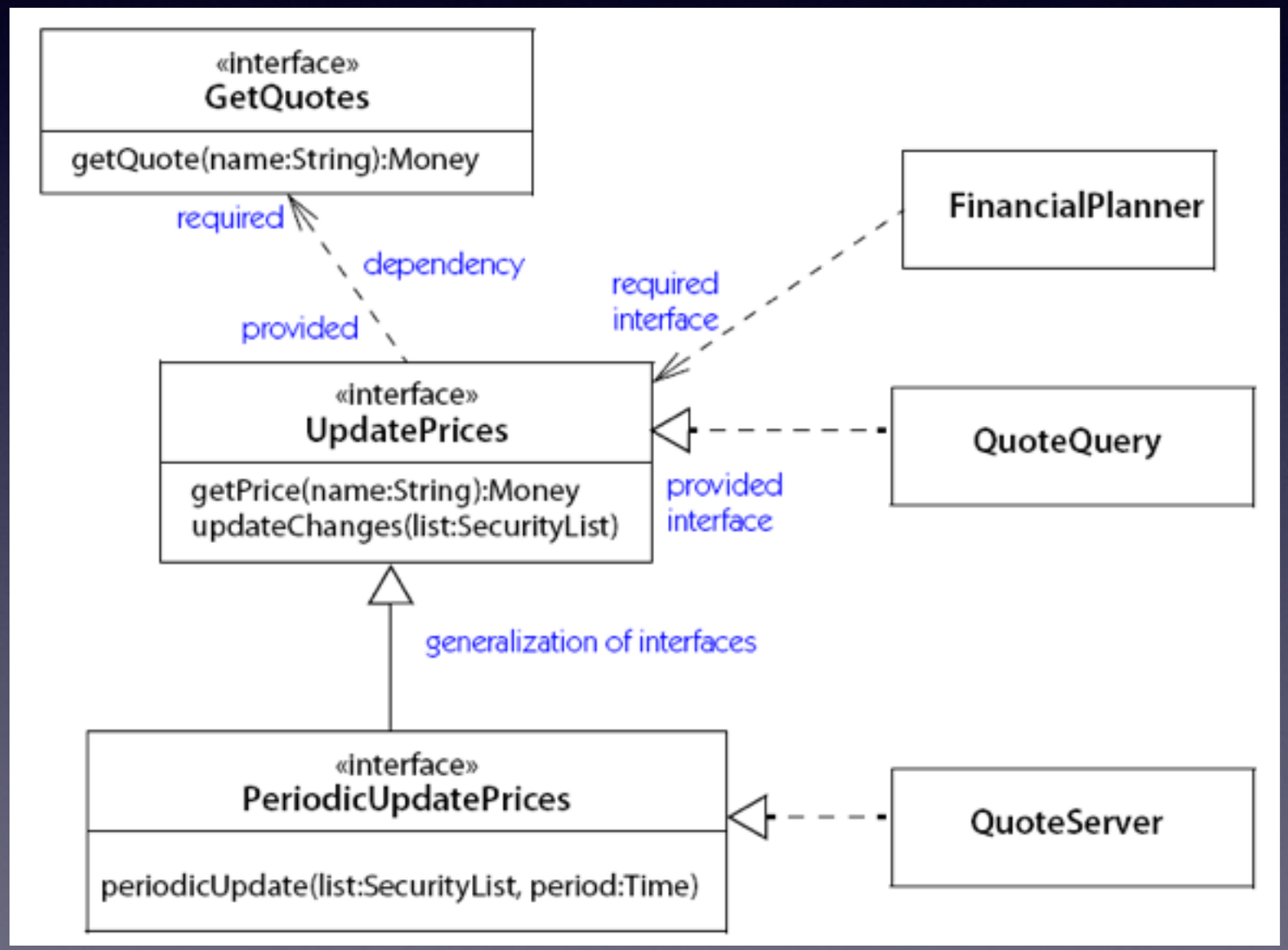
notazione con porti



l’uso di “lollipop” è una notazione imprecisa sulle viste c&c (ma non sulle viste di struttura composita)

