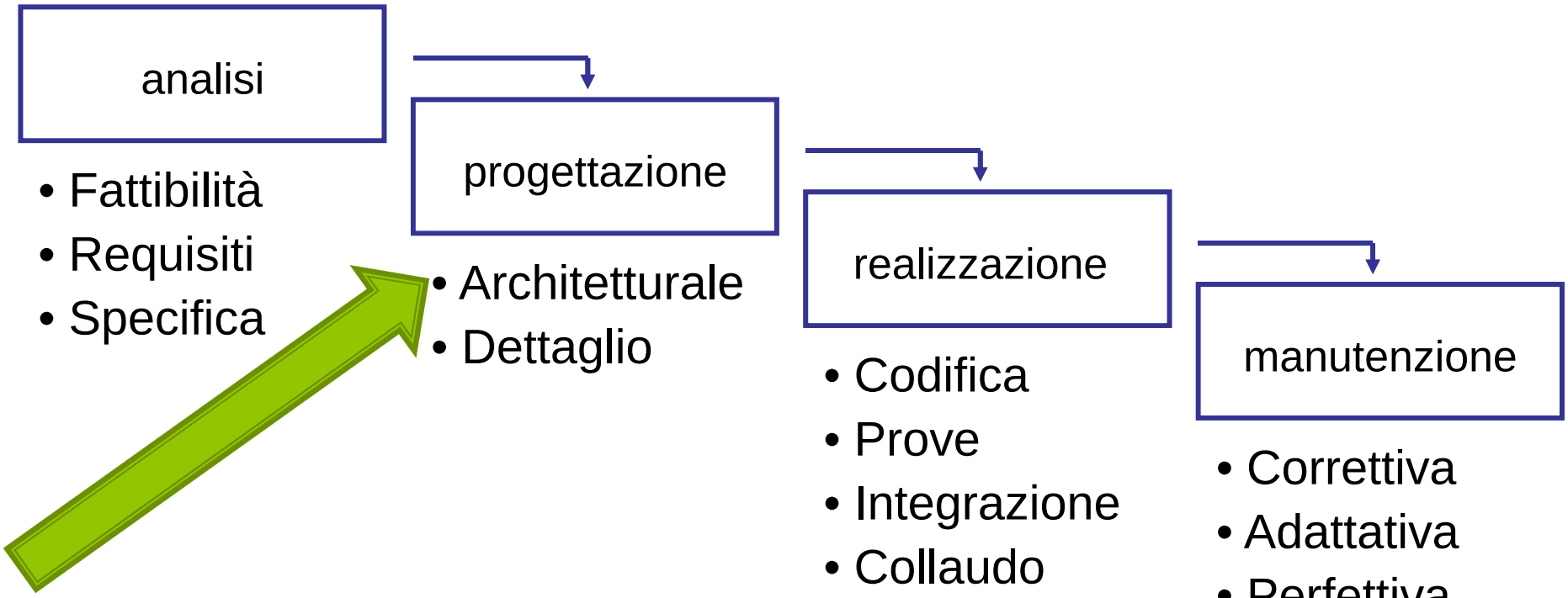
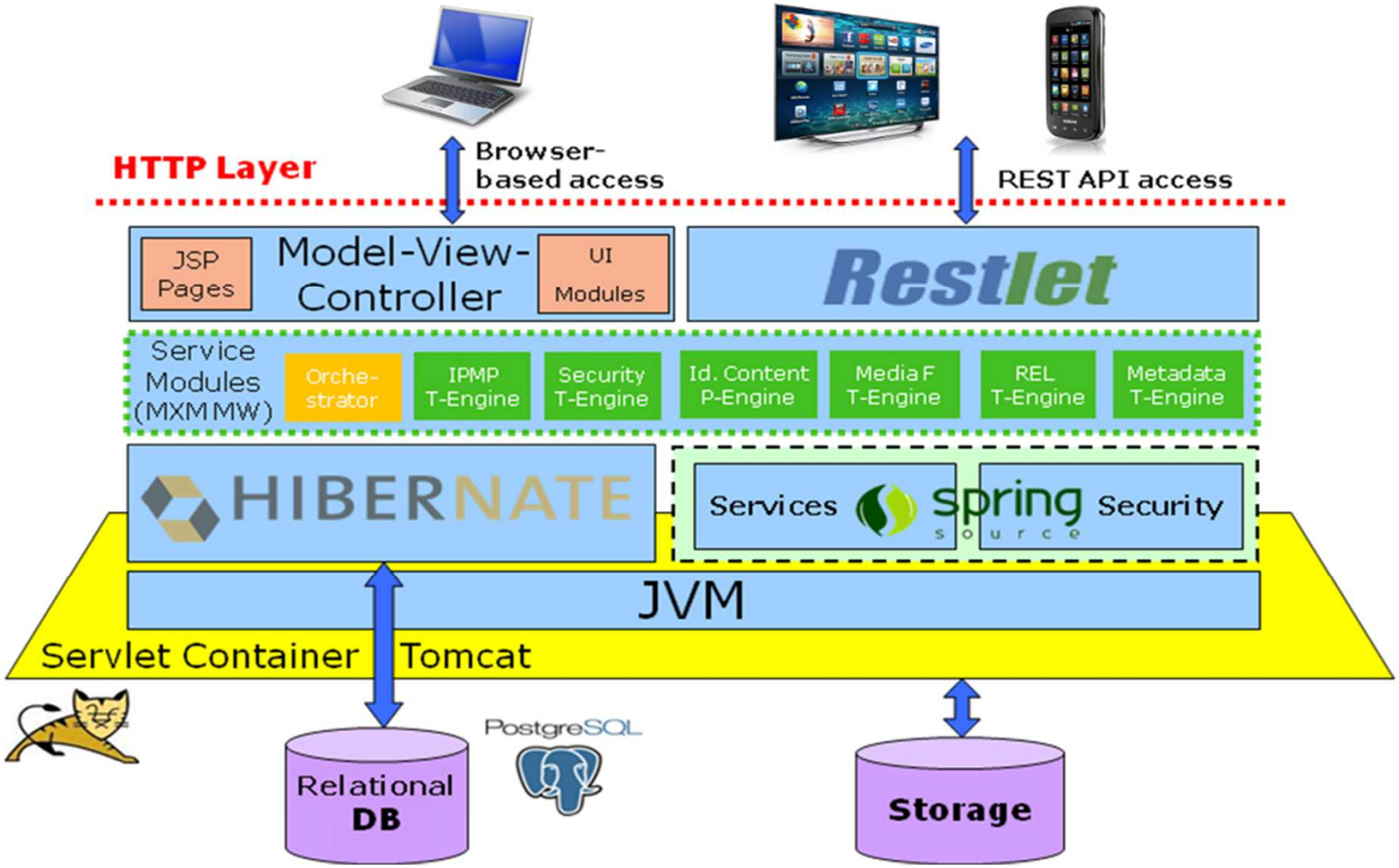
Architetture Software

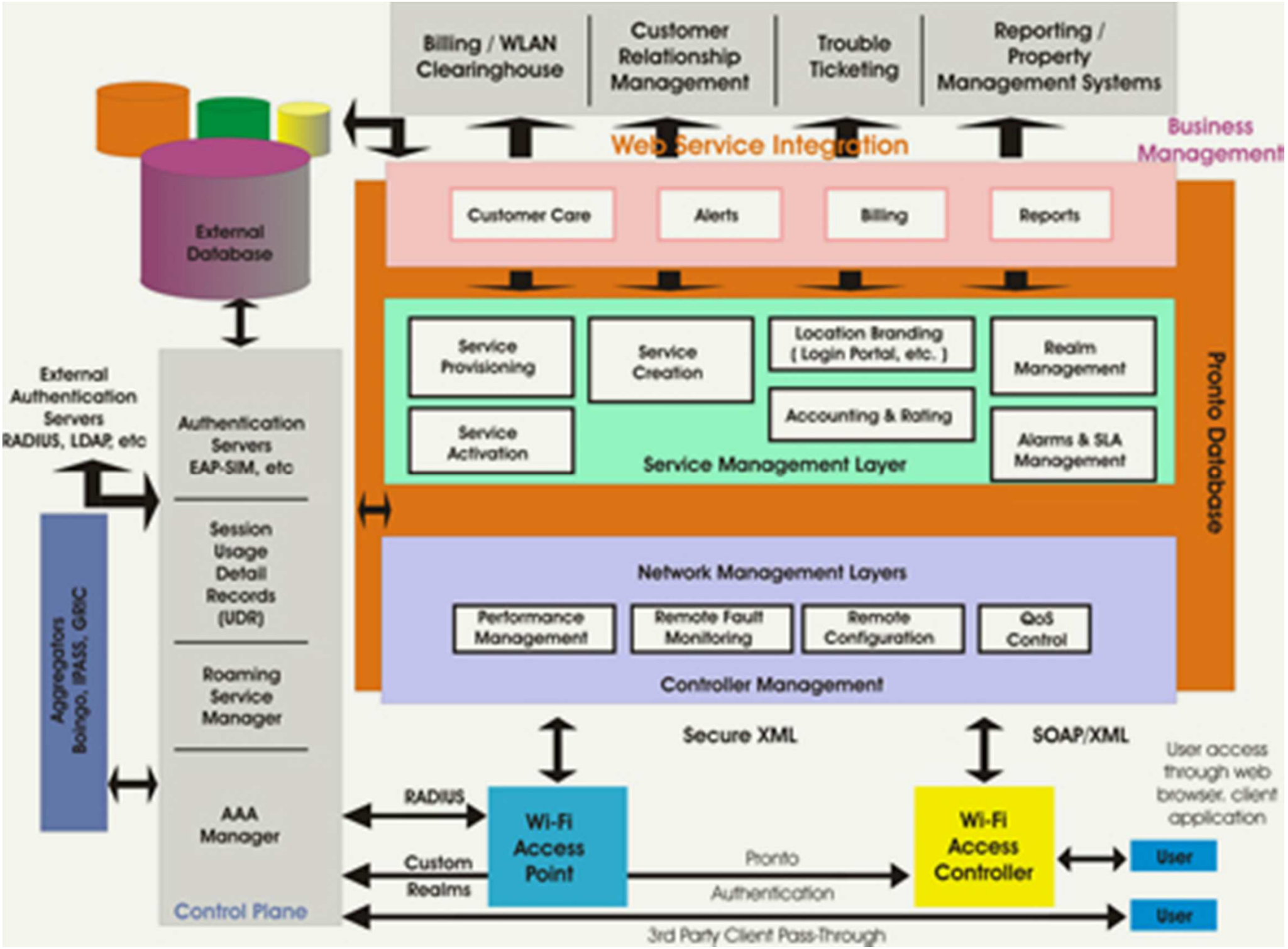
Vincenzo Gervasi, Laura Semini
Ingegneria del Software
Dipartimento di Informatica
Università di Pisa

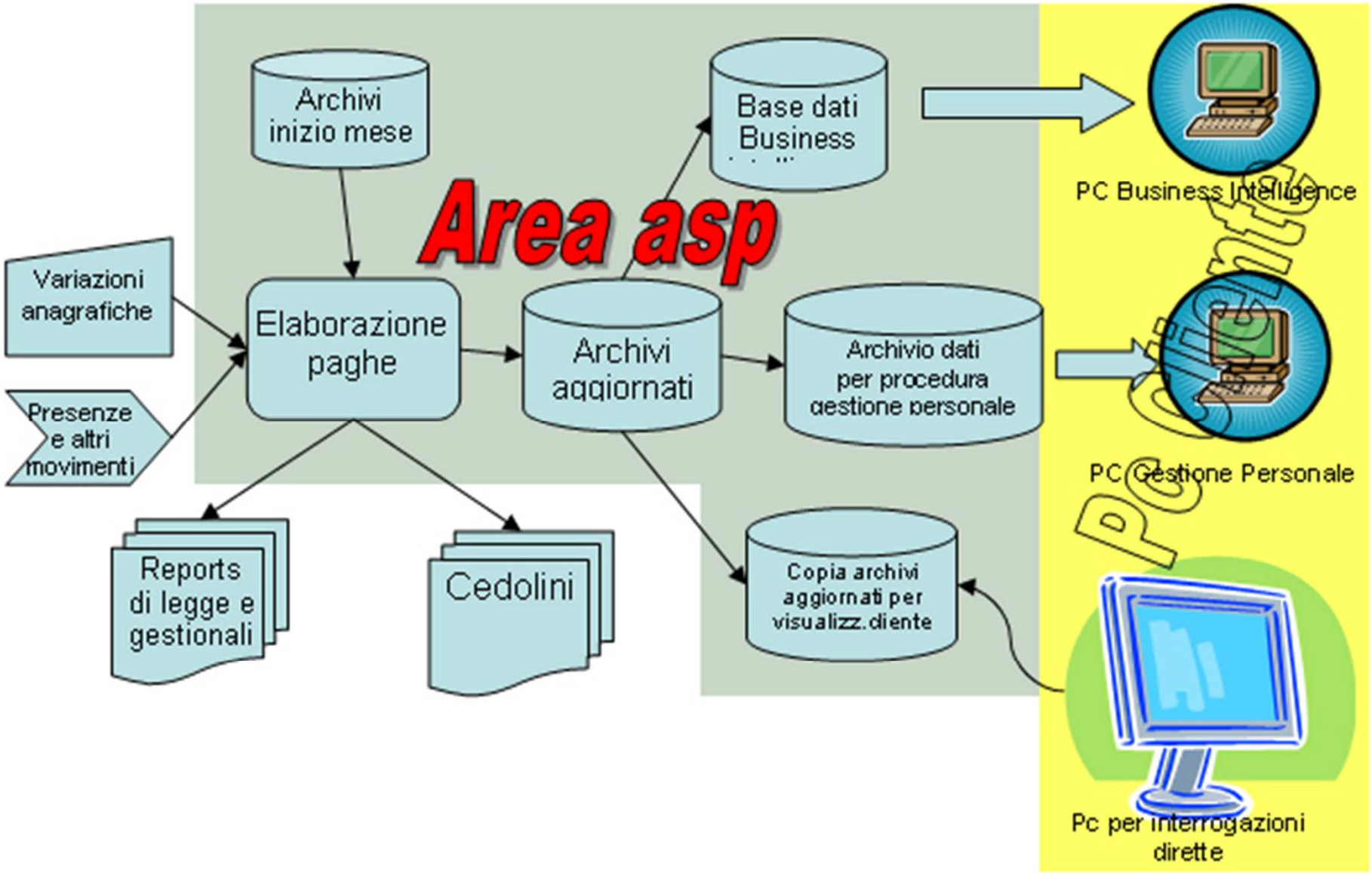
Riassunto lezioni precedenti
Outline delle prossime



