

System Design

Ingegneria del Software

System Design: una panoramica

~ Dalla fase di analisi dei requisiti:

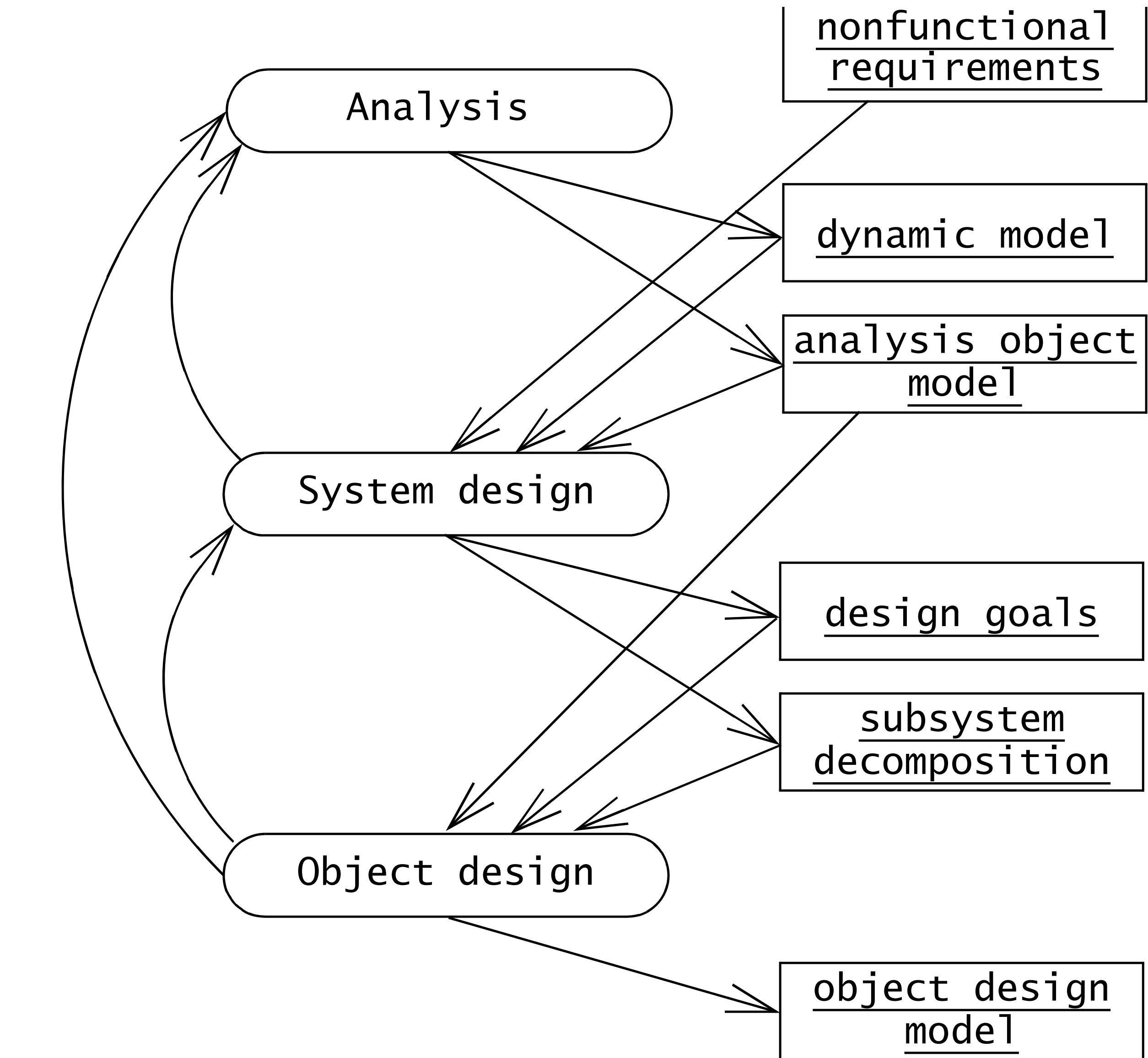
- un set di requisiti non funzionali: esempio il tempo massimo di risposta, il linguaggio di programmazione, etc.
- modello dei casi d'uso: che descrive le funzionalità dal punto di vista degli attori
- modello degli oggetti: che descrive le entità manipolate dal sistema
- un sequence diagram per caso d'uso per mostrare la sequenza di interazioni tra gli oggetti che partecipano al caso d'uso