diagramma di componente

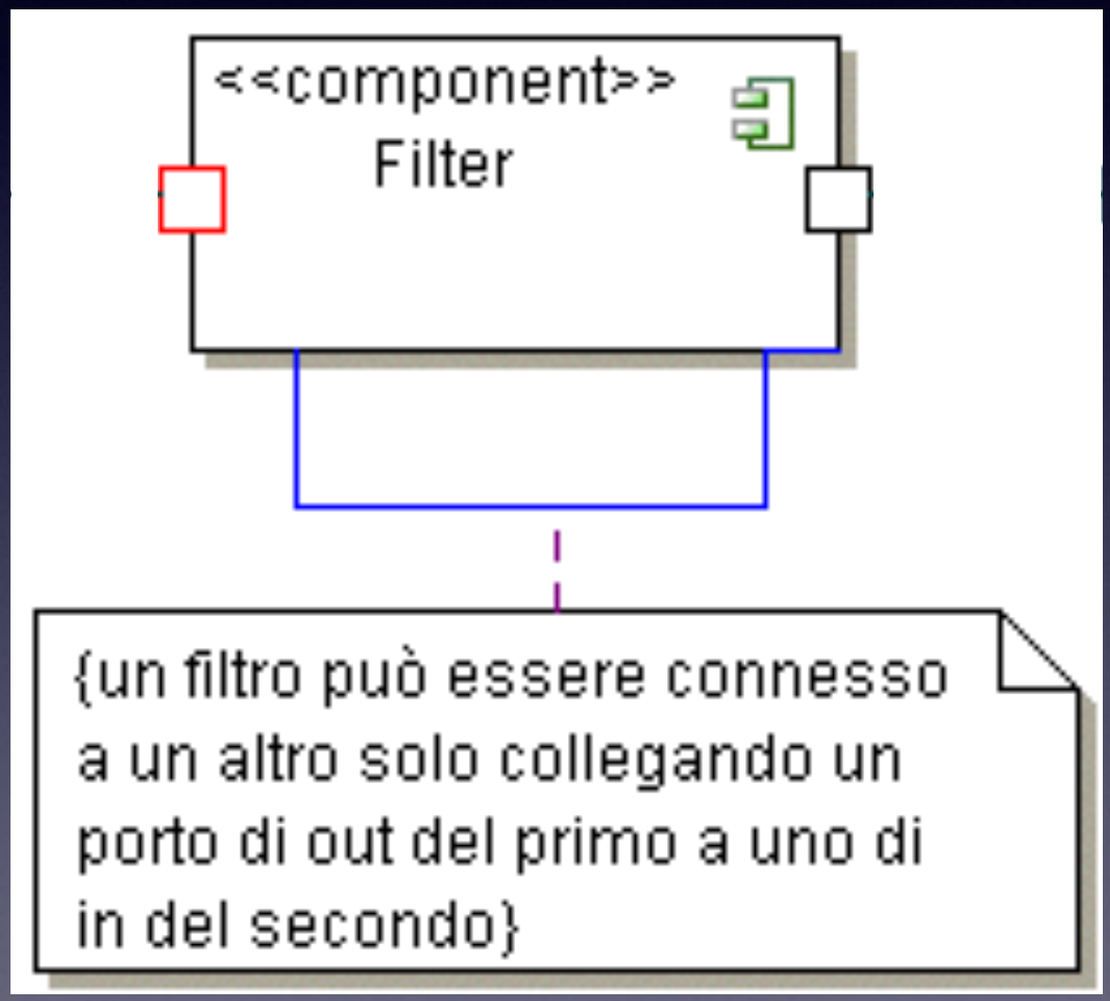
Un diagramma che mostra definizione, struttura interna e dipendenze dei tipi componente. Non c'è demarcazione netta fra i diagr. di componenti e i più generali diagrammi di classi



componenti

- *A component represents a modular part of a system that encapsulates its contents and whose manifestation is replaceable within its environment* [OMG UML 2008]

- Quindi, mentre con le parti si pone l'enfasi sulla struttura, con le componenti si enfatizza la flessibilità (anche a tempo d'esecuzione)



[notazione: in arancione, out nero]