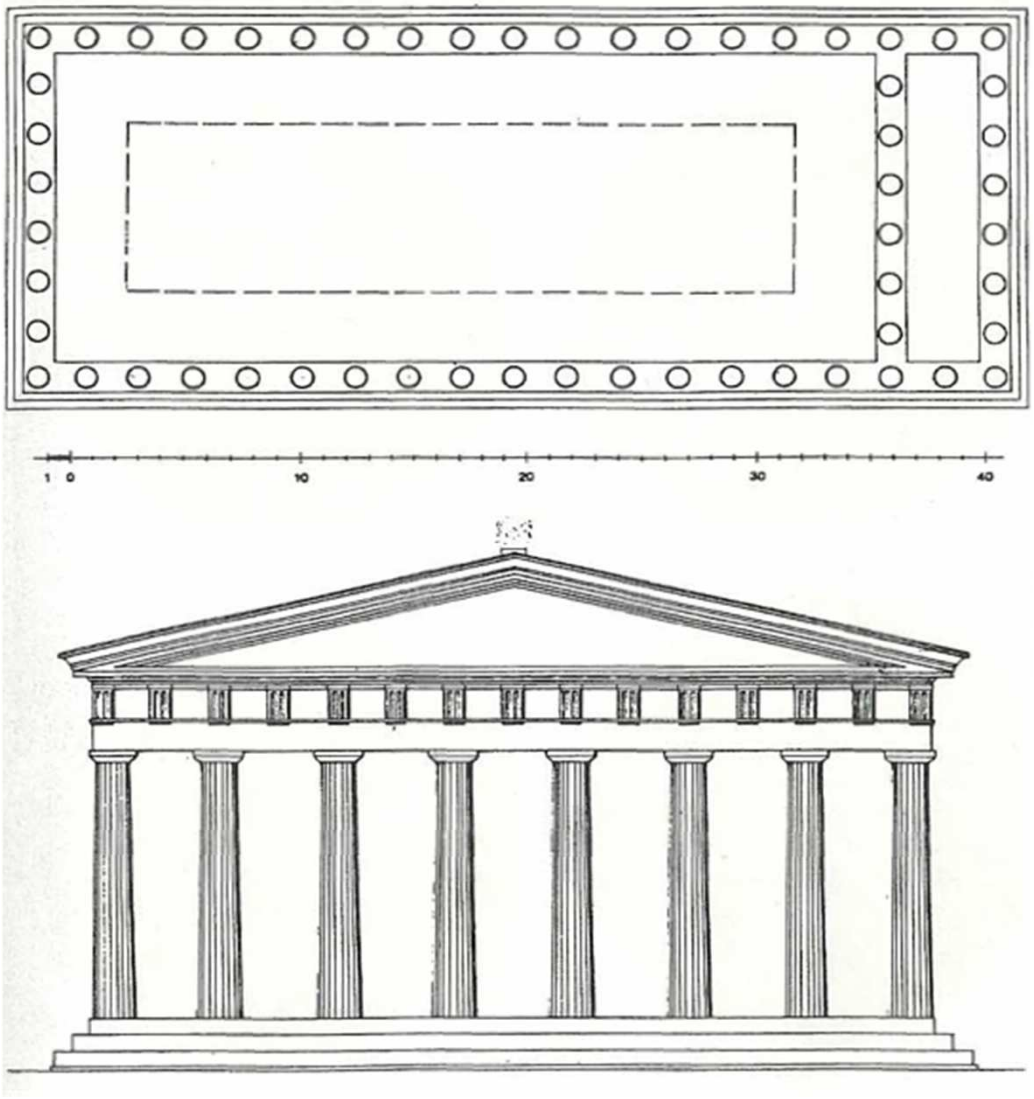
Esempio di "descrizione" di una architettura...



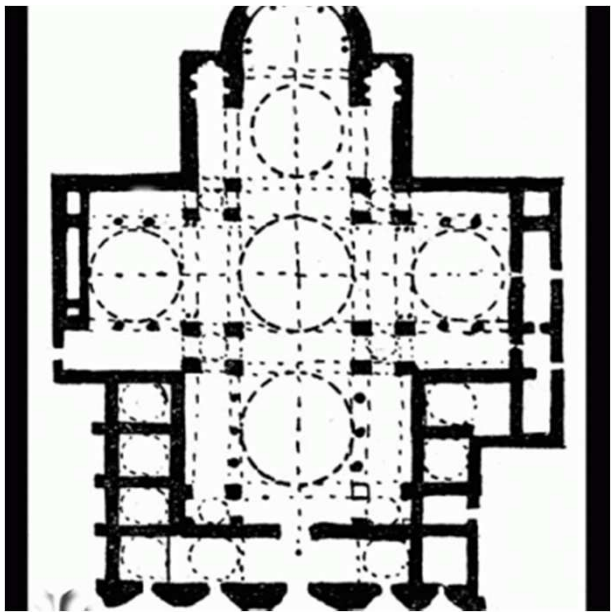

Billing / WLAN Clearinghouse
Customer Relationship Management
Trouble Ticketing
Reporting / Property Management Systems
Business Management
Web Service Integration
External Database
Customer Care
Alerts
Billing
Reports
Service Provisioning
Service Creation
Location Branding (Login Portal, etc.)
Realm Management
Accounting & Rating
Service Activation
Alarms & SLA Management
Service Management Layer
Pronto Database
External Authentication Servers RADIUS, LDAP, etc
Authentication Servers EAP-SIM, etc
Session Usage Detail Records (UDR)
Roaming Service Manager
Aggregators Boingo, IPASS, GRIC
AAA Manager
Control Plane
Network Management Layers
Performance Management
Remote Fault Monitoring
Remote Configuration
QoS Control
Controller Management
Secure XML
SOAP/XML
User access through web browser, client application
RADIUS
Custom Realms
Wi-Fi Access Point
Pronto Authentication
Wi-Fi Access Controller
User
User
3rd Party Client Pass-Through

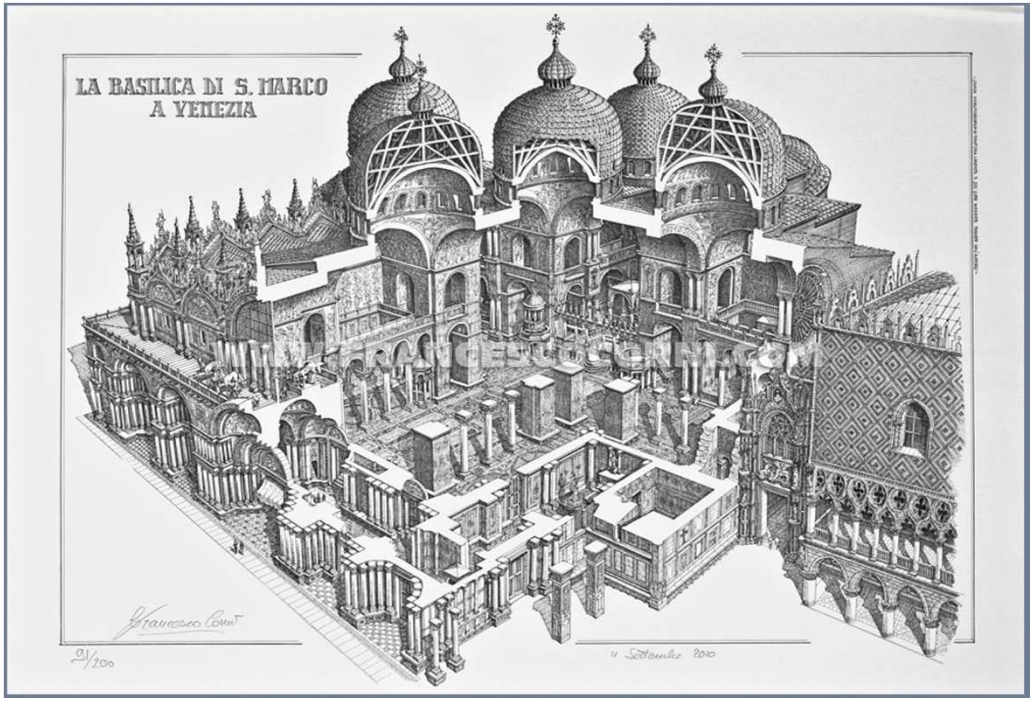

Archivi inizio mese
Base dati Business
Area asp
PC Business Intelligence
Variazioni anagrafiche
Elaborazione paghe
Archivi aggiornati
Archivio dati per procedura gestione personale
Presenze e altri movimenti
PC Gestione Personale
Reports di legge e gestionali
Cedolini
Copia archivi aggiornati per visualizz.diente
Pc per interrogazioni dirette

tornando ai classici...

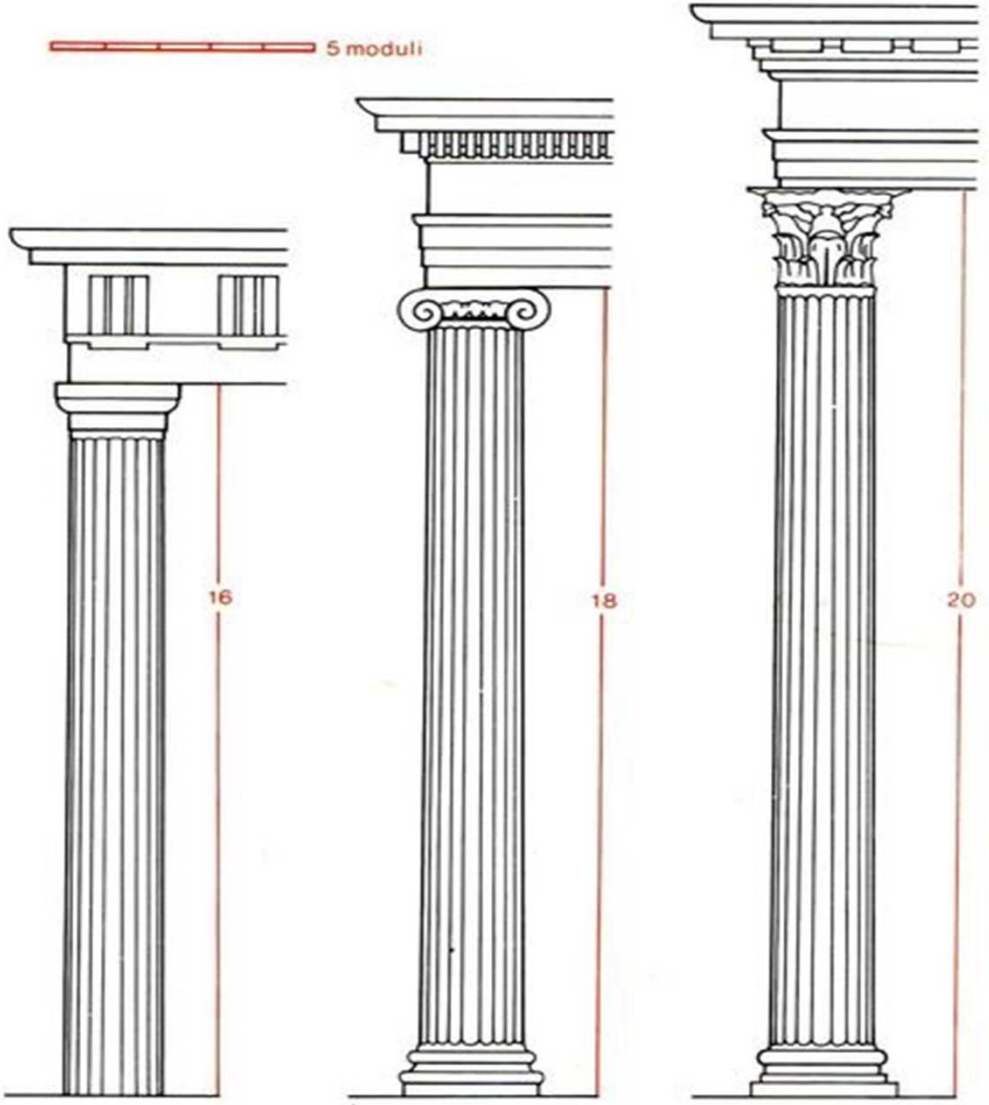


dalle viste all'insieme…

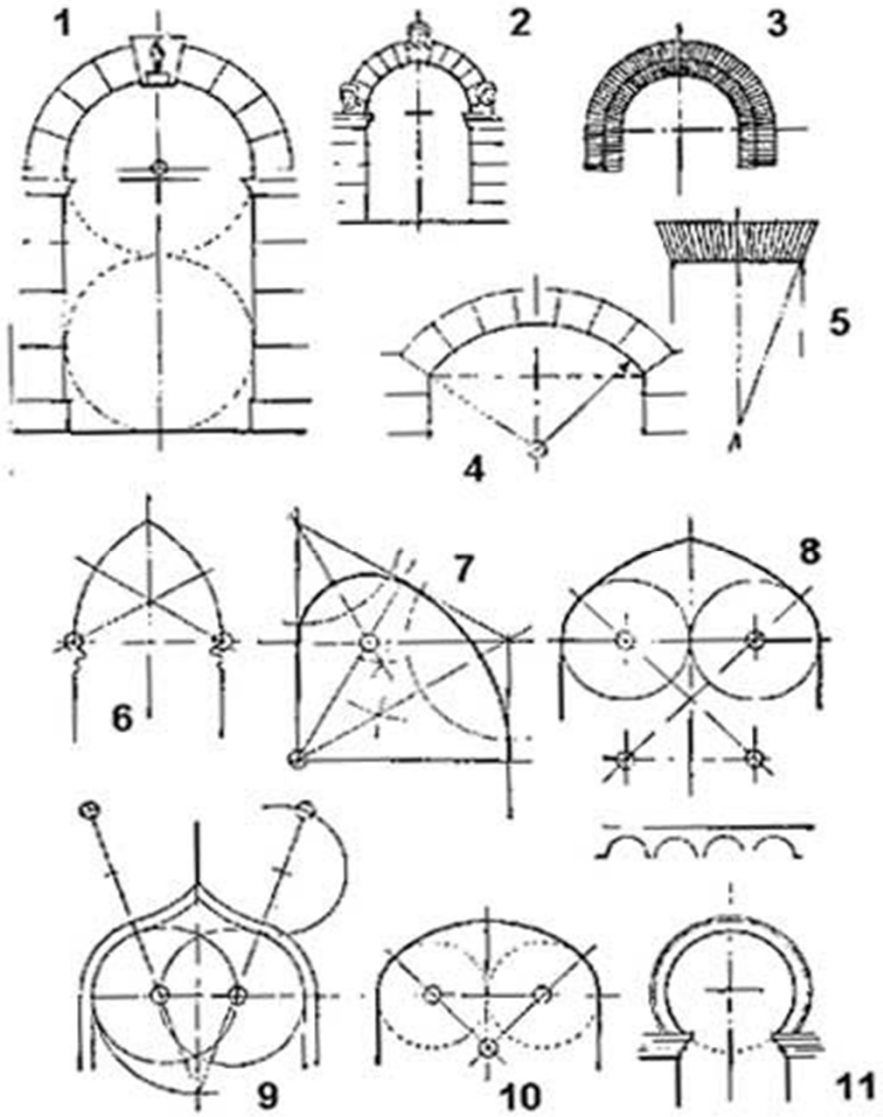




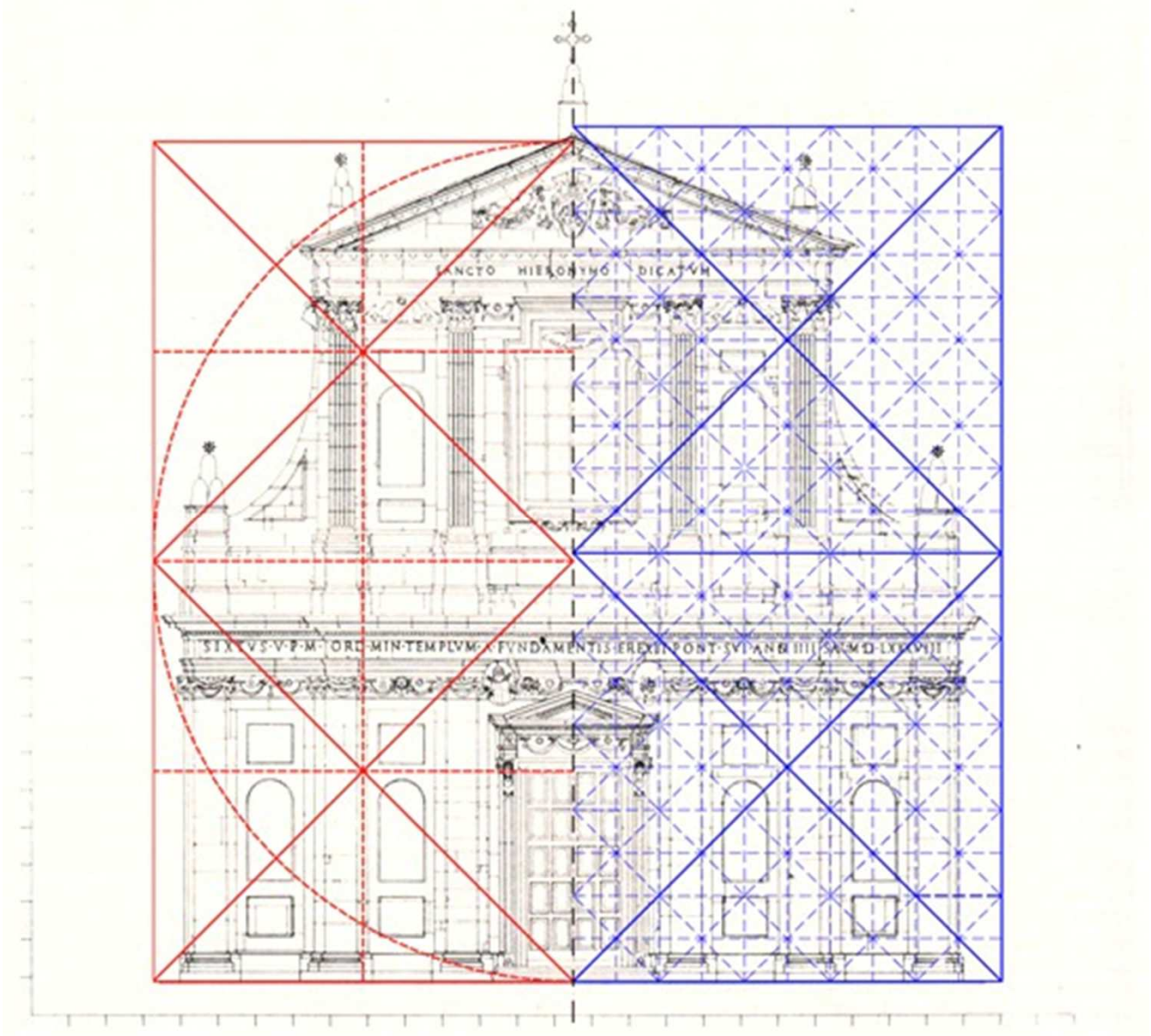
con stili diversi…



e componenti anche diversi…



...e pattern da applicare



roadmap

- Ruolo e usi (della descrizione) di un'architettura
- Viste, stili e tipi
- Tipi di viste
  - strutturali
  - comportamentali
  - logistiche
- Vista d'insieme

problemi "architettonici"