System Design: una panoramica

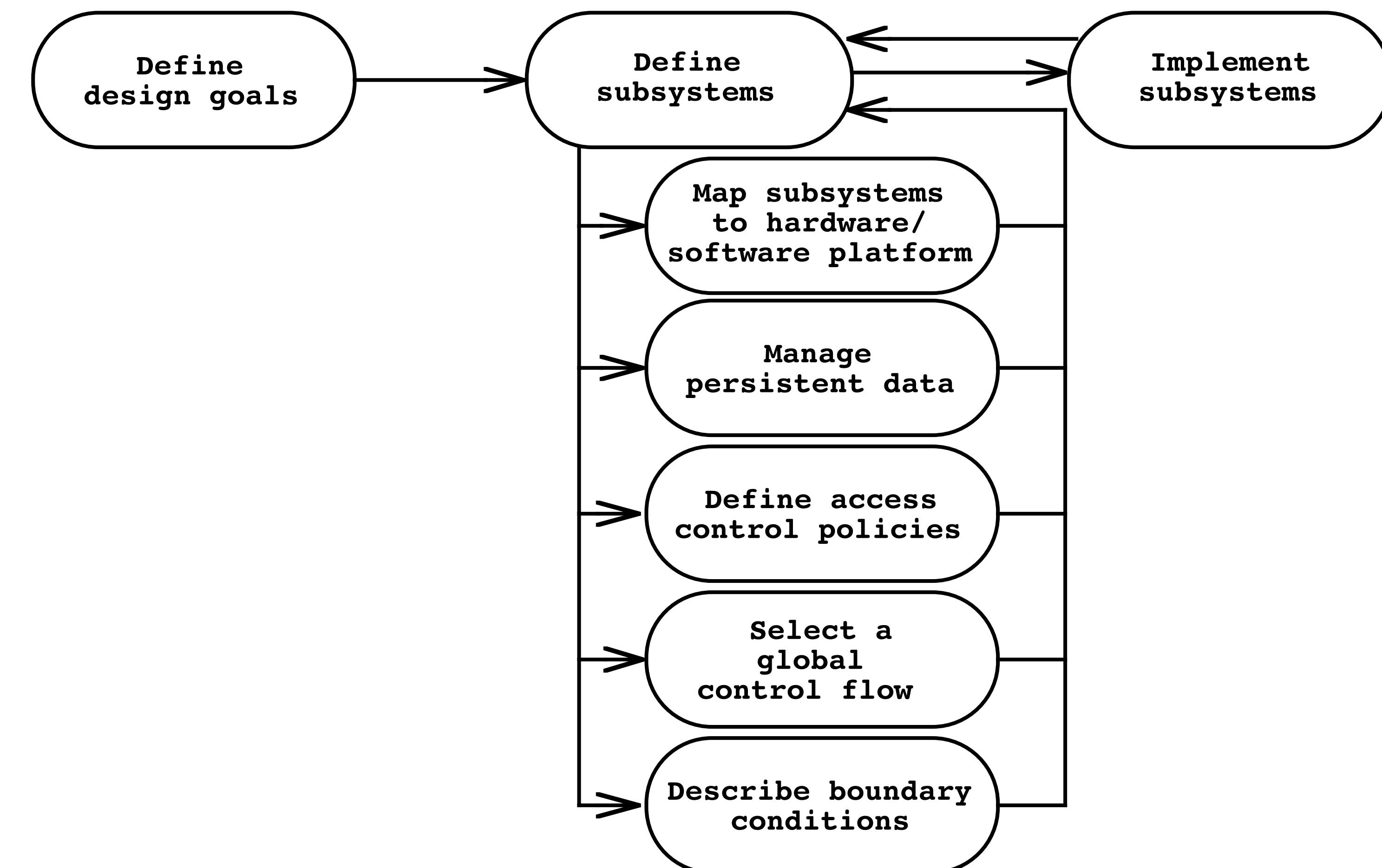
- ~ un modello di analisi descrive un sistema, completamente, dal punto di vista degli attori
- ▶ Serve come base di comunicazione tra clienti e sviluppatori
- ▶ non può contenere informazioni sulla struttura interna e la sua relazione hardware
- ▶ Più generalmente non può contenere dettagli su come il sistema dovrebbe essere realizzato
- ~ System Design

System Design: una panoramica

- ~ Il System Design ha come risultato i seguenti prodotti:
 - ~ una lista di goal di design: descrivono le qualità del sistema che lo sviluppatore deve ottimizzare
 - Strettamente collegati ai requisiti non funzionali
 - architettura software: descrive la decomposizione in sottosistemi in termini di
 - Dipendenza tra sottosistemi
 - Mapping tra sottosistemi e hardware
 - Politiche di decisione come per esempio flusso di controllo e controllo degli accessi



Le attività del System Design



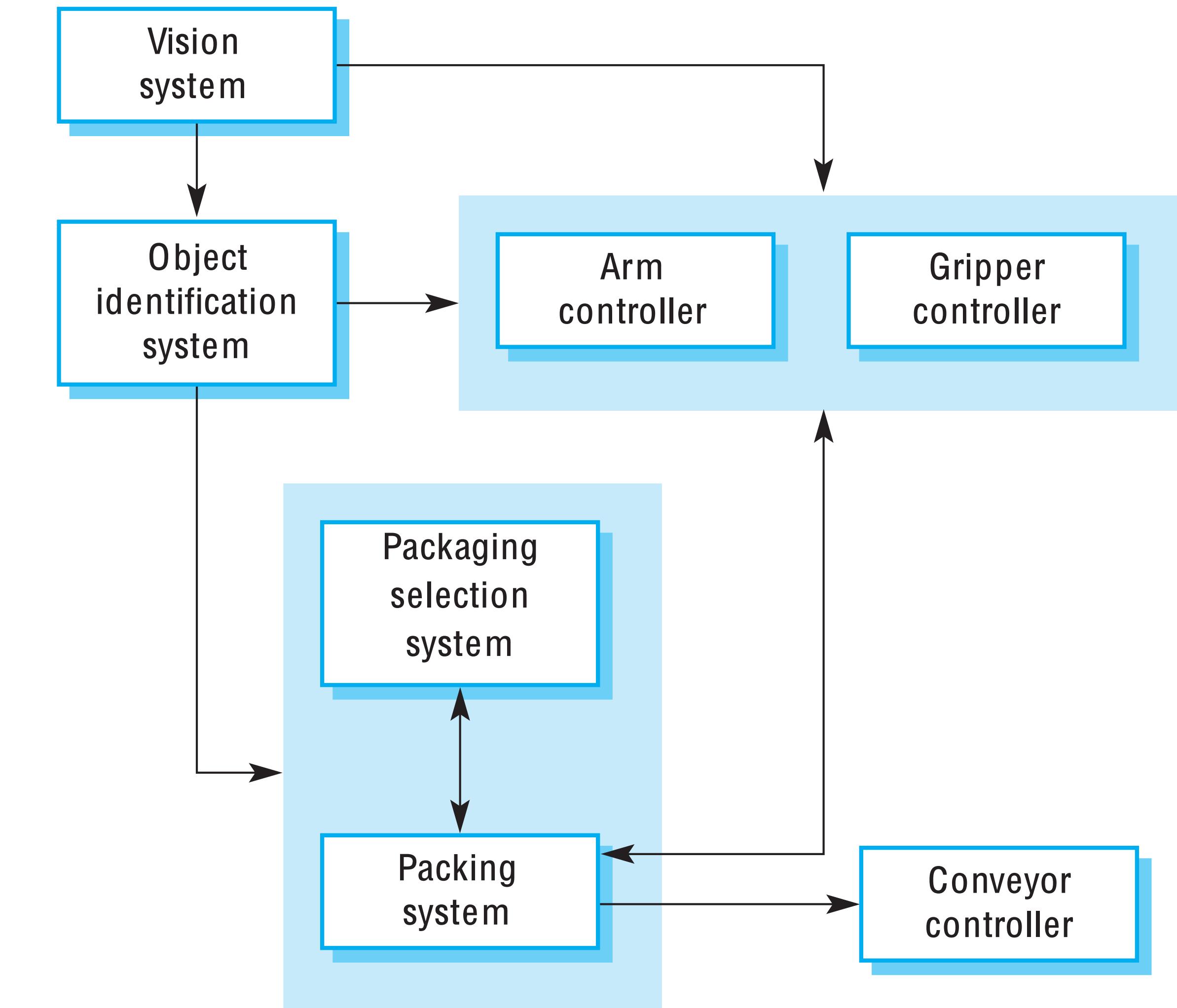
Architetture di sistema

- ~ Semplicemente, una architettura di sistema fornisce indicazioni su come i sottosistemi sono interconnessi tra di loro e come comunicano.