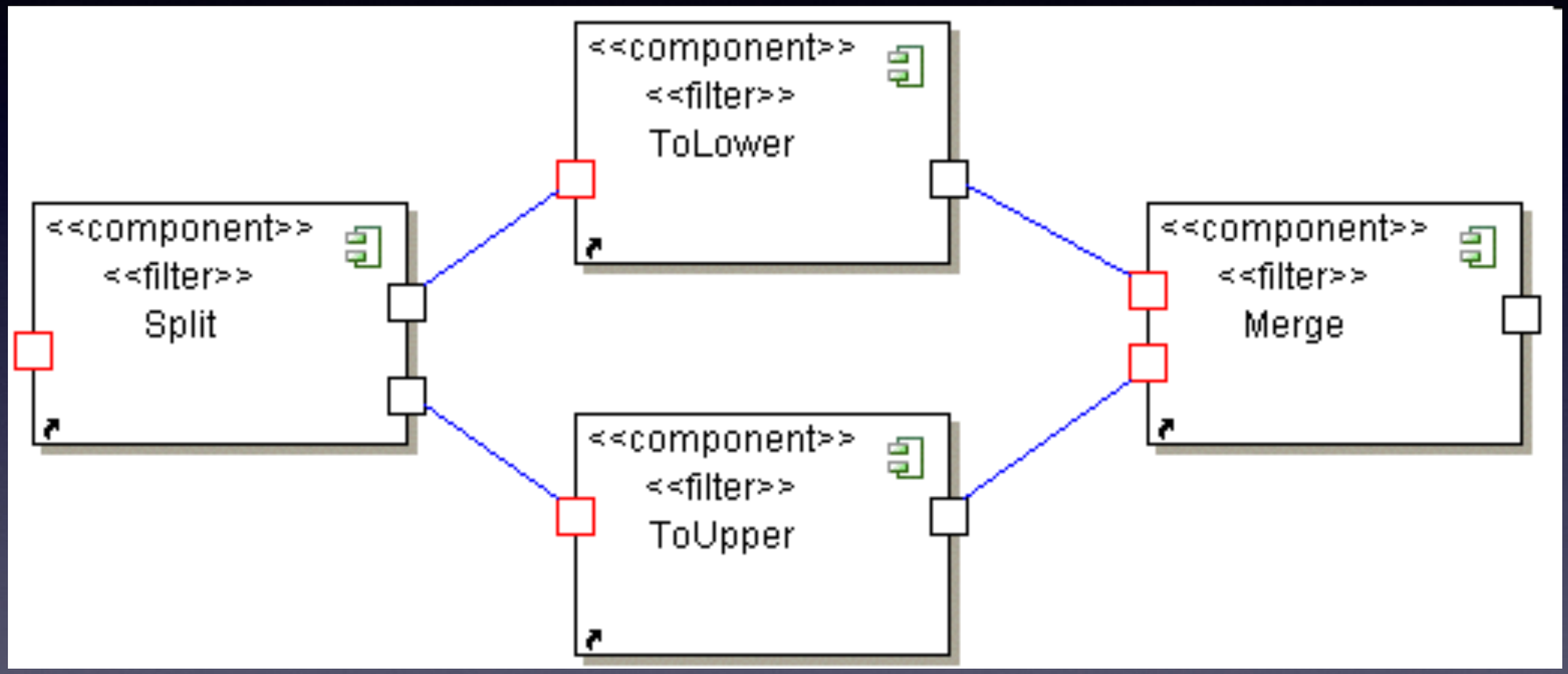
porti

- Un porto è una feature strutturale d un *classificatore* (classe, componente) che incapsula l'interazione fra i contenuti del classificatore e il suo ambiente
  - sono caratterizzati dalle interfacce che forniscono o richiedono
  - sono connessi da linee semplici (connettori generici) con un eventuale stereotipo che le descrive
  - sono indicati con una convenzione sui colori [rosso in, nero out]

stili comportamentali: condotte e filtri (pipe & filter)