- Per le prestazioni
  - decomposizione in processi cooperanti
  - controllo del volume di comunicazioni e di accessi ai dati
  - identificazione di colli di bottiglia
- Per la modificabilità e la portabilità
- Per il rilascio incrementale
  - semplicità delle relazioni di dipendenza
- Per la sicurezza
  - controllo delle relazioni d'uso e delle comunicazioni tra le parti
  - identificazione di parti vulnerabili da attacchi esterni
  - introduzione di componenti fidate

architettura software

- [...] la progettazione e la descrizione della struttura complessiva del sistema emerge come un nuovo tipo di problema. Queste questioni strutturali includono l'organizzazione di massima e la struttura del controllo; i protocolli di comunicazione, sincronizzazione e accesso ai dati... [Garlan & Shaw 1993]
- L'AS è l'organizzazione di base di un sistema, espressa dai suoi componenti, dalle relazioni tra di loro e con l'ambiente, e i principi che ne guidano il progetto e l'evoluzione [ANSI/IEEE 1471–2000, Recommended Practice for Architecture Description of Software-Intensive Systems, il meno normativo]

[superseded by ISO/IEC/IEEE 42010:2011, *Systems and software engineering — Architecture description*]

ancora l'architettura

- L'architettura software è l'insieme delle strutture del sistema, costituite dalle componenti software, le loro proprietà visibili e le relazioni tra di loro [Bass, Clemens & Kazman 1998]
  - Le "proprietà visibili" di una componente definiscono le assunzioni che le altre componenti possono fare su di essa, come servizi forniti, prestazioni, uso di risorse condivise, trattamento di malfunzionamenti, ecc.
- L'architettura di un sistema software (in breve architettura software) è la struttura del sistema, costituita dalle parti del sistema, dalle relazioni tra le parti e dalle loro proprietà visibili

in altre/altrui parole

- L'architettura
  - definisce la struttura del sistema sw
  - specifica le comunicazioni tra componenti
  - considera aspetti non funzionali
  - è un'astrazione
  - è un artefatto complesso → viste

e ancora…

- Il middleware
  - come tubi o fili elettrici edificio
  - permette di collegare componenti
  - con topologie ben note (uno a uno, uno a molti…)
- invisibile all'utente che usa l'applicazione…
  - finché non fallisce!

comunicare tra/per le parti interessate

- tra architetto e cliente: negoziazione di requisiti
- tra architetto e progettisti: negoziazione di risorse (a tempo d'esecuzione)
- per progettisti di sistemi interagenti: definizione di interfacce e protocolli
- per progettisti: definizione di vincoli e possibilità
- per verificatori e integratori: specifica del comportamento delle parti
- per manutentori: base per l'analisi delle conseguenze di modifiche
- per gestori: base per la formazione di gruppi di lavoro, decomposizione del lavoro, allocazione delle risorse, traccia dello stato di avanzamento

ancora sull'uso di AS

- Educazione
  - di nuovi membri, analisti esterni, nuovo architetto
- Analisi
  - delle prestazioni
  - della sicurezza
  - della disponibilità (availability)
  - …

Viste sull'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

Documentazione di un'AS

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

Tipi di vista (Viewtypes)

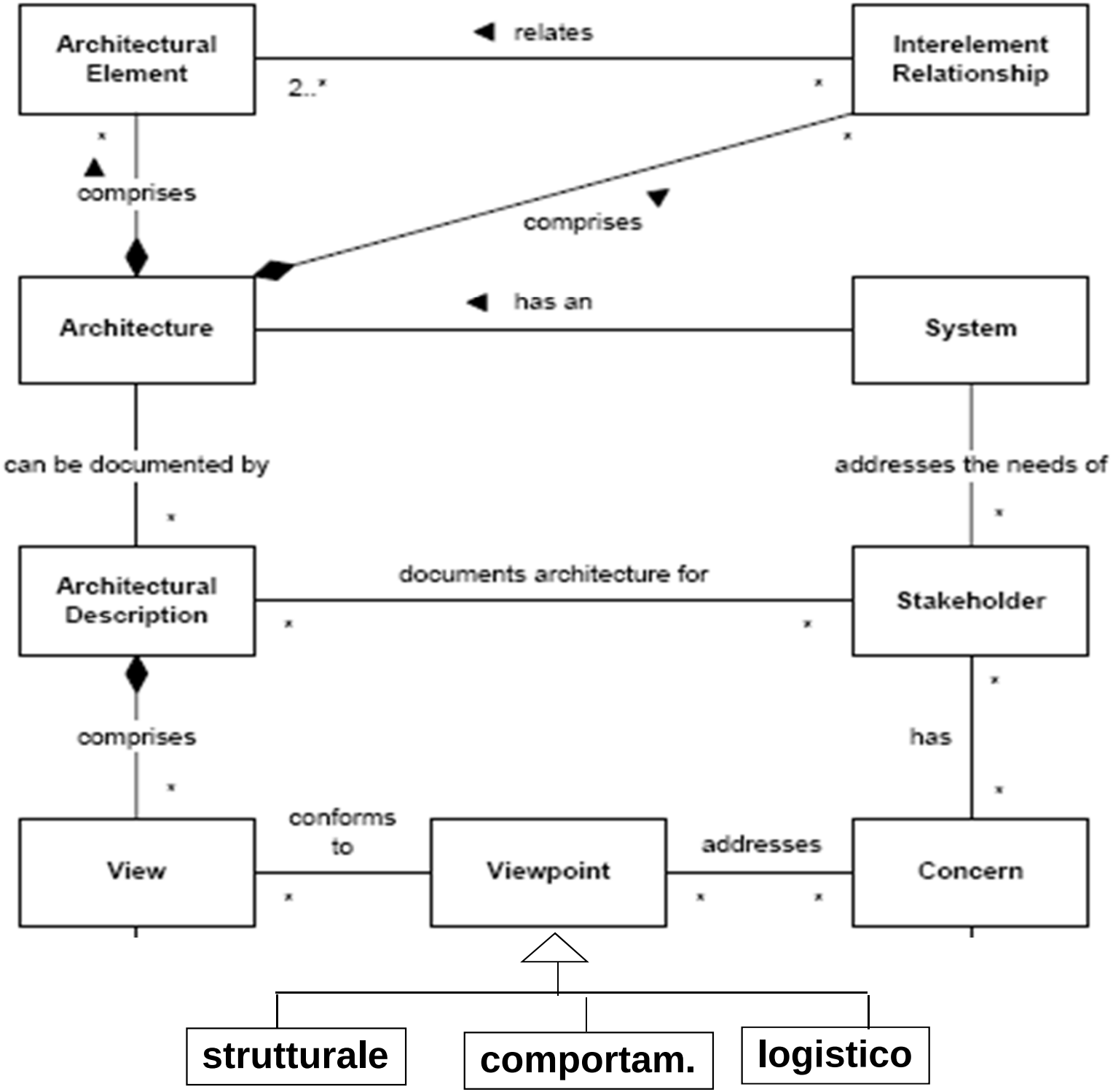
- Un tipo di vista definisce i tipi degli elementi e delle relazioni usati per descrivere l'AS da un particolare punto di vista
- Tre punti di vista simultanei sul sistema
  - la struttura come insieme di unità di realizzazione
    - tipo di vista strutturale
  - la struttura come insieme di unità con comportamenti e interazioni, a tempo d'esecuzione
    - tipo di vista comportamentale (component-and-connector (C&C))
  - le relazioni con strutture diverse dal software nel contesto del sistema
    - tipo di vista 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.
- Proprietà
  - Di elementi e relazioni
- Usi
  - Utilità della vista
- Notazione
  - Costrutti UML utilizzati


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

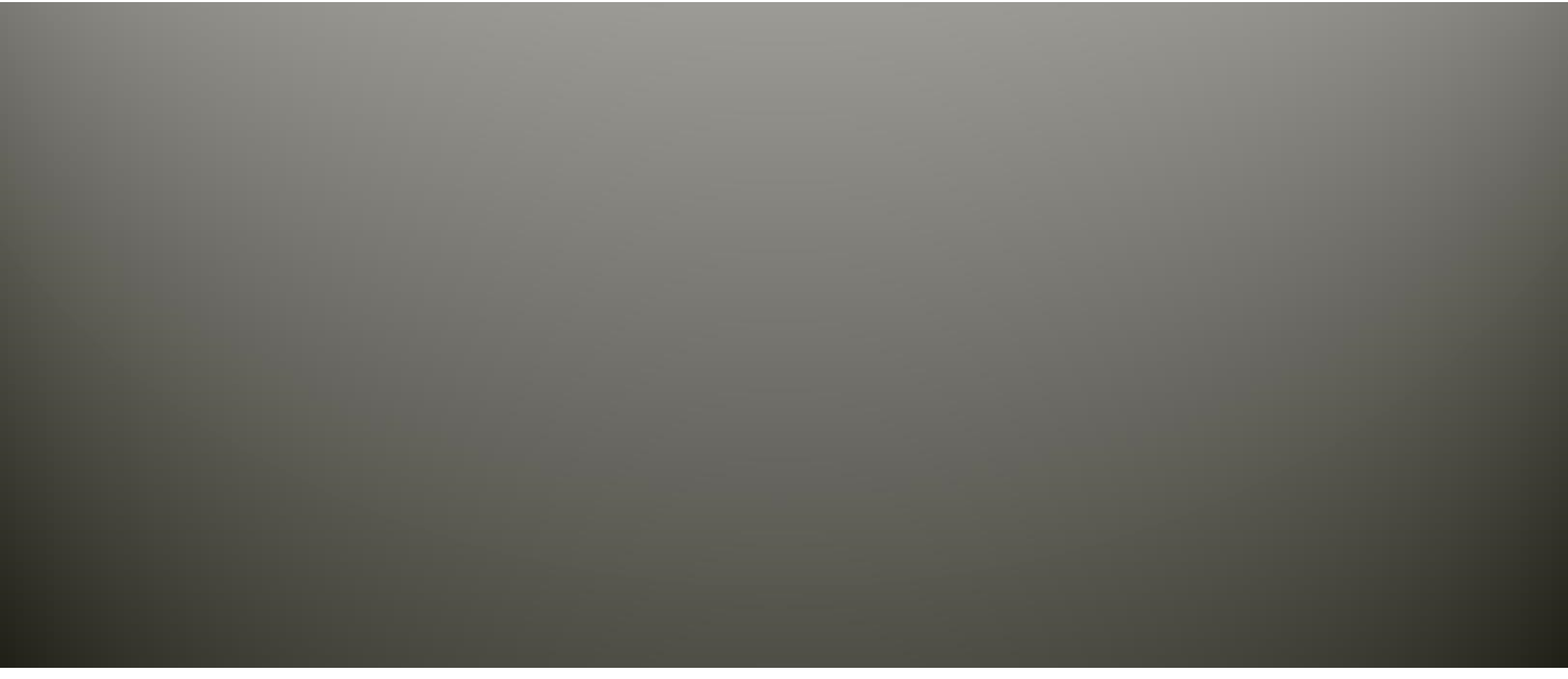
Vista d'insieme

- Organizzazione della documentazione
  - Guida alla documentazione
    - descrizione delle parti
    - elenco delle viste
    - usi per le parti interessate
  - Schema di presentazione di una vista
- L'architettura
  - Descrizione del sistema
  - Corrispondenze tra viste
  - Elenco degli elementi, con riferimento alla definizione
  - Glossario e lista degli acronimi
- Motivazioni
  - Contesto, vincoli di progettazione, giustificazioni delle scelte

Schema di presentazione di una vista

- Presentazione principale
- Catalogo degli elementi
  - Elementi e proprietà
  - Relazioni e proprietà
  - Interfacce degli elementi
  - Comportamento degli elementi
- Diagramma di contesto
- Guida alla variabilità
- Contesto dell'architettura
  - Giustificazioni delle scelte
  - Risultati delle analisi
  - Assunzioni
- Viste correlate

Viste di tipo strutturale



Viste di tipo strutturale

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

Viste di tipo strutturale (2)

- Usi
  - costruzione
    - la vista può fornire lo schema del codice e directory e file sorgente hanno una struttura corrispondente
  - analisi
    - tracciabilità dei requisiti
    - analisi d'impatto per valutare eventuali modifiche
  - 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 packages + notaz. per relazioni