Architetture di sistema

- ~ La progettazione architetturale riguarda la comprensione di:
 - come un sistema software debba essere organizzato e
 - la progettazione della struttura complessiva di tale sistema.
- ~ La progettazione architetturale è il collegamento critico tra la progettazione e l'ingegneria dei requisiti, in quanto identifica i principali componenti strutturali di un sistema e le loro relazioni.
- ~ L'output del processo di progettazione architetturale è un modello architettonico che descrive come il sistema è organizzato come un insieme di componenti comunicanti.

Architettura di un sistema di imballaggio robotizzato



Astrazioni architetturali

- ~ E' possibile progettare l'architettura del software a due livelli di astrazione:
 - Architettura in piccolo - riguarda i singoli programmi. A questo livello, ci occupiamo del modo in cui un singolo programma viene scomposto in componenti.
 - Architettura in grande - riguarda l'architettura di sistemi aziendali complessi che includono altri sistemi, programmi e componenti di programma. Questi sistemi aziendali sono distribuiti su diversi computer, che possono essere di proprietà e gestiti da diverse aziende.

Vantaggi della progettazione architetturale

~ Comunicazione agli stakeholder

- L'architettura può essere utilizzata come punto di discussione da parte degli stakeholder del sistema.

~ Analisi del sistema

- Significa che è possibile analizzare se il sistema può soddisfare i suoi requisiti non funzionali.

~ Riutilizzo su larga scala

- L'architettura può essere riutilizzabile in una serie di sistemi

- Possono essere sviluppate architetture di linee di prodotti.

Modellazione delle architetture di sistema

- ~ I diagrammi a blocchi semplici e informali che mostrano entità e relazioni sono il metodo più utilizzato per documentare le architetture software.
- ~ Ma questi sono stati criticati perché mancano di semantica, non mostrano i tipi di relazioni tra entità né le proprietà visibili delle entità nell'architettura.
- ~ Dipende dall'uso dei modelli architetturali: i requisiti per la semantica dei modelli dipendono da come vengono utilizzati

Modellazione delle architetture di sistema

~ Diagrammi a blocchi:

- Molto astratti - non mostrano la natura delle relazioni dei componenti né le proprietà esternamente visibili dei sottosistemi.
- Tuttavia, utili per la comunicazione con le parti interessate e per la pianificazione del progetto.
- Offrono una vista di alto livello della struttura del sistema
- Facilmente comprensibili da varie persone di discipline diverse interessate al processo di sviluppo.

Uso di modelli architetturali – due modi

- ~ Per facilitare la discussione sulla progettazione del sistema
 - ◆ una visione architettonica di alto livello di un sistema è utile per la comunicazione con gli stakeholder del sistema e la pianificazione del progetto perché non è ingombra di dettagli. Le parti interessate possono relazionarsi con esso e comprendere una visione astratta del sistema. Essi possono quindi discutere il sistema nel suo insieme senza essere confusi dai dettagli.
- ~ Come modo di documentare un' architettura progettata
 - ◆ L' obiettivo è produrre un modello di sistema completo che mostri i diversi componenti di un sistema, le loro interfacce e i loro collegamenti.

Decisioni di progettazione architetturale

- ~ La progettazione architetturale è un **processo creativo**, quindi il processo varia a seconda del tipo di sistema sviluppato.
 - ▶ soddisfa requisiti funzionali e non funzionali d'un sistema
- ~ Tuttavia, una serie di decisioni comuni abbracciano tutti i processi di progettazione e riguardano le caratteristiche non funzionali del sistema.
 - ▶ basandosi sulle proprie conoscenze ed esperienze si deve rispondere alle domande della slide seguente

Is there a generic application architecture that can act as a template for the system that is being designed?

How will the system be distributed across hardware cores or processors?

What architectural patterns or styles might be used?

What will be the fundamental approach used to structure the system?

What strategy will be used to control the operation of the components in the system?

How will the structural components in the system be decomposed into sub-components?

What architectural organization is best for delivering the non-functional requirements of the system?

How should the architecture of the system be documented?