- Componenti
  - filtro: trasforma uno o più flussi di dati dai porti d'ingresso in uno o più flussi sui porti d'uscita
- Connettori
  - condotta (pipe): canale di comunicazione unidirezionale bufferizzato che preserva l'ordine dei dati dal ruolo d'ingresso a quello d'uscita
- Usi
  - pre-elaborazione in sistemi di elaborazione di segnali
  - analisi dei flussi dei dati, e.g. dimensioni dei buffer

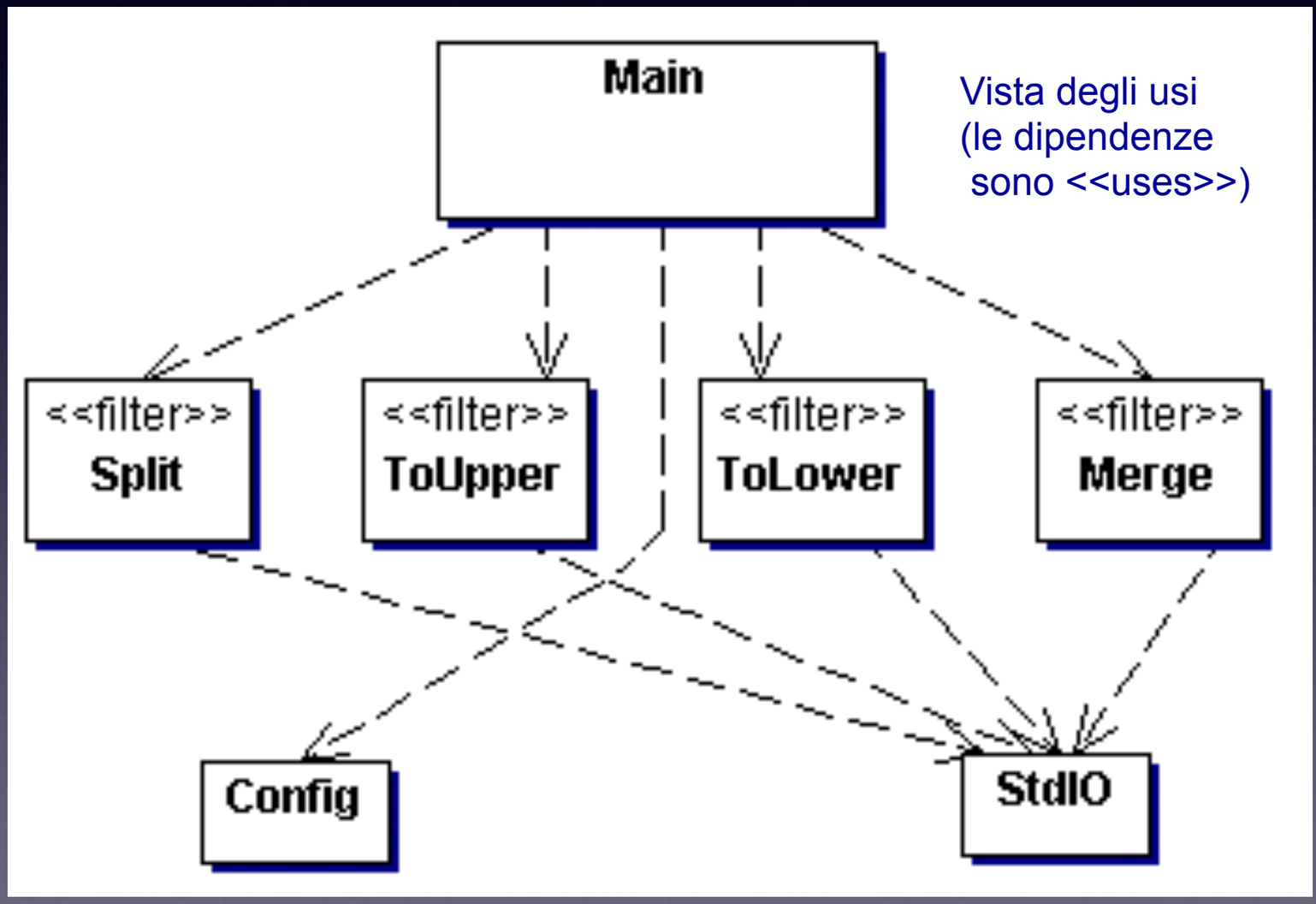
condotte e filtri: applicazione



- I connettori tra porti di colori diversi indicano la presenza di una relazione di *pipe* tra i due filtri, quelli tra porti dello stesso colore (uno della struttura composta, uno di una parte) la connessione di ‘delega’ – la parte fa ciò che dovrebbe fare il tutto. Il comportamento, e la specifica dei vari filtri, è deducibile dall’esempio