Notazione: packages

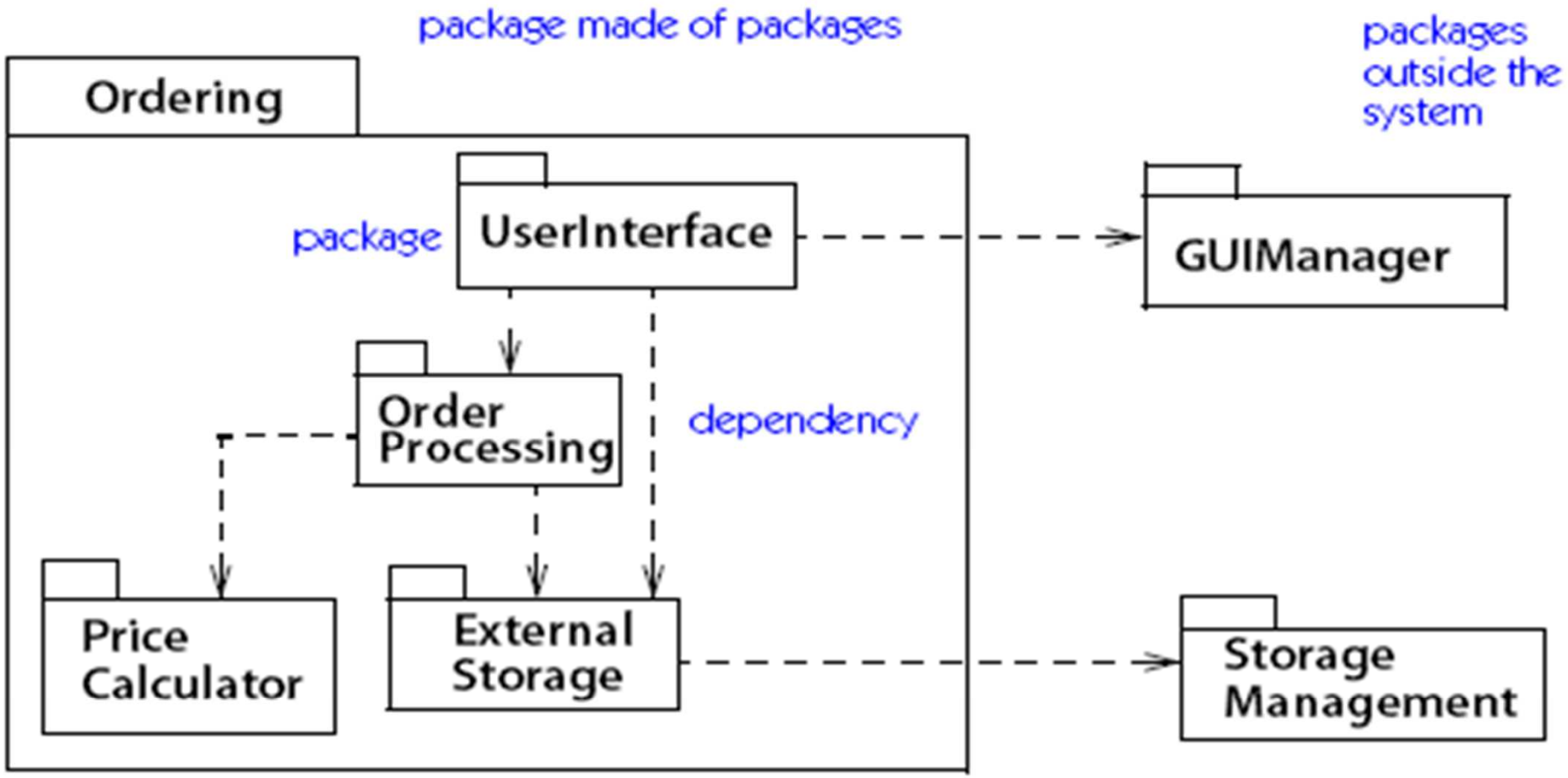
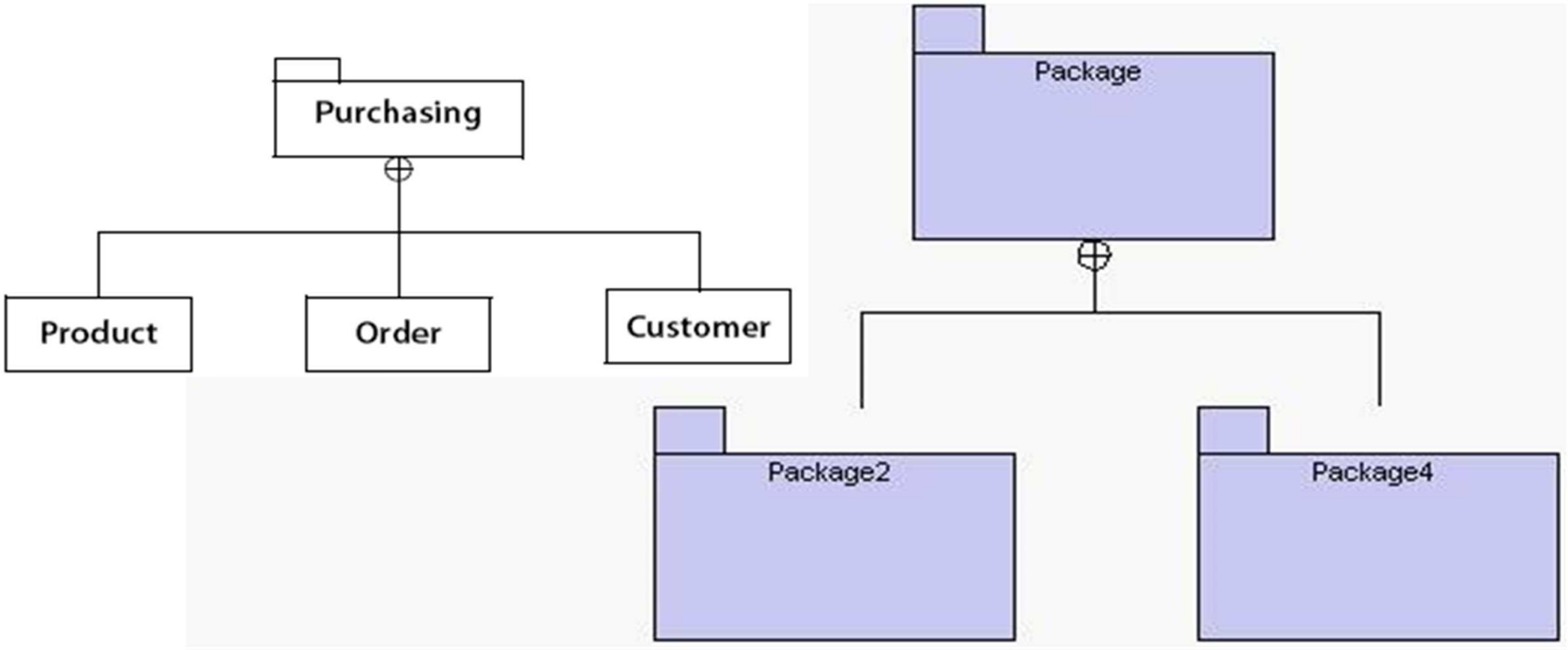


Figure 14-203. *Packages and their relationships*

Vista strutturale di decomposizione

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

Notazione (oppure 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
    - un modulo che segnala un errore funziona correttamente indipendentemente da cosa fa il modulo che riceve la segnalazione, per cui lo invoca, ma non lo usa (Non è suo cliente in una dipendenza).
  - abilita lo sviluppo incrementale e il rilascio di sottosistemi utili
  - cicli permessi ma pericolosi
- Usi
  - pianificazione di sviluppo incrementale, di estensioni e ritagli del sistema, del testing incrementale
  - analisi d'impatto

Notazione

Uso

Vista strutturale 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 esplicitamente detto)
- Usi
  - modificabilità e portabilità
  - controllo della complessità
  - analisi d'impatto

Notazione

Un esempio noto: vista strutturale o C&C ?

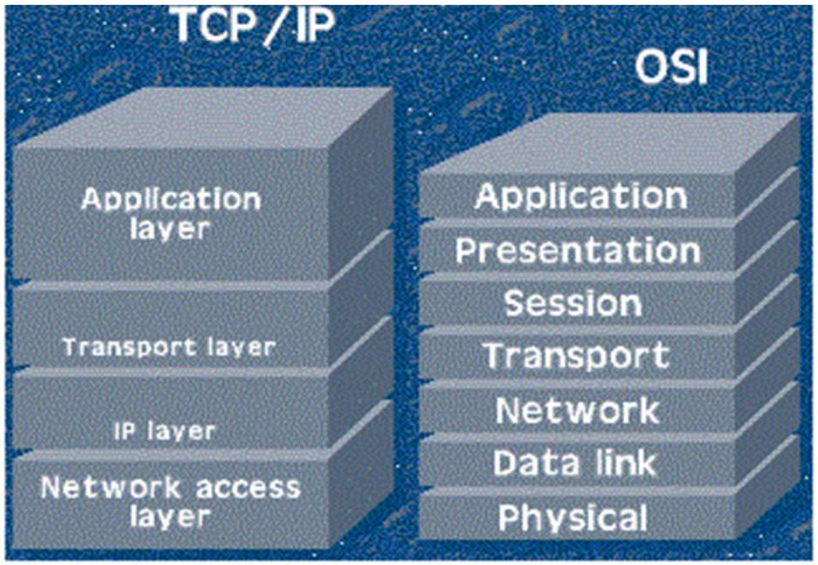
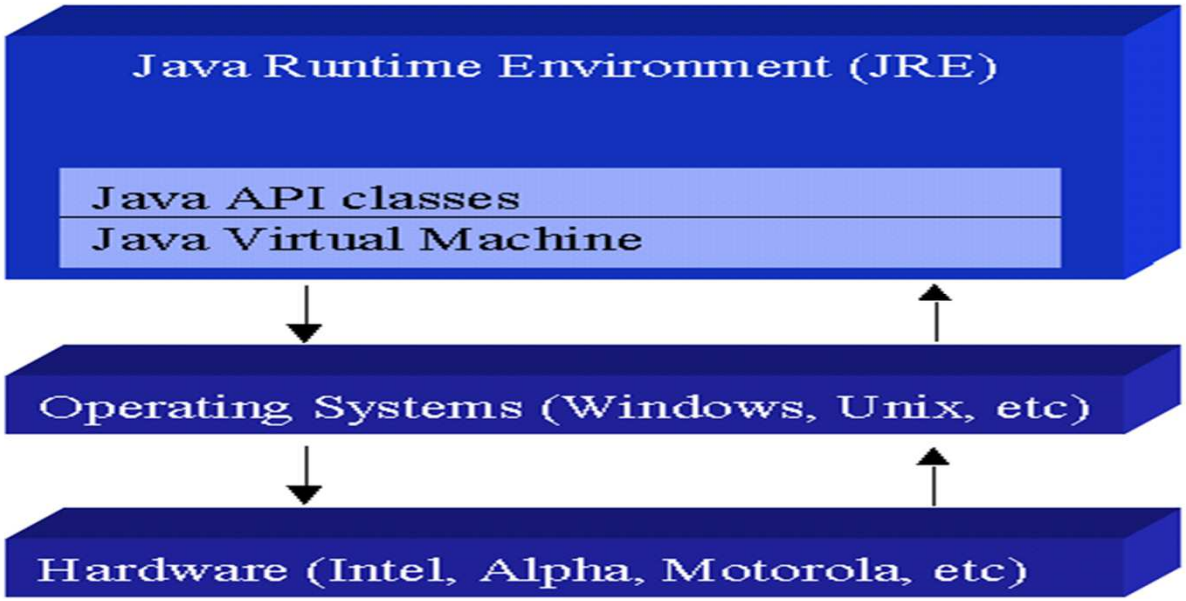


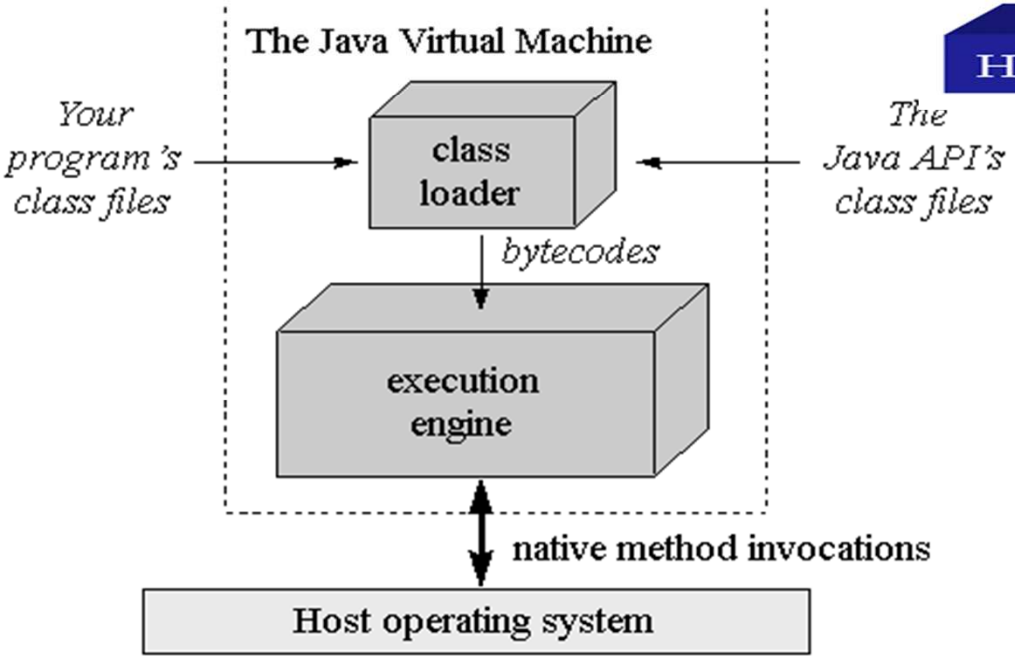
Fig. 1.2.3 - Strati OSI e strati dell'architettura TCP/IP

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)

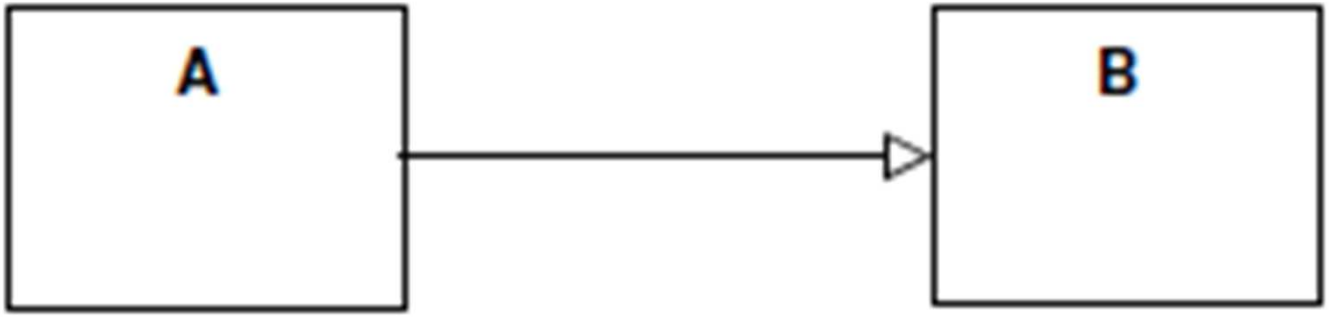
Fig. 1.2.3 - Strati OSI relativi all'architettura TCP/IP

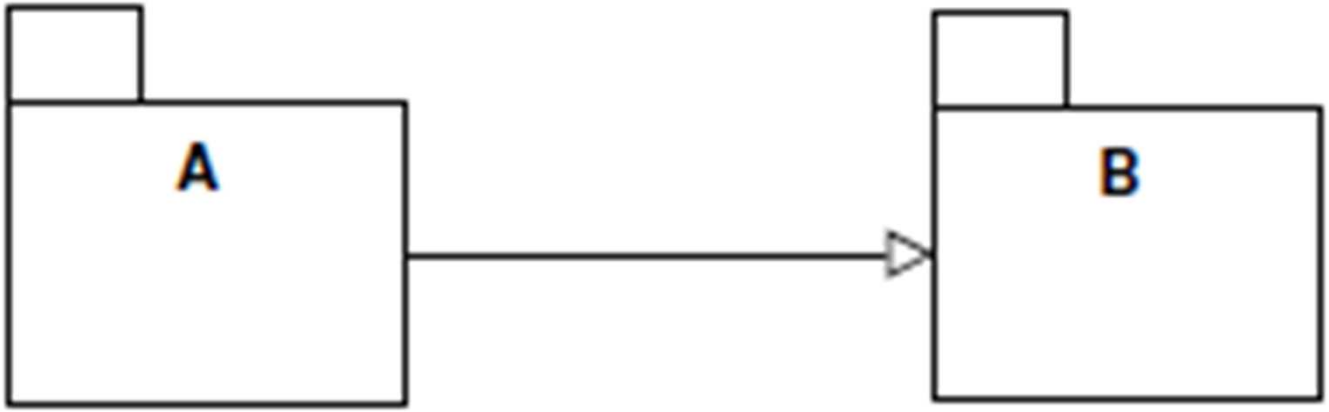
Un altro esempio : vista strutturale o C&C ?