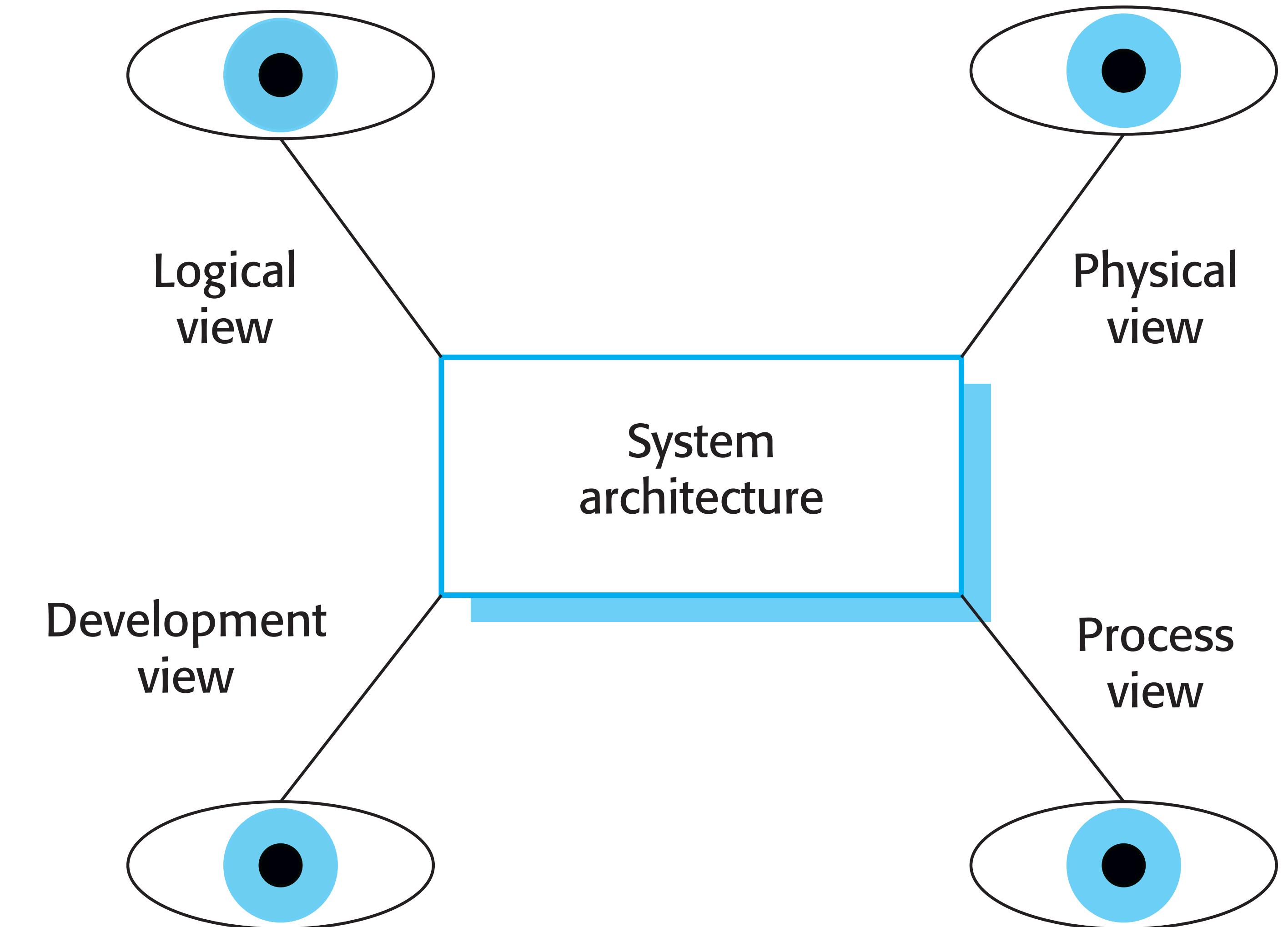
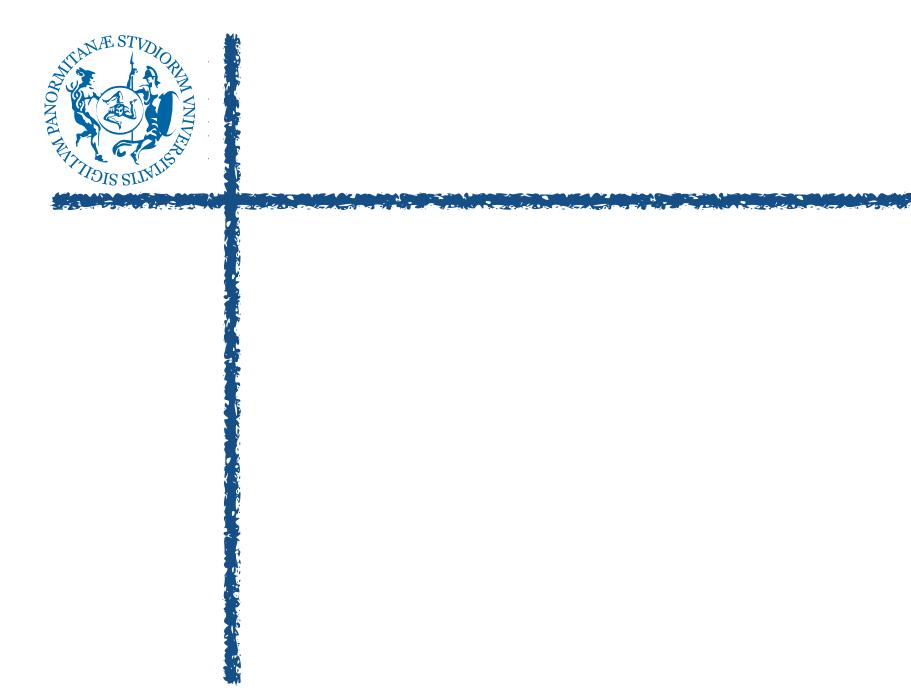


Riuso

- ~ I sistemi nello stesso settore hanno spesso architetture simili che riflettono concetti di dominio.
- ~ Per esempio, le linee di prodotti applicativi sono costruite attorno ad un'architettura core con varianti che soddisfano particolari esigenze del cliente.
- ~ Quando si progetta un'architettura di un sistema si deve decidere cosa hanno in comune il sistema e le classi di applicazioni più generali e stabilire quanto può essere riusato da queste architetture applicative
- ~ L'architettura di un sistema può essere progettata intorno a uno o più modelli o stili architettonici.
 - ◆ Questi catturano l'essenza di un'architettura e possono essere istanziati in modi diversi.

Viste architetturali

- ~ Uno schema architetturale è una descrizione dell'organizzazione del sistema
- ~ Quali punti di vista o prospettive sono utili per progettare e documentare l'architettura di un sistema?
- ~ Quali notazioni dovrebbero essere usate per descrivere i modelli architettonici?
- ~ Ogni modello architettonico mostra una sola vista o prospettiva del sistema.
- ~ Potrebbe mostrare come un sistema viene scomposto in moduli, come interagiscono i processi di run-time o i diversi modi in cui i componenti del sistema sono distribuiti su una rete.
- ~ Sia per il design che per la documentazione, di solito è necessario presentare viste multiple dell'architettura software.



4 + 1 modelli di visualizzazione dell' architettura software

- ~ una vista logica, che mostra le principali astrazioni nel sistema come oggetti o classi di oggetti.
- ~ una vista del processo, che mostra come, al tempo di esecuzione, il sistema sia composto da processi interagenti.
- ~ una vista di sviluppo, che mostra come il software viene decomposto per lo sviluppo.
- ~ una vista fisica, che mostra l' hardware del sistema e come i componenti software sono distribuiti tra i processori del sistema.
- ~ casi o scenari di utilizzo correlati (+1)

Rappresentare le viste architetturali

- ~ Alcuni sostengono che l' Unified Modeling Language (UML) sia una notazione appropriata per descrivere e documentare le architetture di sistema.
- ~ C'è chi non è d'accordo perché UML potrebbe non includere appropriate viste astratte per una descrizione di alto livello del sistema.
- ~ Linguaggi di descrizione architettonica (ADL) sono stati sviluppati, ma non sono ampiamente utilizzati