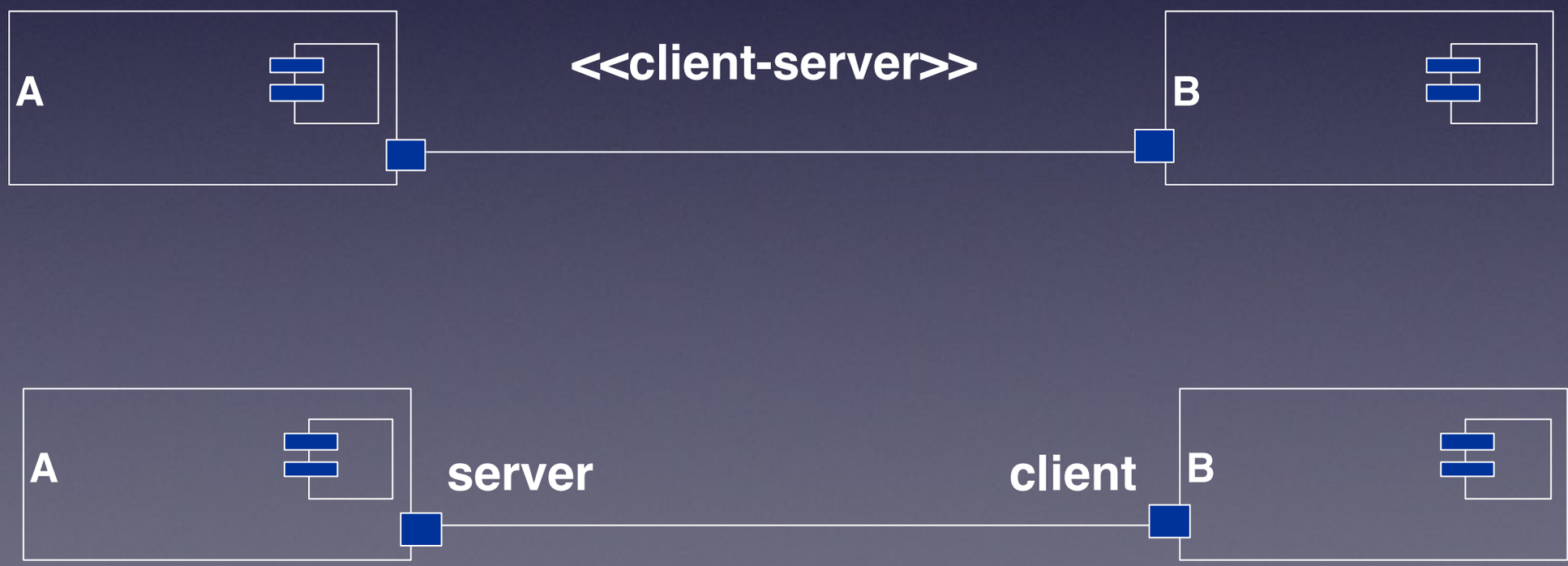
strutturale vs. comportamentale

- anche qui i filtri si chiamano, ma non si usano…
- e il main li configura per metterli in comunicazione via StdIO (realizzazione del connettore pipe)
  - suggerisce anche una vista a strati...