Vista strutturale di generalizzazione





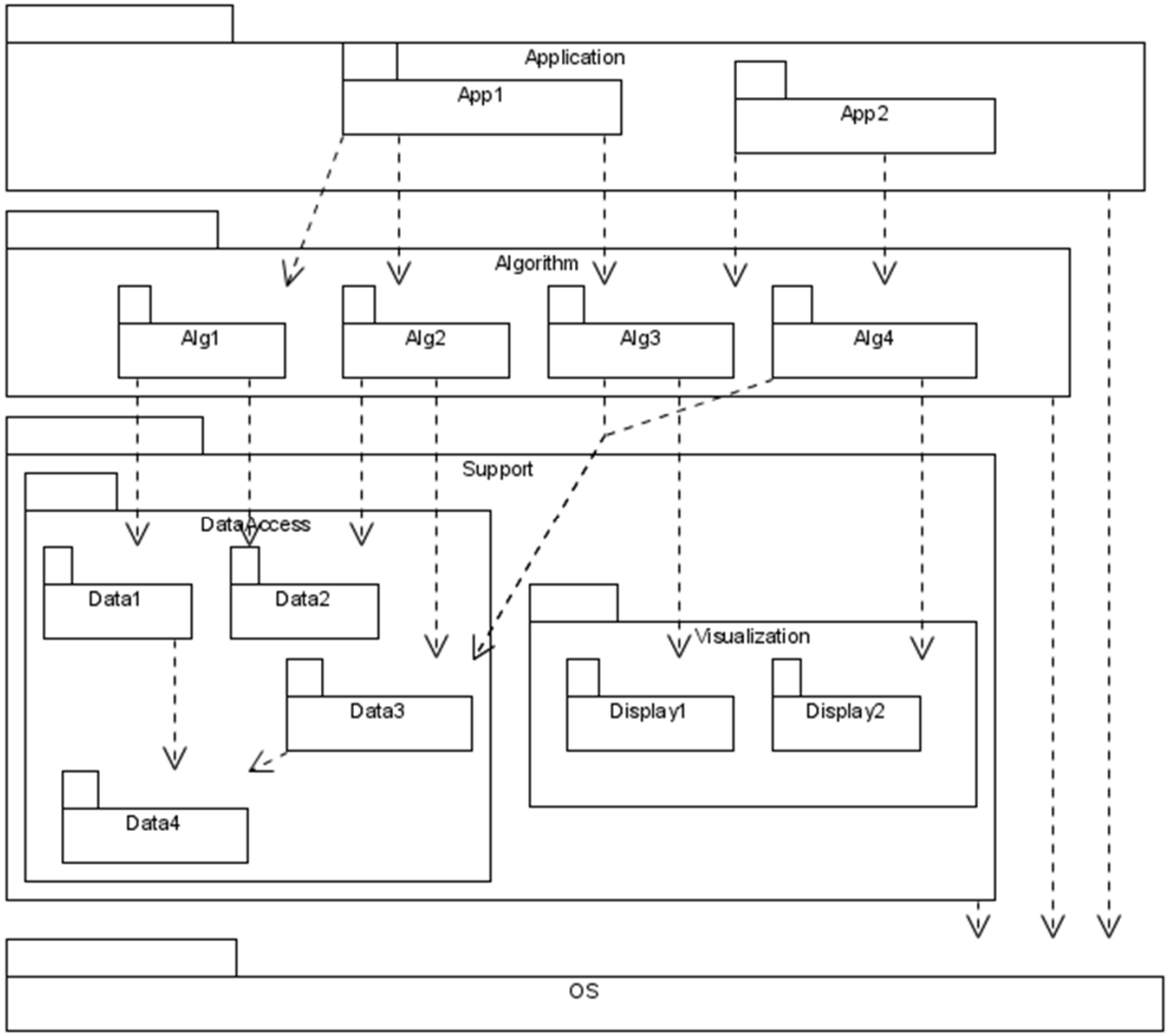
- Framework

Esercizio

Module	Module Type	Invokes Module
APP-1	Application	ALGO-1, ALGO-2, ALGO-3
APP-2	Application	ALGO-3, ALGO-4
ALGO-1	Algorithm	DATA-1, DATA-2
ALGO-2	Algorithm	DATA-2, DATA-3
ALGO-3	Algorithm	DATA-3, DISP-1
ALGO-4	Algorithm	DATA-3, DISP-2
DATA-1	Data Access	DATA-4
DATA-2	Data Access	
DATA-3	Data Access	DATA-4
DATA-4	Data Access	
DISP-1	Display Output	
DISP-2	Display Output	
All Modules use Modules in the Operating System		

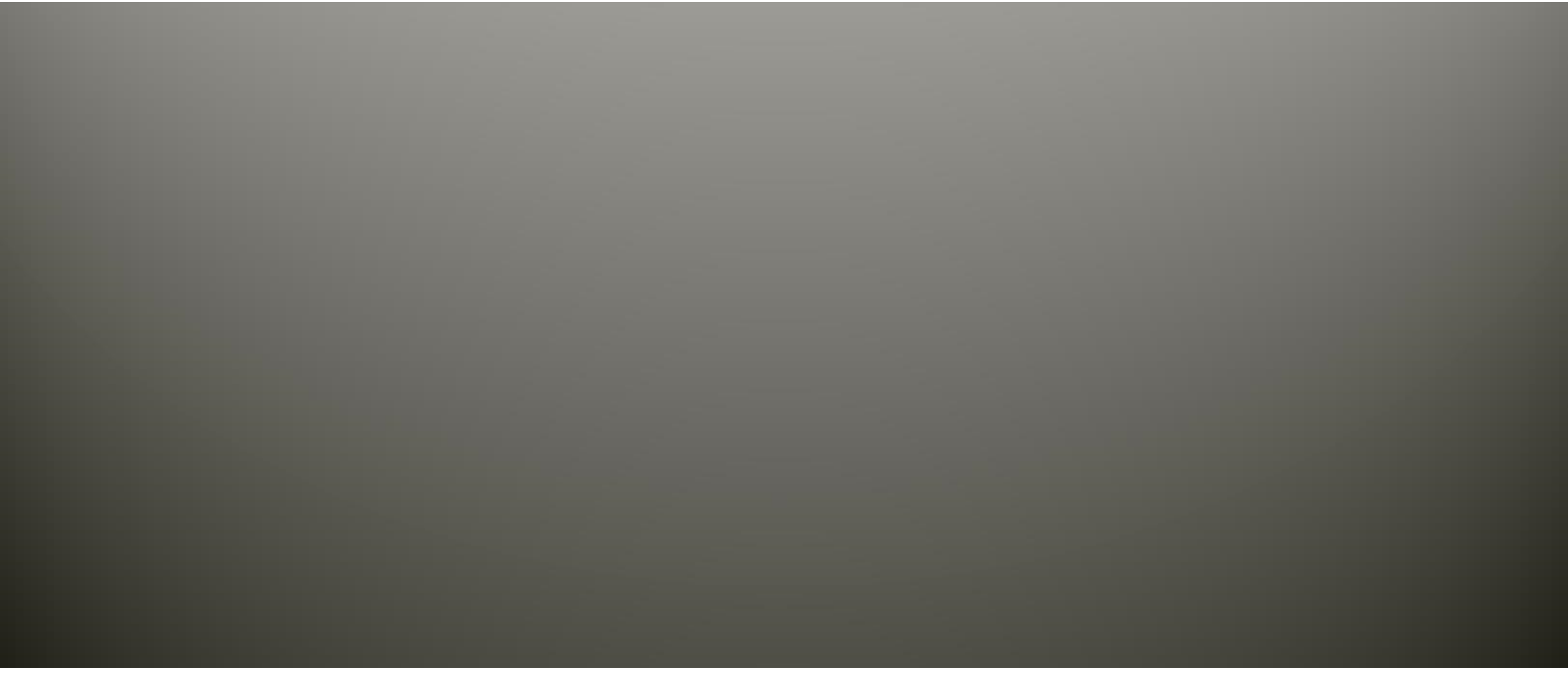
a.Sketch a graphical module view that illustrates the uses relationships implied by this table. Could these modules be grouped into packages of modules? Could these modules (or the packages) be grouped into layers? If so, try to incorporate your groupings into your view.
b.Sketch the dependency tree for the APP-1 application.
c.Suppose that we want to create a separate runtime component for each of the applications. Which modules would participate when the APP-2 component is running?
d.Assume that the original plan was to have the data access modules use the file system of the operating system for data management. Now, suppose that we decide to purchase a separate database system for data storage and retrieval. The database, of course, uses modules in the operating system. How would you incorporate this change into your graphical module view?

Soluzione


Application
App1
App2
Algorithm
Alg1
Alg2
Alg3
Alg4
Support
DataAccess
Data1
Data2
Data3
Data4
Visualization
Display1
Display2
OS

Viste di tipo comportamentale

Aka C&C aka componenti e conettori



Viste di tipo comportamentale

- 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, …
- Relazioni
  - connessione di ruoli a porti

Viste di tipo comportamentale(2)

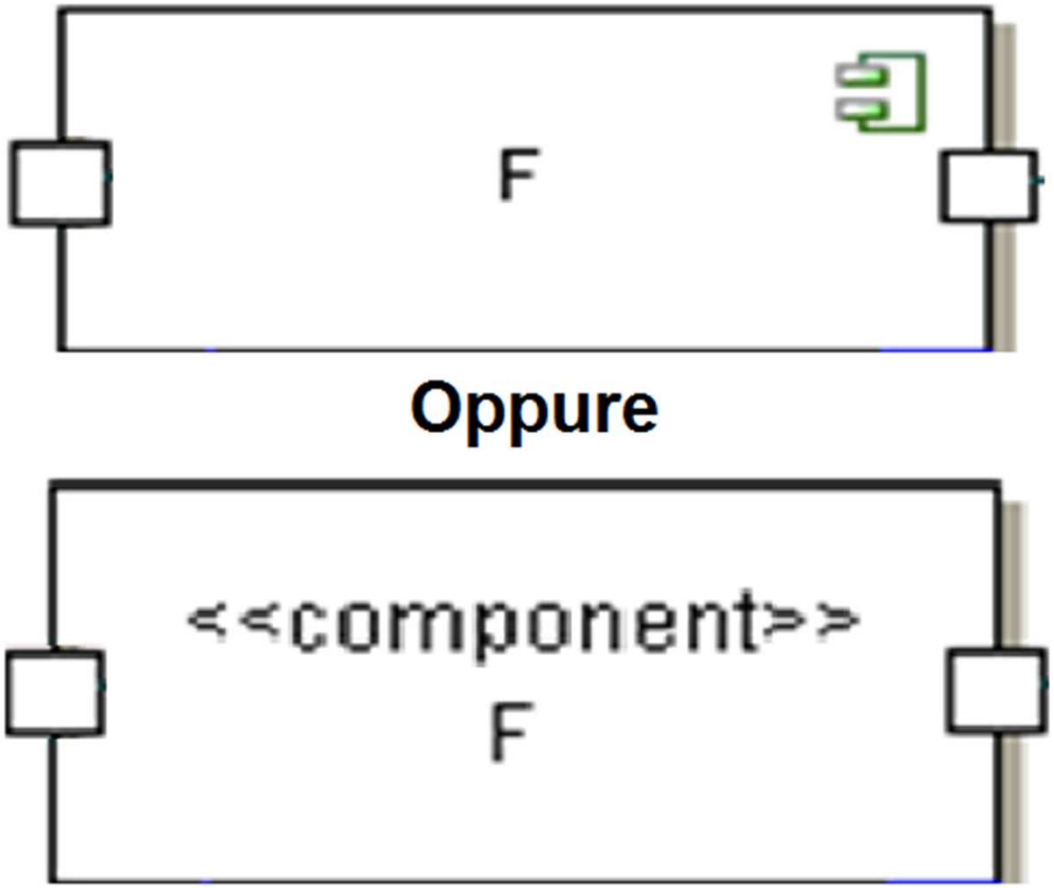
- Usi
  - analisi delle caratteristiche di qualità a tempo d'esecuzione
    - prestazioni, affidabilità, disponibilità, sicurezza, …
  - comunicazione della struttura del sistema in esecuzione
    - flusso dei dati, dinamica, parallelismo, replicazioni, …

Vista comportamentale: notazione UML

component diagram

A diagram that shows the definition, internal structure, and dependencies of component types. There is no sharp line between component diagrams and general class diagrams.

Componenti: notazione UML



Porti

- I quadratini sono “porti”, che identificano i punti di interazione di un classificatore (classe, componente), e sono caratterizzati dalle interfacce che forniscono o richiedono.
- Possono avere associata una molteplicità.

1..n

Porti

port

A structural feature of a classifier that encapsulates interaction between the contents of the classifier and its environment.

See interface, structured classifier, structured part.

Semantics

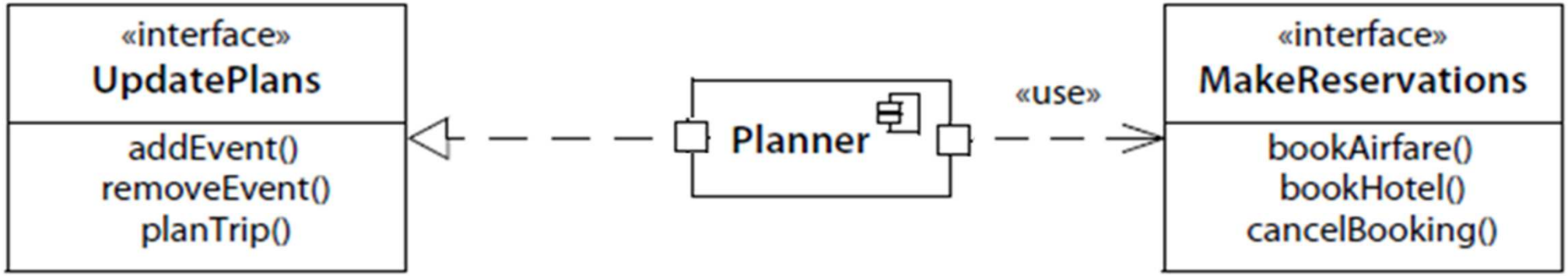
A port is a connection point between a classifier and its environment. Connections from the outside world are made to ports according to provided and required interfaces declared on a port. The outside clients of the classifier have no visibility or direct interaction with the contents or implementation of the classifier. When a classifier is implemented, ports on internal parts are connected to its external ports in accordance with the declared interfaces. The use of ports permits the internal structure of a classifier to be modified without affecting the external clients, provided the interfaces on the ports are correctly supported. Ports permit encapsulated parts to be "wired together" to form the implementation of a classifier.

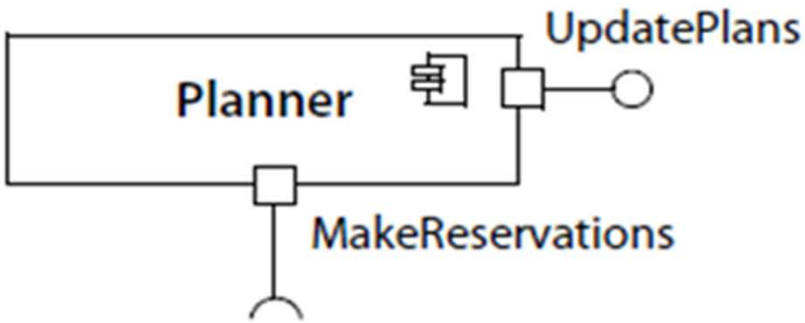
The behavior of a port is specified by its provided and required interfaces.

Ports are created as part of the instantiation of their classifier and destroyed with it. They may not be created or destroyed during the life of an object.