Pattern architetturali

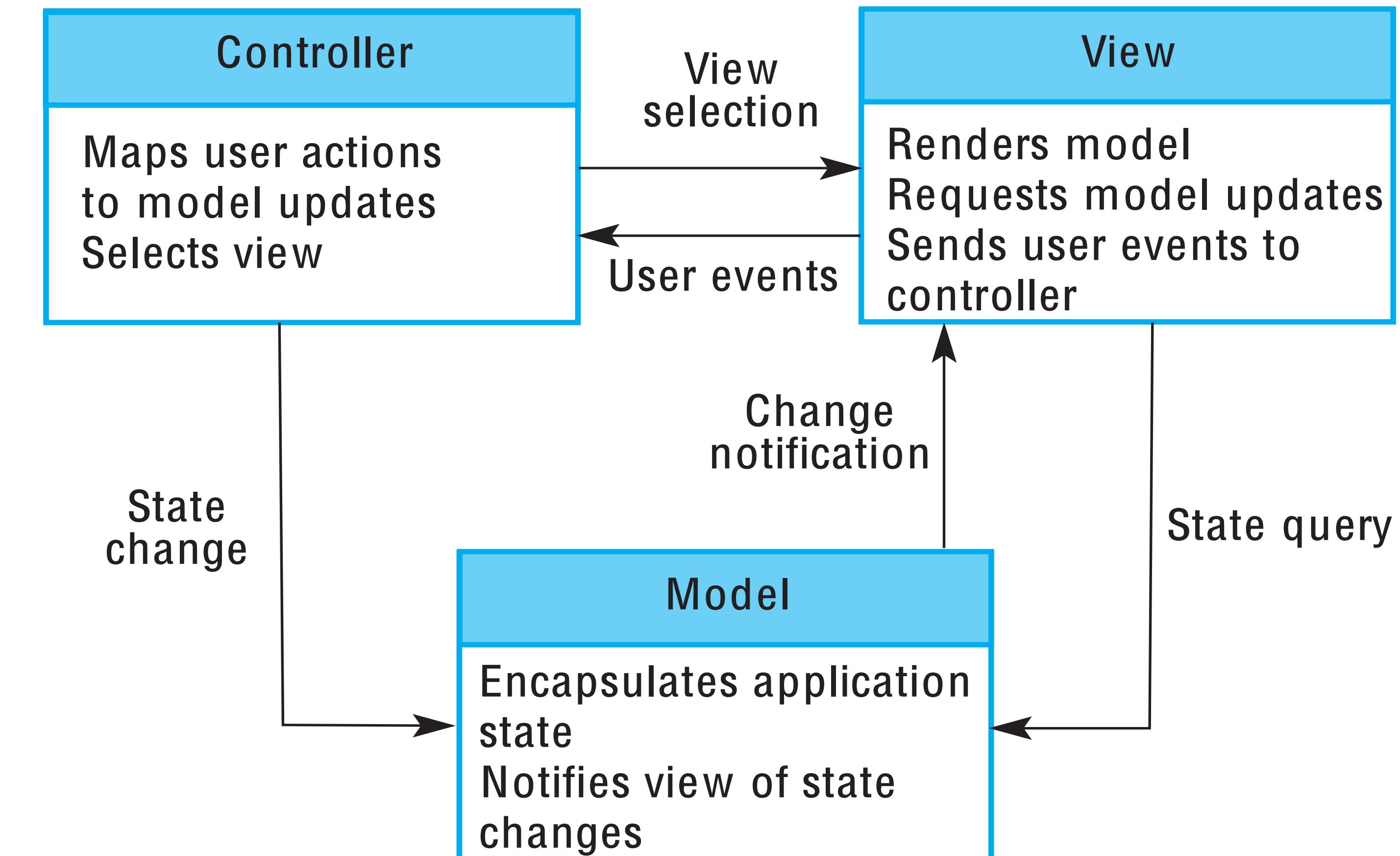
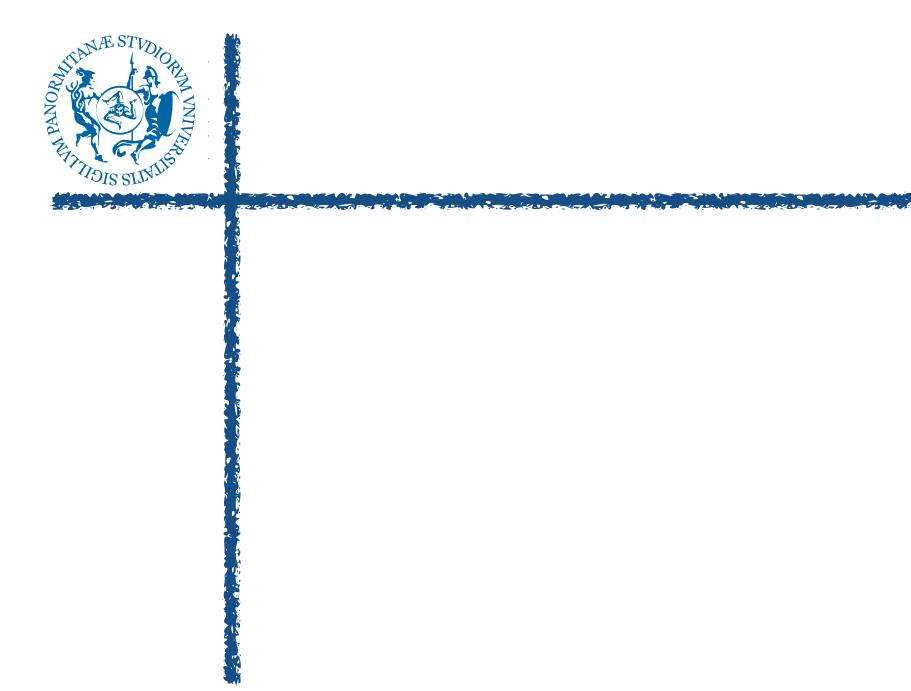
- ~ I modelli sono un mezzo per rappresentare, condividere e riutilizzare la conoscenza.
- ~ Un modello architettonico è una descrizione stilizzata delle buone pratiche di progettazione, che è stata sperimentata e testata in diversi ambienti.
- ~ I modelli dovrebbero includere informazioni su quando sono e quando non sono utili.
- ~ Gli schemi possono essere rappresentati utilizzando descrizioni tabulari e grafiche.

Architettura Model View Controller

Name	MVC (Model-View-Controller)
Description	Separates presentation and interaction from the system data. The system is structured into three logical components that interact with each other. The Model component manages the system data and associated operations on that data. The View component defines and manages how the data is presented to the user. The Controller component manages user interaction (e.g., key presses, mouse clicks, etc.) and passes these interactions to the View and the Model. See Figure 6.3.
Example	Figure 6.4 shows the architecture of a web-based application system organized using the MVC pattern.
When used	Used when there are multiple ways to view and interact with data. Also used when the future requirements for interaction and presentation of data are unknown.
Advantages	Allows the data to change independently of its representation and vice versa. Supports presentation of the same data in different ways with changes made in one representation shown in all of them.
Disadvantages	Can involve additional code and code complexity when the data model and interactions are simple.