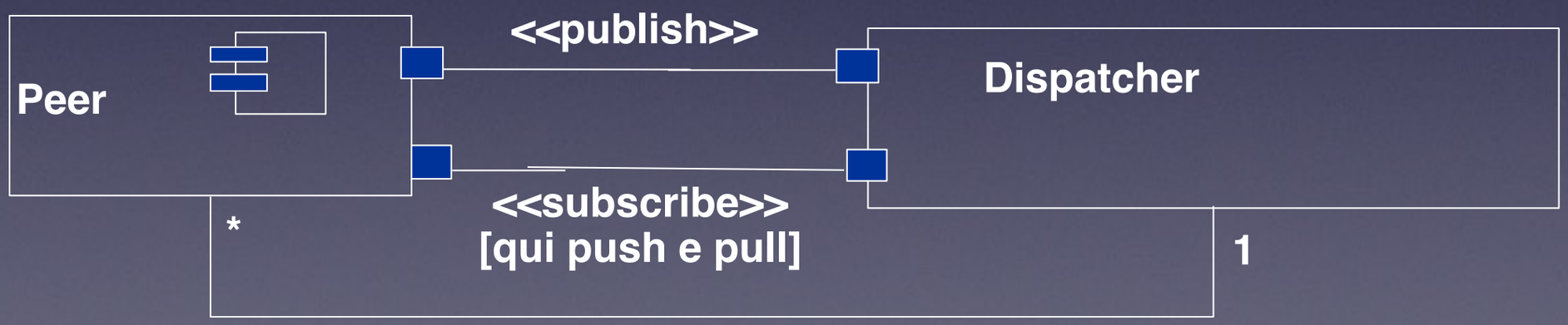
stili comportamentali cliente-servente

- le componenti interagiscono chiedendo servizi
- comunicazioni appaiate: richiesta e fornitura del servizio



stile publish-subscribe

- le componenti interagiscono annunciando eventi: ciascuna componente si “abbona” a classi di eventi rilevanti per il suo scopo
- ciascuna componente, volendo, può essere sia produttore che consumatore di eventi
- disaccoppia produttori e consumatori di eventi e favorisce le modifiche dinamiche del sistema



Le operazioni: subscribe, unsubscribe, publish, notify (push), letMeKnow (pull)

[versione con lollipop in Binato et alii]

ancora publish-subscribe

- Mittenti e destinatari di messaggi dialogano attraverso un tramite (detto dispatcher o broker). Il mittente di un messaggio (publisher) non deve essere consapevole dell'identità dei destinatari (subscriber); esso si limita a "pubblicare" il proprio messaggio al dispatcher. I destinatari si rivolgono a loro volta al dispatcher "abbonandosi" alla ricezione di messaggi. Il dispatcher quindi inoltra ogni messaggio inviato da un publisher a tutti i subscriber interessati a quel messaggio