A port may be specified with multiplicity. If the multiplicity is not a single inte-

Interfacce

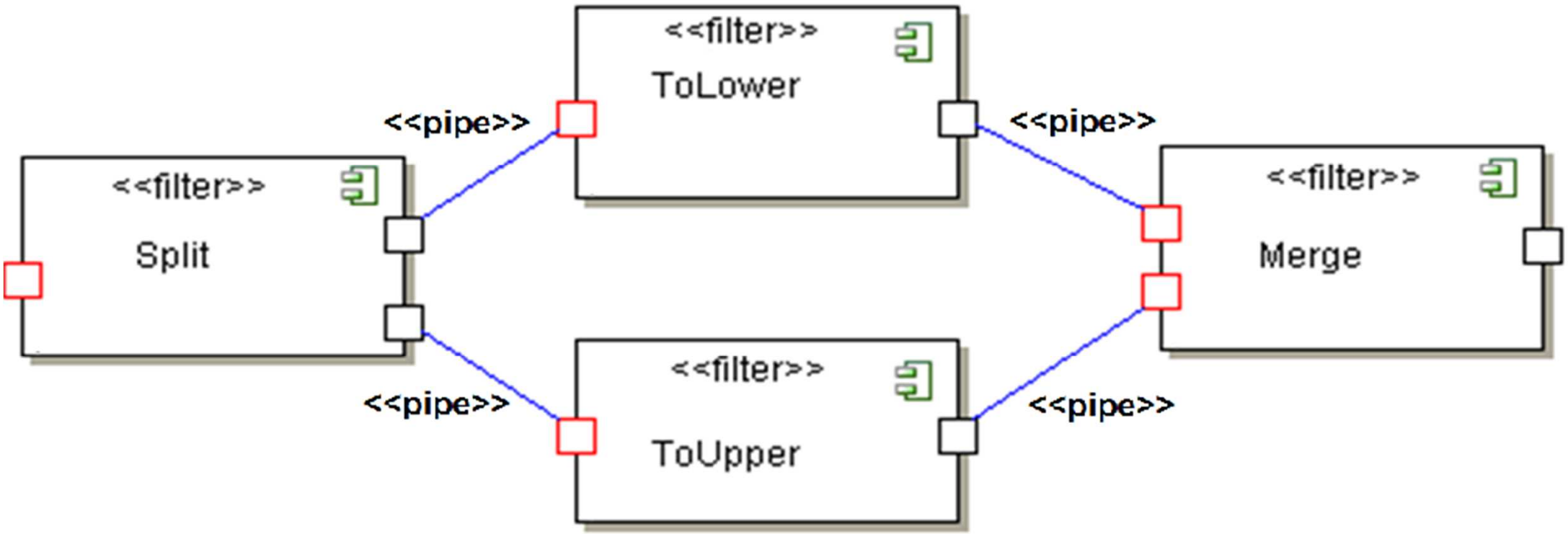

«interface»
UpdatePlans
addEvent()
removeEvent()
planTrip()
Planner
«use»
«interface»
MakeReservations
bookAirfare()
bookHotel()
cancelBooking()


Planner
UpdatePlans
MakeReservations

Stili comportamentali: condotte e filtri

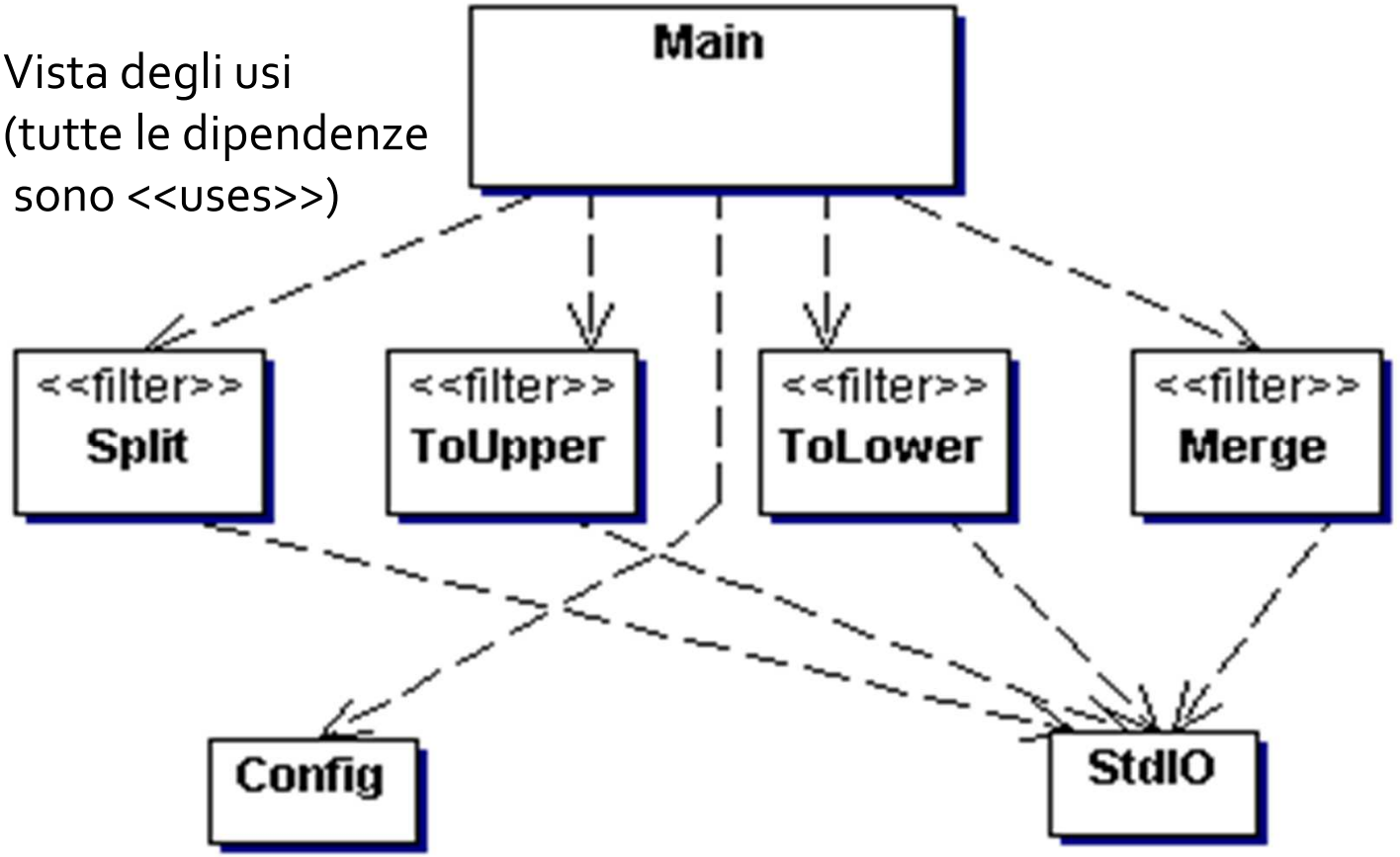
- Componenti
  - filtro: trasforma uno o più flussi di dati dai porti d'ingresso in uno o più flussi sui porti d'uscita
- Connettori
  - condotta (pipe): canale di comunicazione unidirezionale bufferizzato che preserva l'ordine dei dati dal ruolo d'ingresso a quello d'uscita
- Usi
  - pre-elaborazione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

Condotte e filtri: esempio



Strutturale vs comportamentale

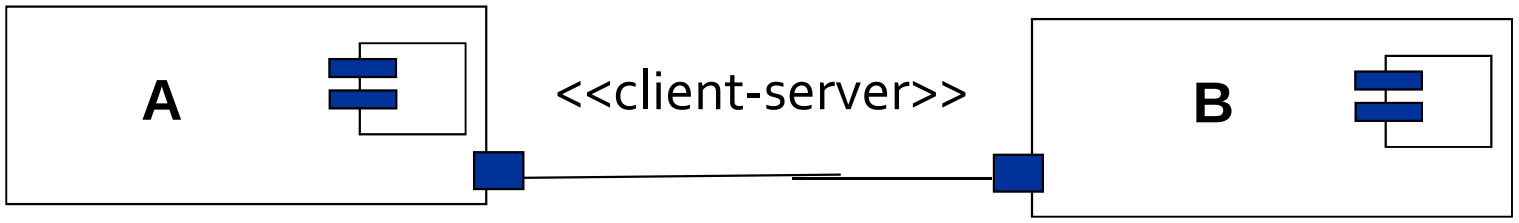
Vista degli usi
(tutte le dipendenze
sono <<uses>>)

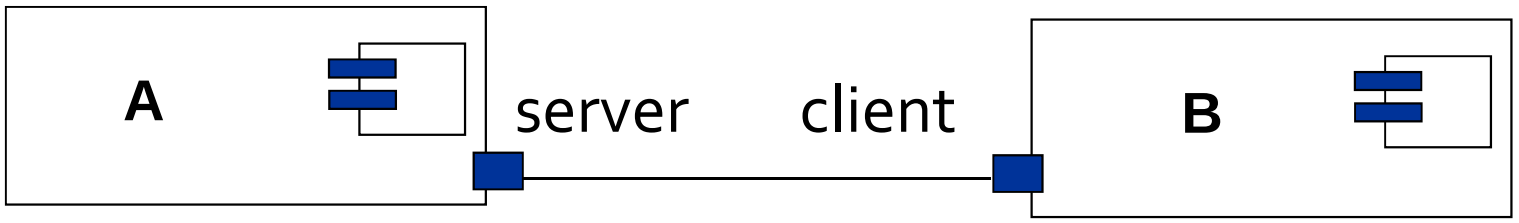


- i filtri si chiamano tra loro, ma non si usano…
- e il main li configura per metterli in comunincazione via StdIO (realizzazione del connettore pipe)
- suggerisce anche una vista a strati…

Stili comportamentali: client server

- Cliente-servente (Client-server)
  - le componenti interagiscono chiedendo servizi ad altre componenti
  - comunicazioni appaiate: richiesta e fornitura del servizio





Stili comportamentali: Publish-subscribe

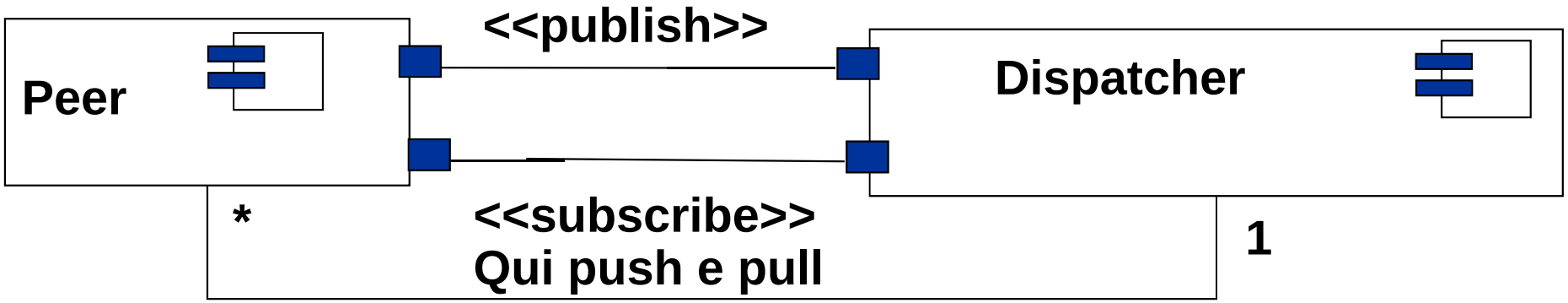
- le componenti interagiscono annunciando eventi: ciascuna componente si "abbona" a classi di eventi rilevanti per il suo scopo
- Ciascuna componente, volendo, può essere sia produttore che consumatore di eventi
- disaccoppia produttori e consumatori di eventi e favorisce le modifiche dinamiche del sistema

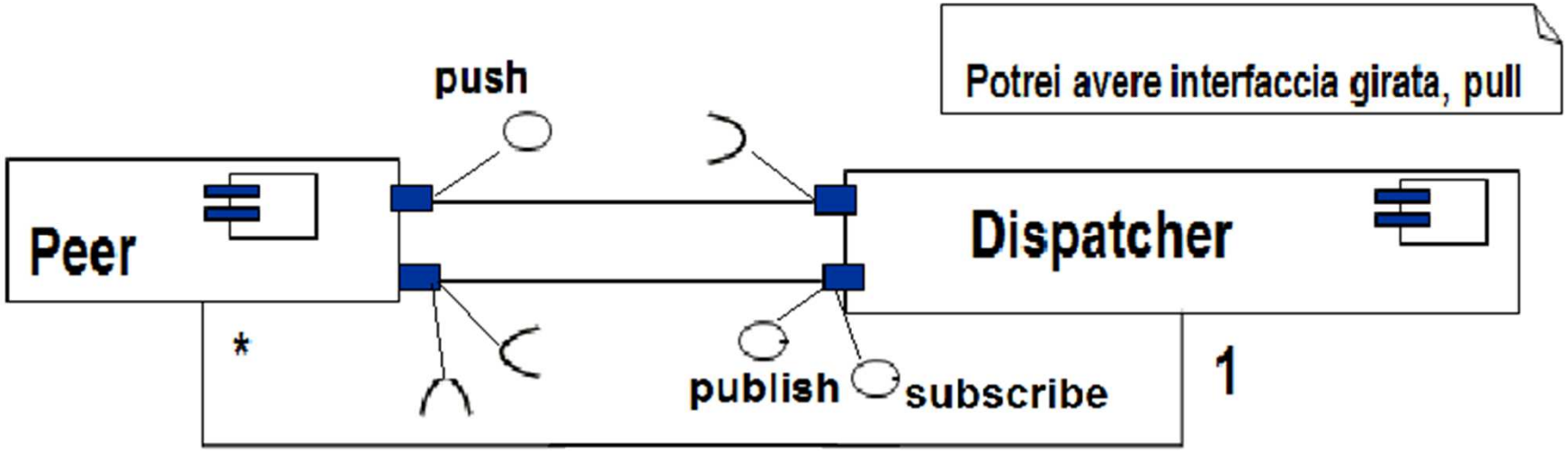
Publish-subscribe

- In questo stile, mittenti e destinatari di messaggi dialogano attraverso un tramite, che può essere detto dispatcher o broker. Il mittente di un messaggio (detto publisher) non deve essere consapevole dell'identità dei destinatari (detti subscriber); esso si limita a "pubblicare" (in inglese to publish) il proprio messaggio al dispatcher. I destinatari si rivolgono a loro volta al dispatcher "abbonandosi" (in inglese to subscribe) 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. Per 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.