Architettura Model/View/Controller

- ~ I sottosistemi sono classificati in tre diversi tipi:
 - ◆ Model subsystem - responsabile di mantenere la conoscenza del dominio
 - ◆ View subsystem - responsabile di esibire gli elementi del dominio all'utente
 - ◆ Controller subsystem - responsabile di gestire la sequenza delle interazioni con l'utente



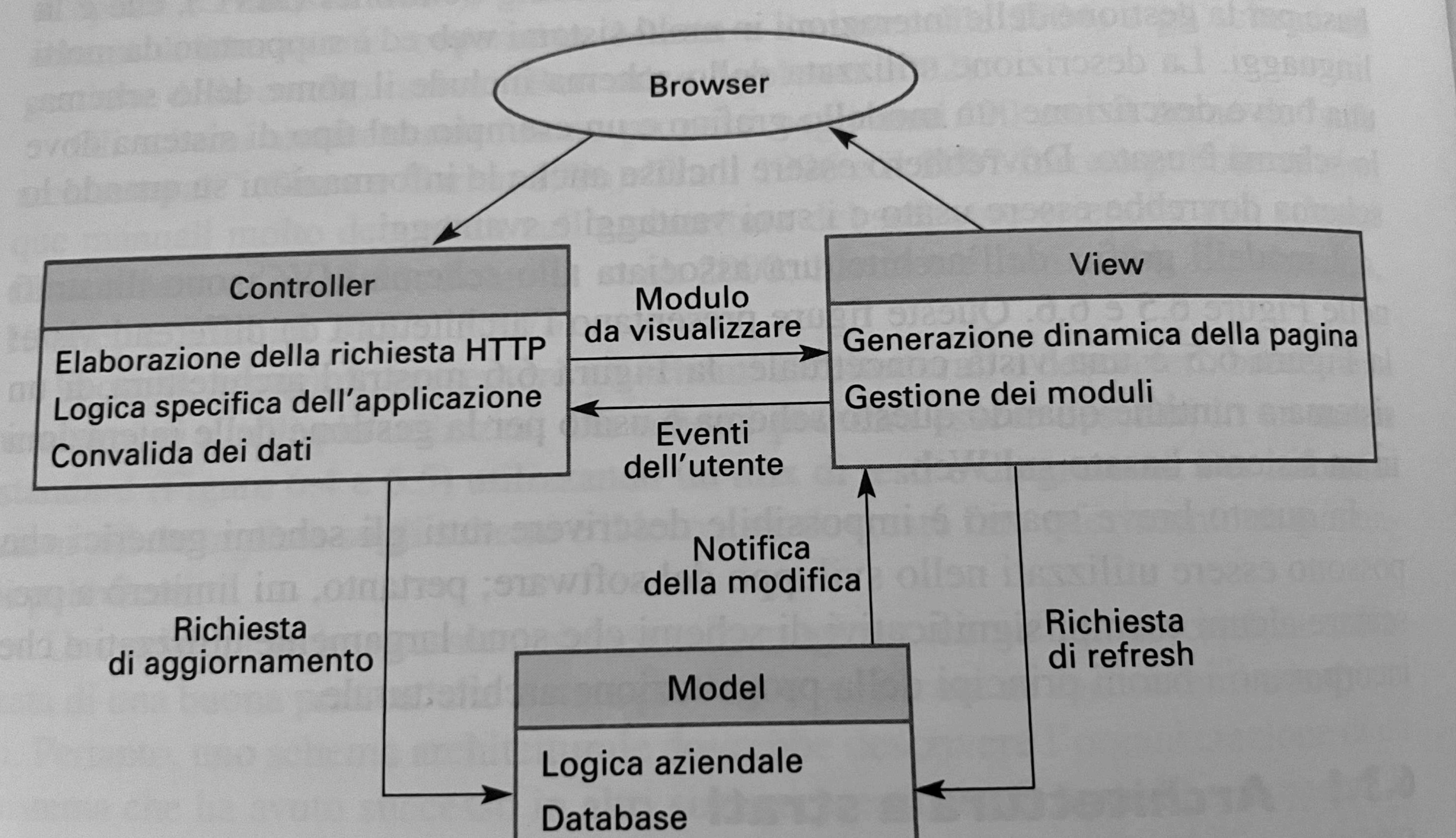
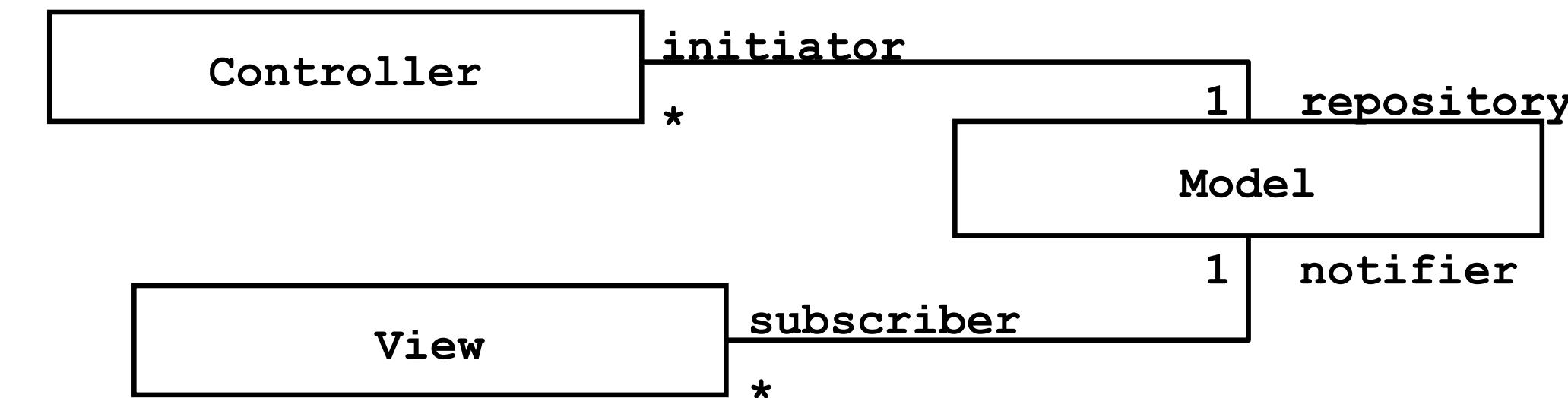


Figura 6.6 Architettura di applicazioni web che usa lo schema MVC.