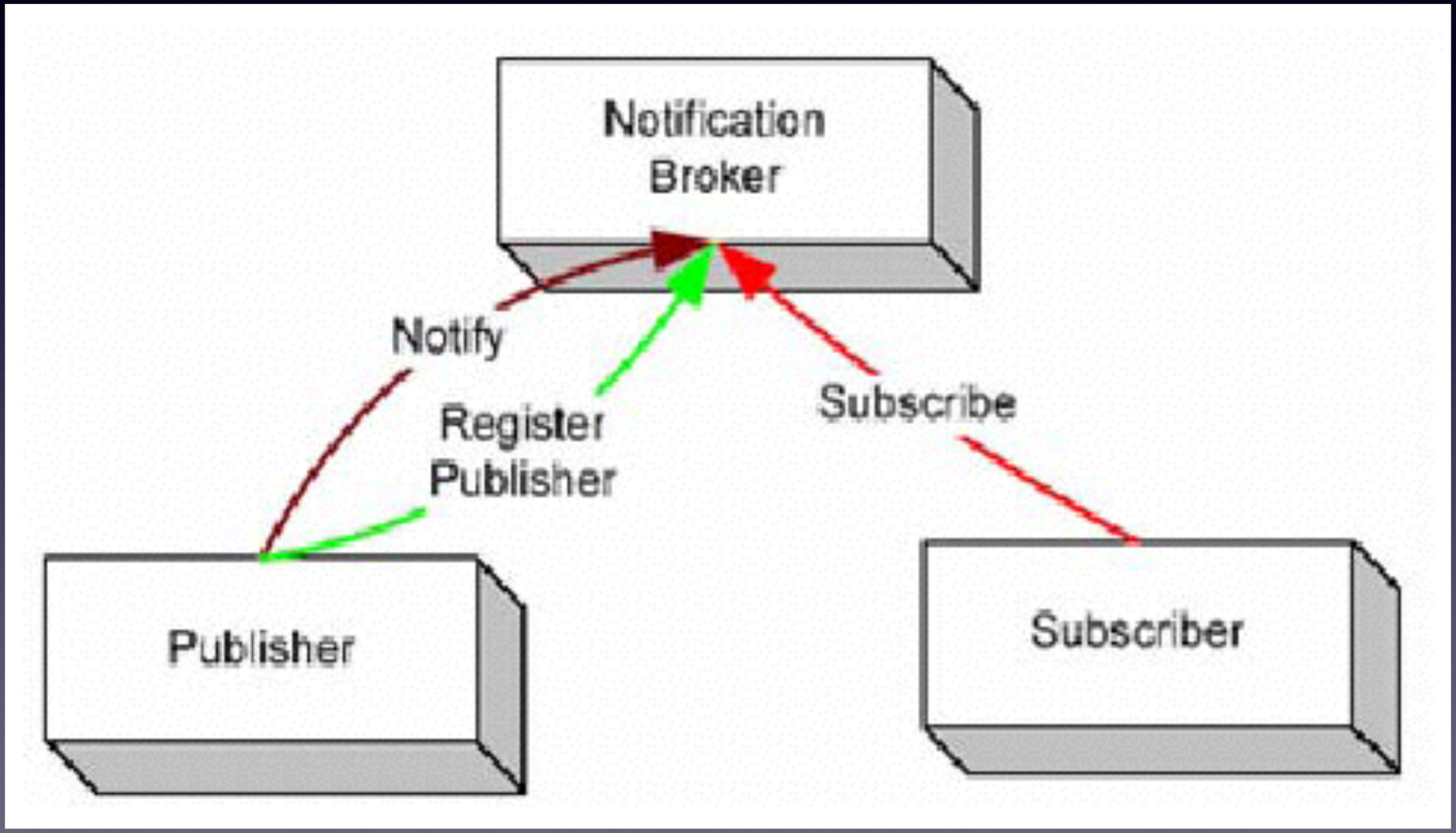
- In genere, il meccanismo di sottoscrizione consente ai subscriber di precisare nel modo più specifico possibile a quali messaggi sono interessati: ad esempio, un subscriber potrebbe "abbonarsi" solo alla ricezione di messaggi da determinati publisher, oppure aventi certe caratteristiche

- Questo schema implica che ai publisher non sia noto quanti e quali sono i subscriber e viceversa. Questo può contribuire alla scalabilità del sistema

esempi di publish-subscribe

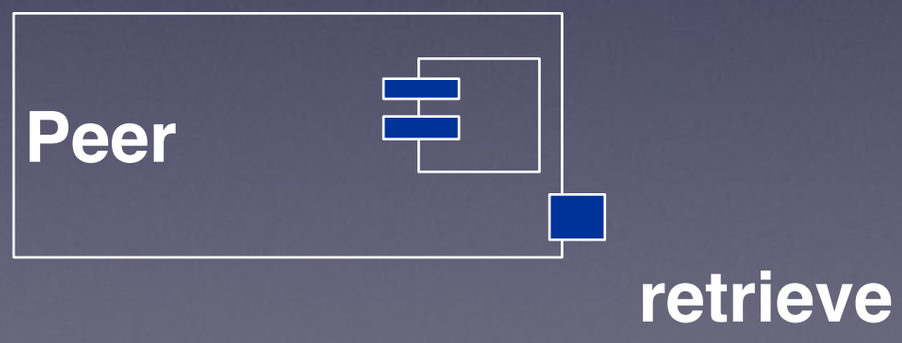
- Instant messaging
  - Implementazione peer-to-peer (WASTE) (sparisce il dispatcher ?)
  - Implementazione centralizzata (IRC, Jabber)
- HTTP streaming
- Reti: multicast con algoritmi di flooding
- Il Data Distribution Service for Real Time Systems (DDS) è uno standard emanato dall'Object Management Group (OMG) che definisce un middleware per la distribuzione di dati in tempo reale secondo il paradigma publish/subscribe