Publish-subscribe

Operazioni: subscribe, unsubscribe, publish, notify (push), letMeKnow (pull)



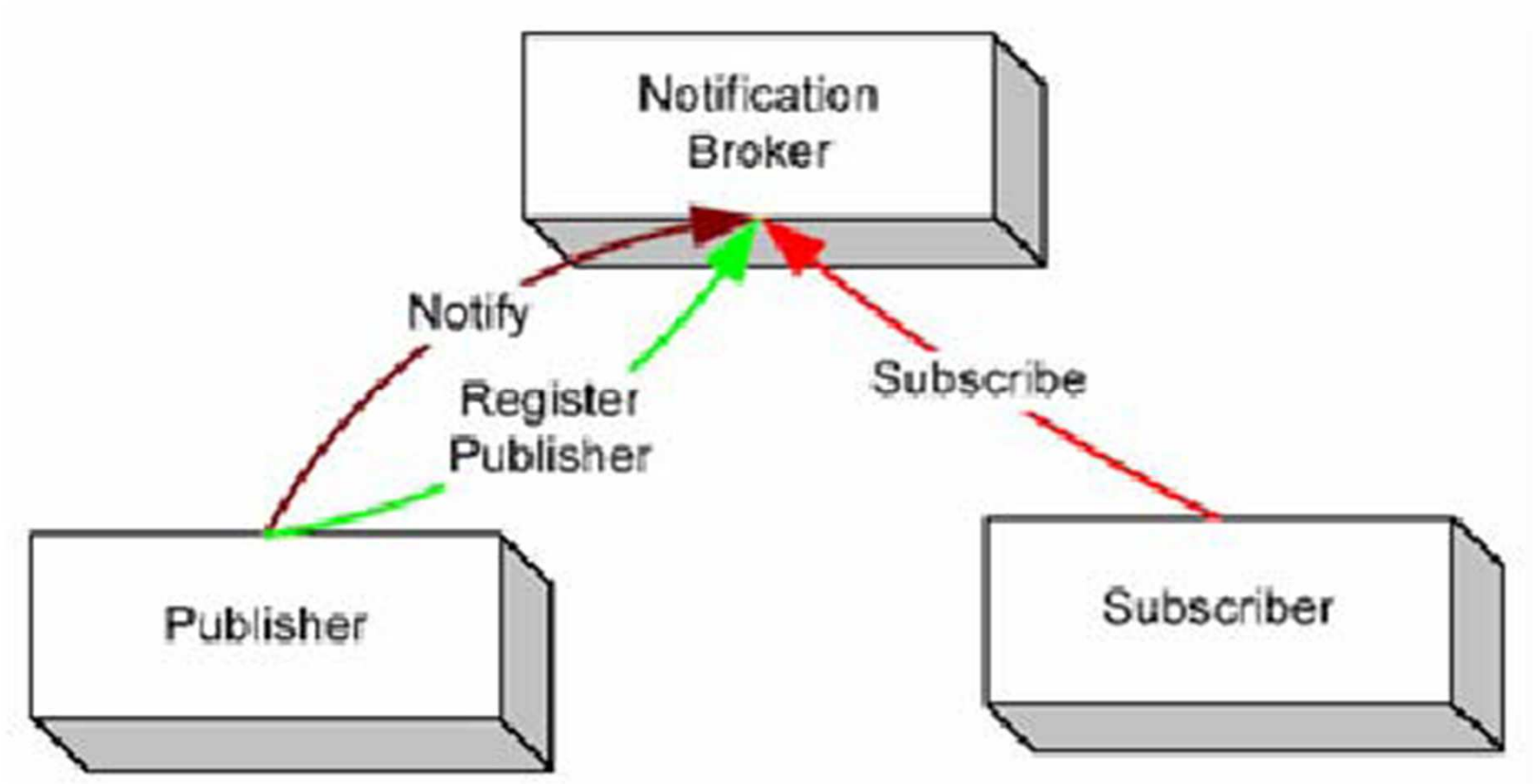


Publish-subscribe: middleware e reti

- 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.

Publish-subscribe e web services (con notazione errata)

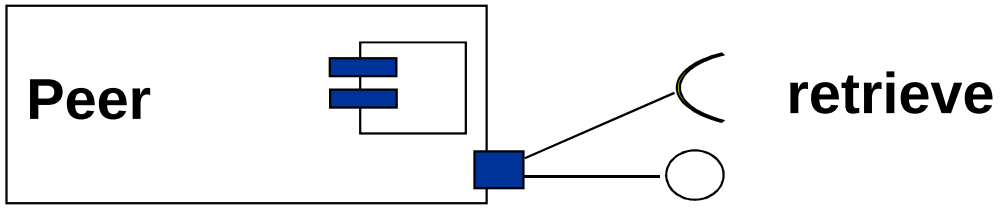


Web services

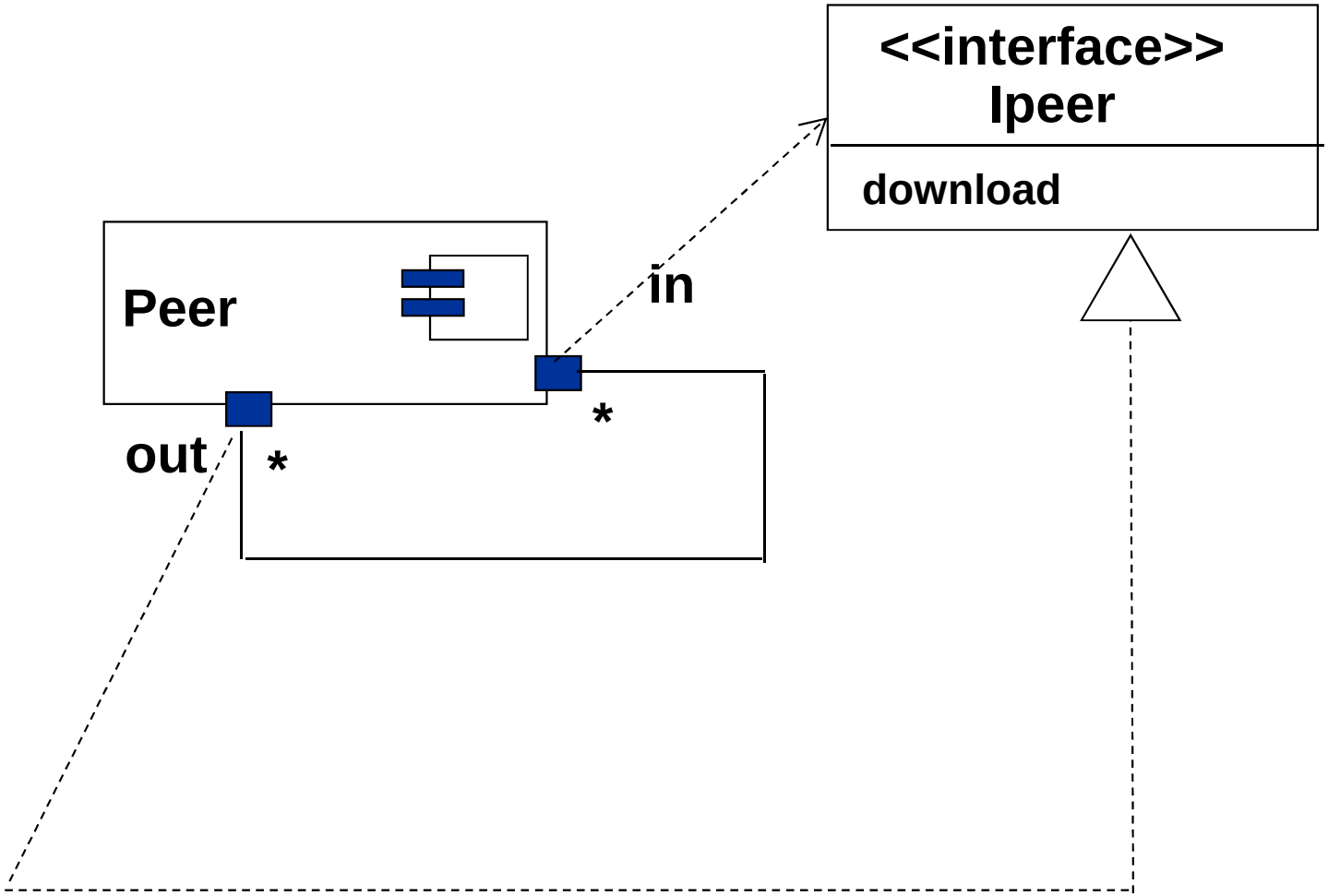
- Il publisher preliminarmente deve aver contattato il broker mediante il metodo RegisterPublisher dell'interfaccia del broker dichiarando di essere disposto a pubblicare notifiche di eventi relativi a specifici topics.
- Una volta registratosi presso il broker, il publisher invierà le notifiche degli eventi che verranno prodotti soltanto al broker presso cui si è registrato. Il broker sarà in grado di capire verso quali consumer dirigere le notifiche di evento poiché manterrà la lista di tutte le sottoscrizioni.
- Spetterà poi ai brokers contattare i consumers sottoscritti inviando copie del messaggio di notifica ricevuto dal publisher. La presenza di un broker non modifica la semantica comportamentale del subscription manager.

Altri stili comportamentali

- Da pari a pari (peer to peer)
  - scambio di servizi alla pari
  - i connettori non sono asimmetrici, come in cliente-servente



P2P



Model–View–Controller (MVC)

- Architectural pattern. The pattern isolates business logic from input and presentation, permitting independent development, testing and maintenance of each.
- Model
  - Is the domain-specific representation of the data upon which the application operates. When a model changes its state, it notifies its associated views so they can refresh.
  - Many applications use a persistent storage mechanism (such as a database) to store data. MVC does not specifically mention the data access layer because it is understood to be underneath or encapsulated by the model. Models are not data access objects although in very simple apps, with little
- View
  - Renders the model into a form suitable for interaction, typically a user interface element. Multiple views can exist for a single model for different purposes.
- Controller
  - Receives input and initiates a response by making calls on model objects.

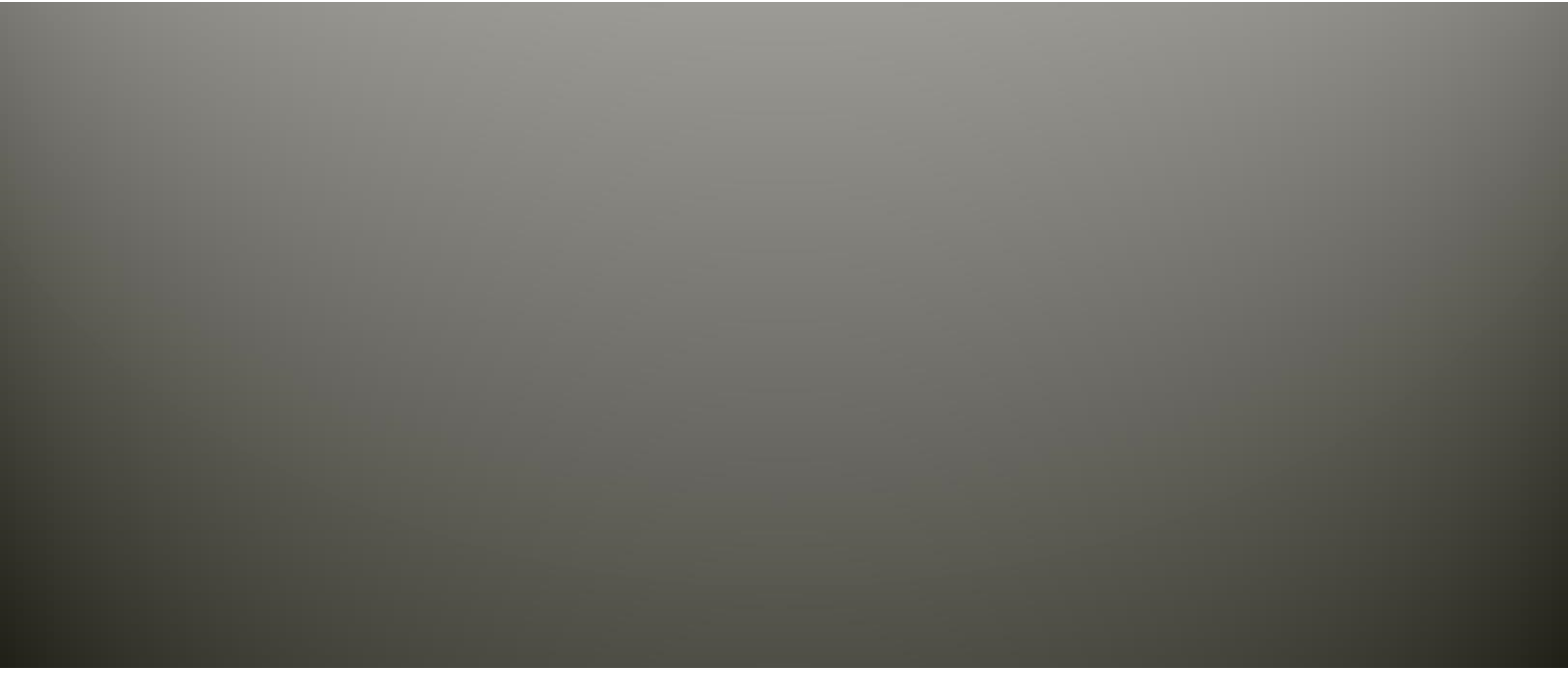
Processi comunicanti

- un non stile…
- vari connettori: sincronizzazione, scambio di messaggi, …

Stereotipi per i connettori tra componenti (diagramma C&C)

- clientServer
- dataAccess (specializzazione di clientServer)
- pipe
- peer2peer
- publish
- subscribe

Viste di tipo logistico



Viste di tipo logistico

- Elementi
  - software: moduli, componenti e connettori
  - dell'ambiente: hardware, struttura di sviluppo, file system
- Relazione
  - elemento software allocato a elemento dell'ambiente
- Proprietà
  - richieste dal software, fornite dall'ambiente
- Usi
  - analisi delle prestazioni (stile di dislocazione - deployment)
  - pianificazione dello sviluppo (stile di assegnamento del lavoro)
  - gestione delle configurazioni (stile di realizzazione)

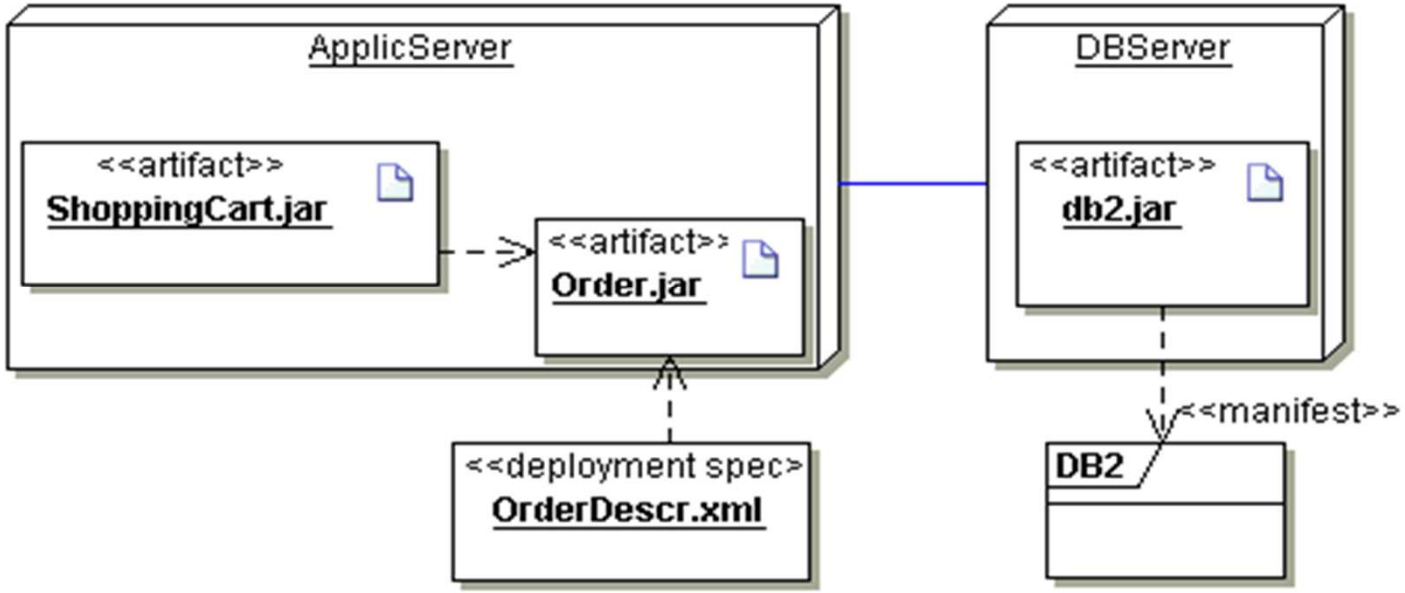
Vista logistica di dislocazione

- Elementi
  - software: componenti e connettori
  - dell'ambiente: hardwarel, ambiente di esecuzione
- Proprietà del software:
  - consumo di risorse, criticità
- Proprietà dell'ambiente:
  - CPU: frequenza di clock, numero di processori, …
  - memoria: dimensioni, velocità, dischi, …
  - canali di comunicazione: ampiezza di banda

Artifact

- An artifact is the specification of a physical piece of information that is used or produced by a software development process, or by deployment and operation of a system. Examples of artifacts include model files, source files, scripts, and binary executable files, a table in a database system, a development deliverable, or a word-processing document, a mail message.