Architettura Model/View/Controller



- ~ MVC è un caso speciale della architettura repository
- ▶ i sottosistemi model implementano la struttura dati centrale e i sottosistemi controller esplicitamente dettano il flusso di controllo

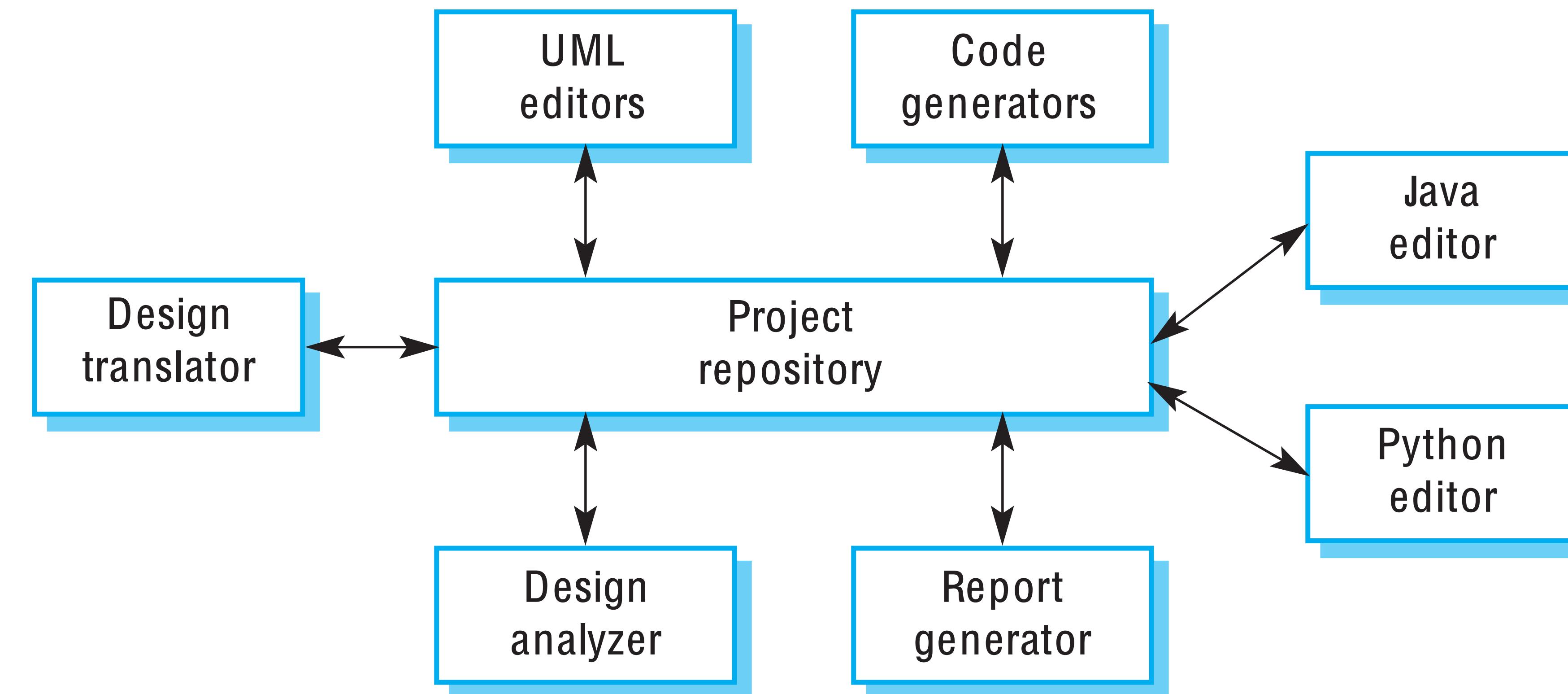
Architettura Repository

- ~ I sottosistemi devono scambiare dati. Ciò può avvenire in due modi:
 - ◆ I dati condivisi sono conservati in una banca dati o in un archivio centrale e possono essere consultati da tutti i sottosistemi;
 - ◆ Ogni sottosistema mantiene il proprio database e trasmette i dati esplicitamente ad altri sottosistemi.
- ~ Quando si devono condividere grandi quantità di dati, il modello di condivisione del repository è più comunemente utilizzato
 - ◆ è un meccanismo di condivisione dei dati efficiente.

Architettura Repository

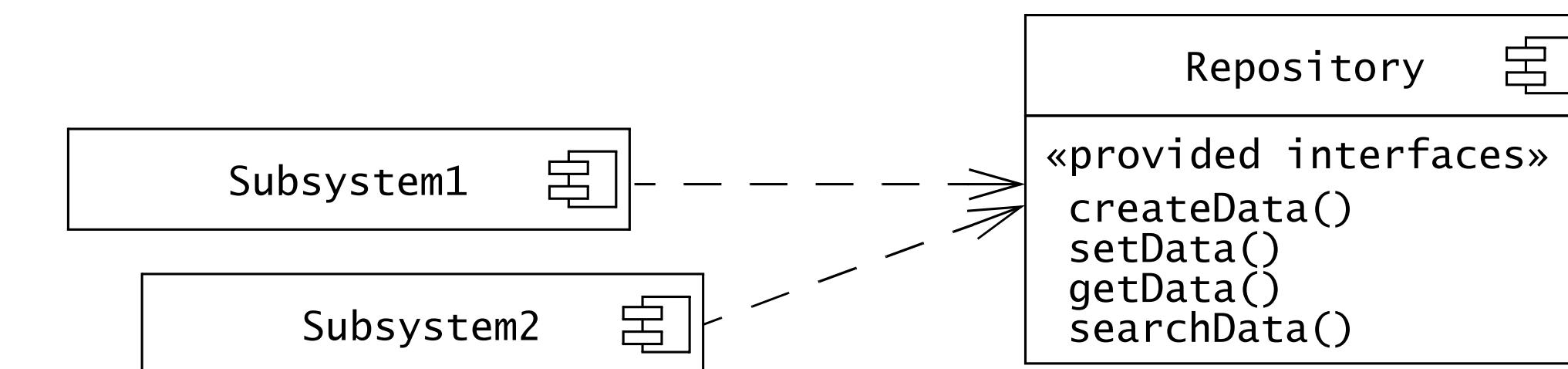
Name	Repository
Description	All data in a system is managed in a central repository that is accessible to all system components. Components do not interact directly, only through the repository.
Example	Figure 6.9 is an example of an IDE where the components use a repository of system design information. Each software tool generates information which is then available for use by other tools.
When used	You should use this pattern when you have a system in which large volumes of information are generated that has to be stored for a long time. You may also use it in data-driven systems where the inclusion of data in the repository triggers an action or tool.
Advantages	Components can be independent—they do not need to know of the existence of other components. Changes made by one component can be propagated to all components. All data can be managed consistently (e.g., backups done at the same time) as it is all in one place.
Disadvantages	The repository is a single point of failure so problems in the repository affect the whole system. May be inefficiencies in organizing all communication through the repository. Distributing the repository across several computers may be difficult.