PS for web-services



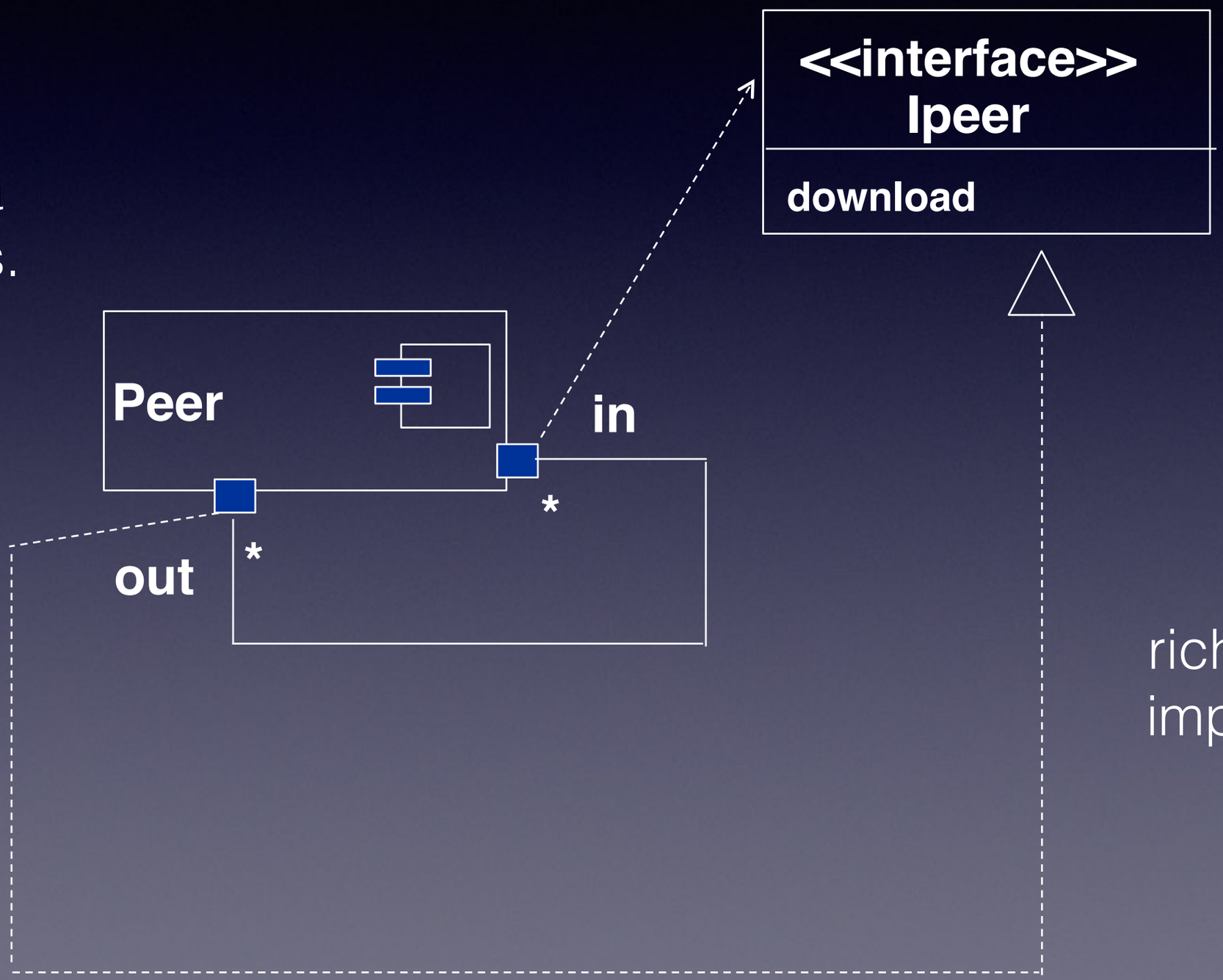
stile peer-to-peer

- i connettori non sono asimmetrici, come in cliente-servente
- infrastrutture per oggetti distribuiti, come CORBA, COM+



ancora peer-to-peer

interfaccia implicita vs. esplicita



richiede vs. implementa