UML: diagrammi di dislocazione



- Nodo: rappresenta un tipo di hardware; le relazioni tra nodi sono connessioni fisiche. Un nodo può anche rappresentare un abiente di esecuione
- Artefatto: informazione prodotta dallo sviluppo o esecuzione di sw
- Deployment spec: Parametri d'installazione e d'esecuzione
- Manifestazione: dipendenza verso il modello realizzato

Vista logistica di realizzazione

- Elementi
  - software: moduli
  - ambiente: elementi di configurazione (configuration items)
    - spesso gerarchici
- Usi
  - gestione dei file che corrispondono agli elementi software
  - gestione delle versioni

Vista di realizzazione, esempio

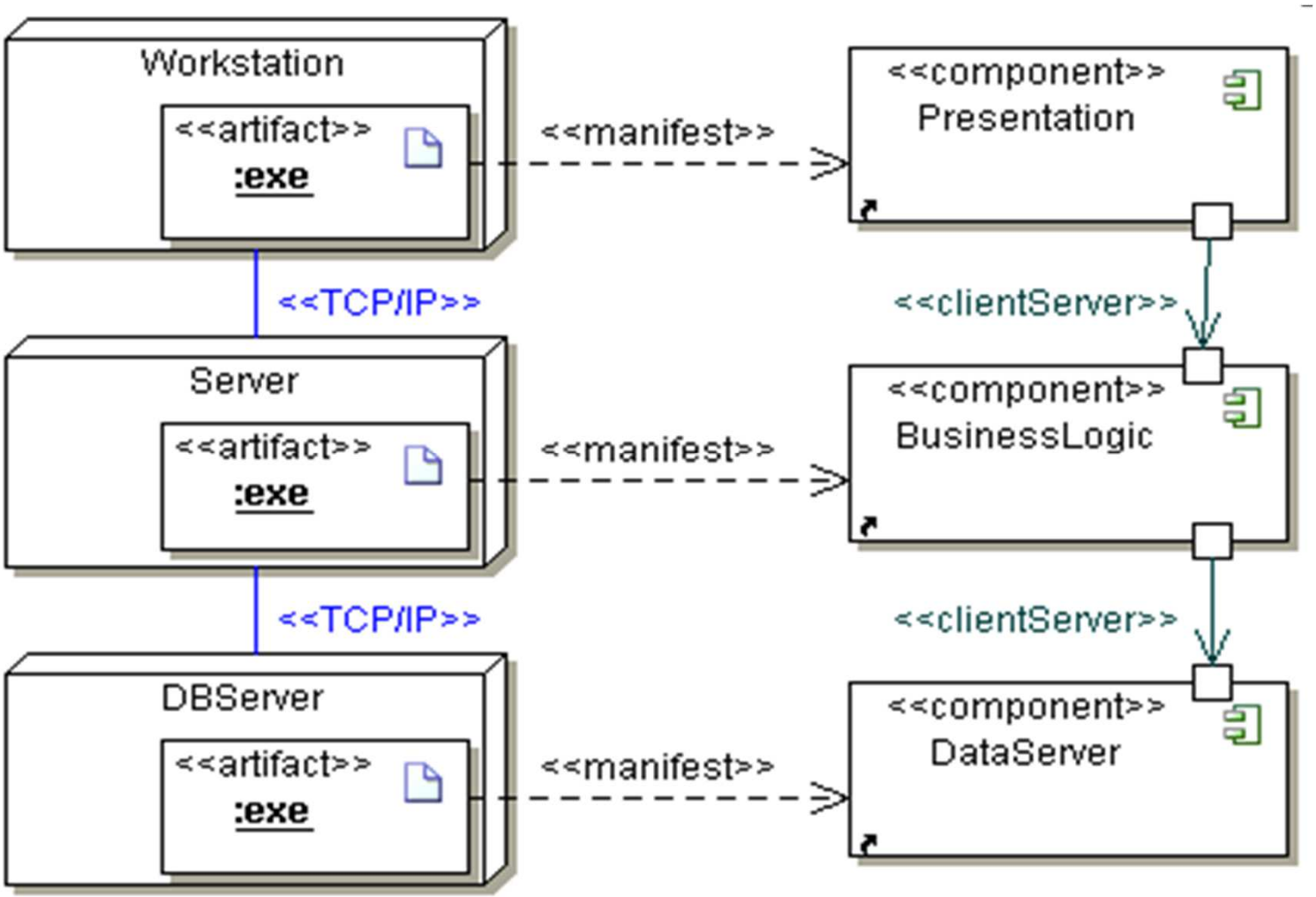
- La directory che realizza il modulo nomemodulo contiene
  - make
  - src
    - NomeModulo.cc
  - include
    - NomeModulo.h
  - doc
    - NomeModulo.doc
  - config
    - NomeModulo.conf

Stili logistici: assegnamento del lavoro

- Elementi
  - software: moduli
  - ambiente: unità organizzativa
    - persona, gruppo, dipartimento, sotto-contraente, …
- Proprietà dell'assegnamento
  - completezza e non sovrapposizione
- Usi
  - comunicazione della struttura del progetto
  - base della WBS (struttura di decomposizione del lavoro) e delle stime dettagliate di costi e tempi

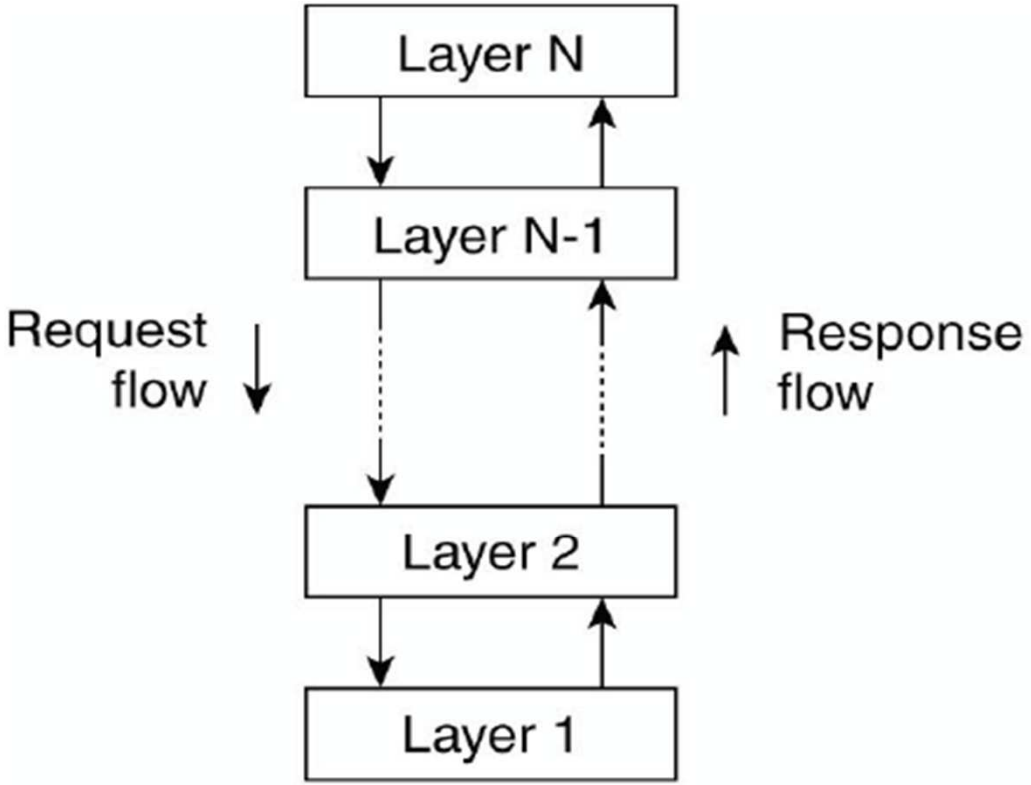
Viste ibride (C&C e dislocazione)

- 3-tier



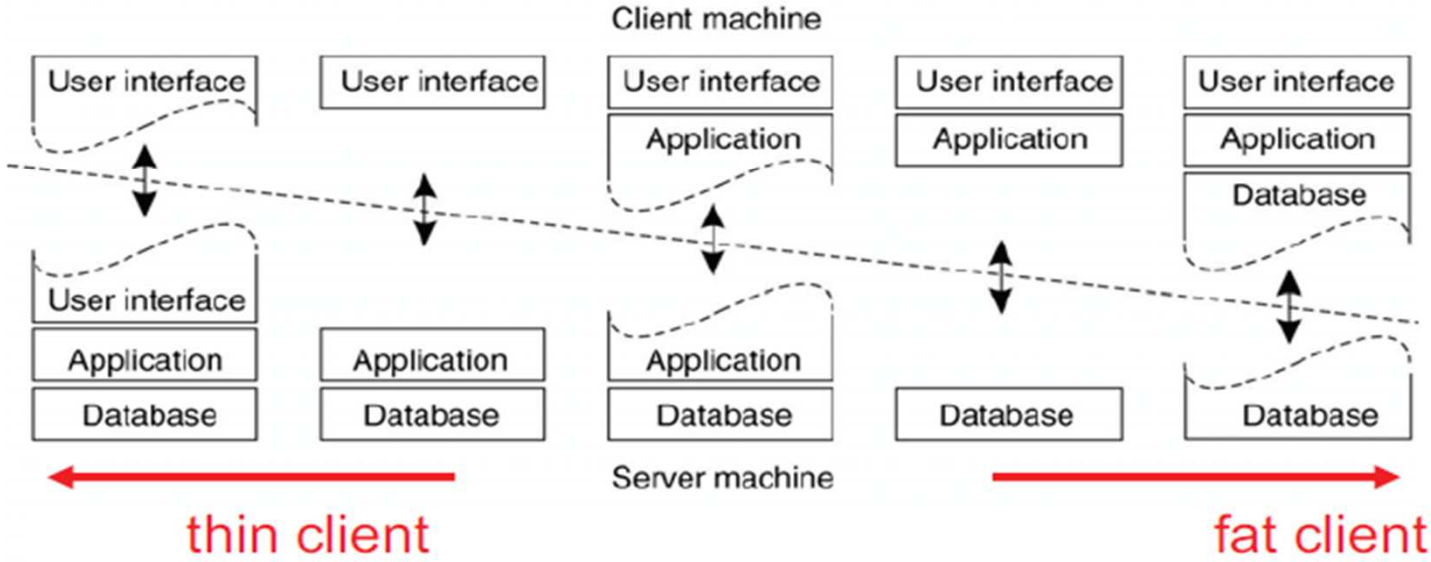
Architettura a livelli

- Componenti organizzati in livelli (*layer*)
- Un componente a livello *i* può invocare un componente del livello sottostante *i-1*
- Le richieste scendono lungo la gerarchia, mentre le risposte risalgono
- Largamente adottato dalla comunità della rete



Architetture multilivello

- Mapping tra livelli logici (layer) e livelli fisici (tier)
- Architettura ad un livello (single-tier): configurazione monolitica mainframe e terminale “stupido” (non è C/S!)
- Architettura a due livelli (two-tier): due livelli fisici (macchina client/singolo server)
- Architettura a tre livelli (three-tier): ciascun livello su una macchina separata
- Diverse configurazioni two-tier

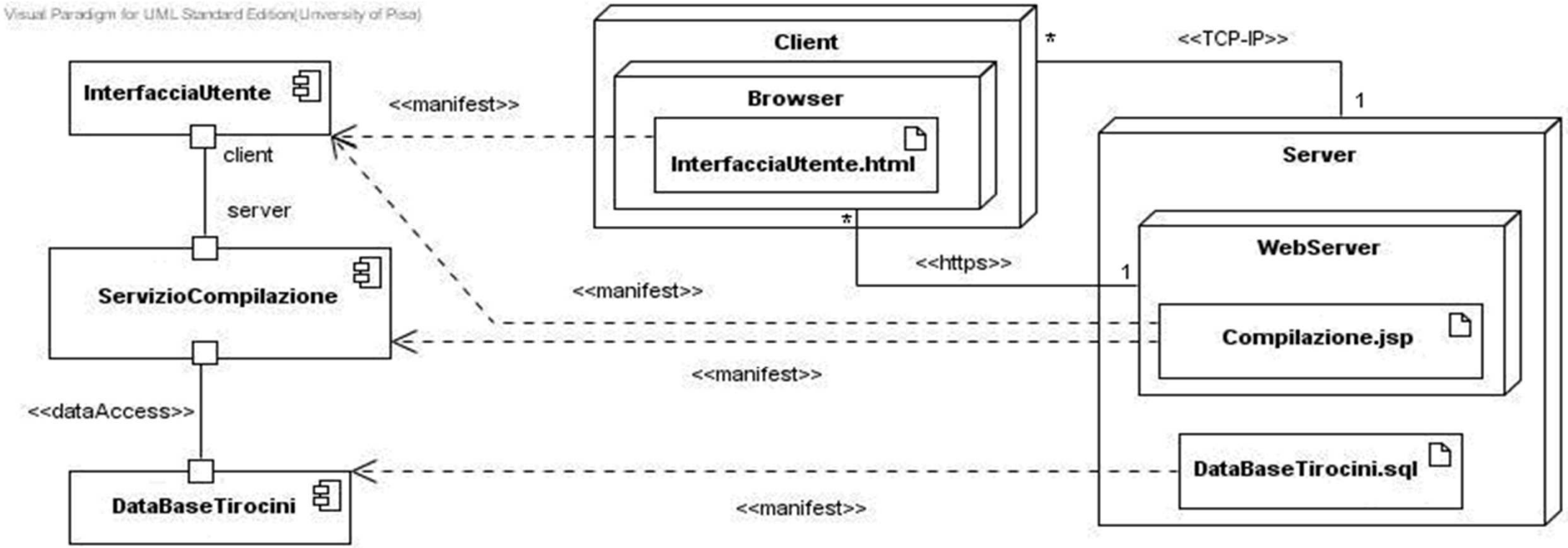


Architetture multilivello (cont'd)

- Da un livello ad N livelli
- Con l'introduzione di ciascun livello
  - L'architettura guadagna in flessibilità, funzionalità e possibilità di distribuzione
- Ma l'architettura multilivello potrebbe introdurre un problema di prestazioni
  - Aumenta il costo della comunicazione
  - Viene introdotta più complessità, in termini di gestione ed ottimizzazione

Esempio
Tirocini: 3-layers, 2-tier

Vista ibrida (C&C e dislocazione) dell'architettura del sotto-sistema di Compilazione, assumendo che gli artefatti che manifestano le componenti citate siano: Compilazione.html, visualizzato da un browser di una macchina client e Compilazione.jsp (dislocata su un web server) e DataBaseTirocini.sql, mantenute su una macchina server.



Syllabus

- Dispensa di architettura, alias Carlo Montangero e Laura Semini, *Architetture software e Progettazione di dettaglio, Note per il Corso di Ingegneria del software, 2014*
- Arlow:
  - cap 11 package di analisi
  - cap 17 interfacce e componenti
  - cap 22 Deployment
- Fuggetta: cap 7, pp129-142

Riferimenti

- Shaw, M e Garlan, D. Software Architecture. Prentice-Hall, 1996.
- Soni, D. et al. Software Architecture in Industrial Application. Proc. 17° Int. Conf. on Software Engineering, 1995.
- Parnas, D. Designing Software for the ease of Extention and Contraction. IEEE Transaction on SE 5(2), 1979.
- Krutchen, P. The 4+1 View Model of Architecture. IEEE Software 12(6), 1995.
- Magee, J. et al. Specifying Distributed Software Architecures. Proc. Fifth Europ. Software Eng. Conf. LNCS 989, Springer Verlag, 1995.
- Clemens, P. et al. Documenting Software Architectures. Addison Wesley, 2003.