Una Repository per un IDE



Architettura Repository

- ~ I sottosistemi accedono o modificano dati da una singola struttura dati
- ~ I sottosistemi sono scarsamente accoppiati - interagiscono solo attraverso repository
- ~ Il flusso di controllo è dettato da repository centrale o dai sottosistemi



Architettura Repository

- ~ Una architettura repository è tipica per i database management system
- ~ Il principale svantaggio: il repository centrale può spesso diventare un collo di bottiglia

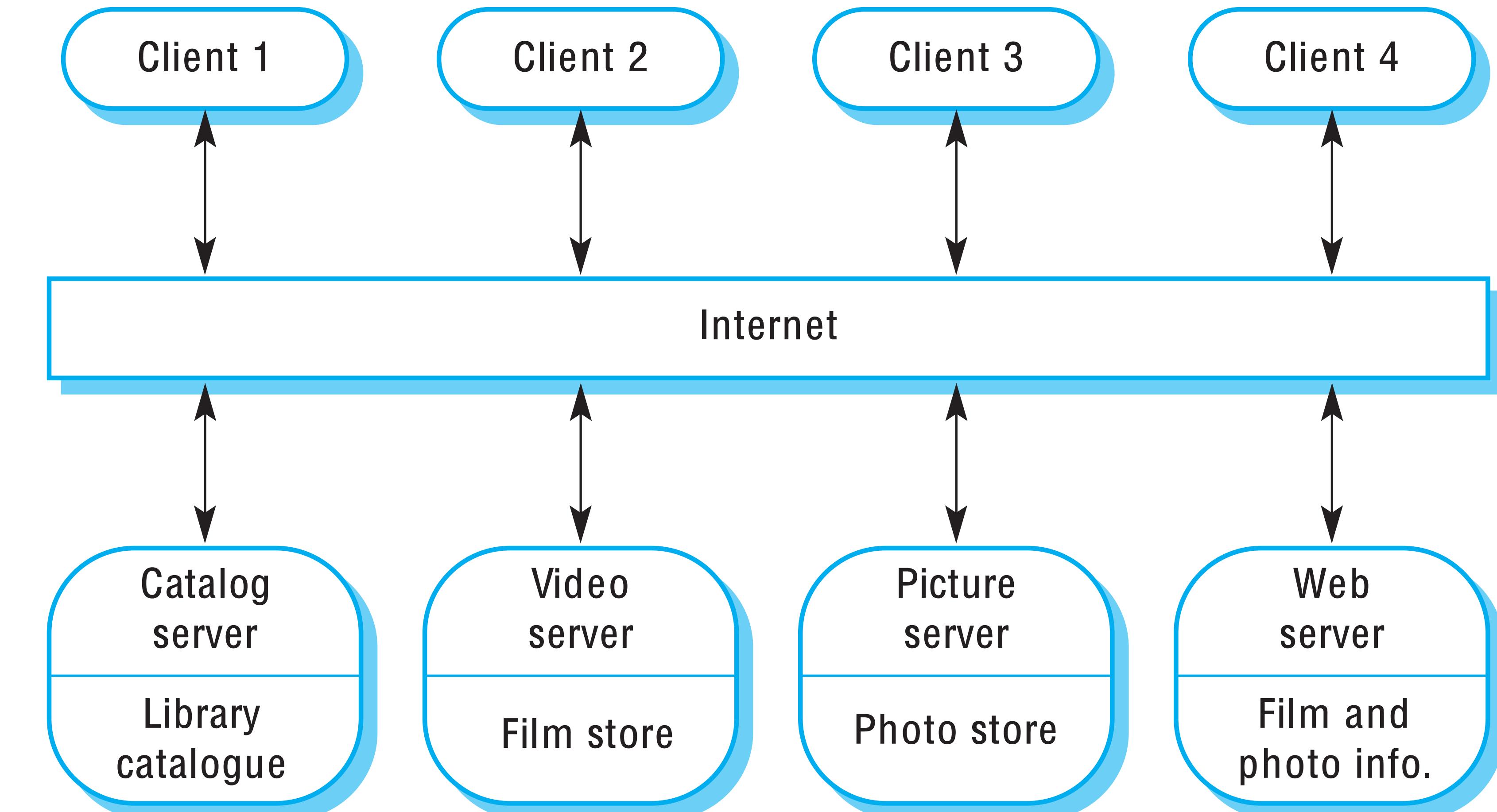
Architettura Client - Server

- ~ Modello di sistema distribuito che mostra come i dati e l'elaborazione sono distribuiti su una serie di componenti e come questi sono organizzati a runtime.
- ~ Può essere implementato su un singolo computer.
- ~ I principali modelli di questo sistema sono:
 - ▶ Set di server stand-alone che forniscono servizi ad altri sottosistemi, per esempio servizi specifici come la stampa, la gestione dei file, ecc. I server sono componenti software e molti di questi possono essere eseguiti nello stesso computer.
 - ▶ Set di client che ricorrono a questi servizi, richiedono servizi a questi server. Di solito si tratta di varie istanze di programmi client eseguiti contemporaneamente su computer diversi.
 - ▶ Rete che consente ai client di accedere ai servizi. Di solito i sistemi client-server sono implementati come distribuiti e collegati attraverso protocolli internet.

Architettura Client – Server

Name	Client-server
Description	In a client–server architecture, the functionality of the system is organized into services, with each service delivered from a separate server. Clients are users of these services and access servers to make use of them.
Example	Figure 6.11 is an example of a film and video/DVD library organized as a client–server system.
When used	Used when data in a shared database has to be accessed from a range of locations. Because servers can be replicated, may also be used when the load on a system is variable.
Advantages	The principal advantage of this model is that servers can be distributed across a network. General functionality (e.g., a printing service) can be available to all clients and does not need to be implemented by all services.
Disadvantages	Each service is a single point of failure so susceptible to denial of service attacks or server failure. Performance may be unpredictable because it depends on the network as well as the system. May be management problems if servers are owned by different organizations.

Un' architettura client-server per una cineteca

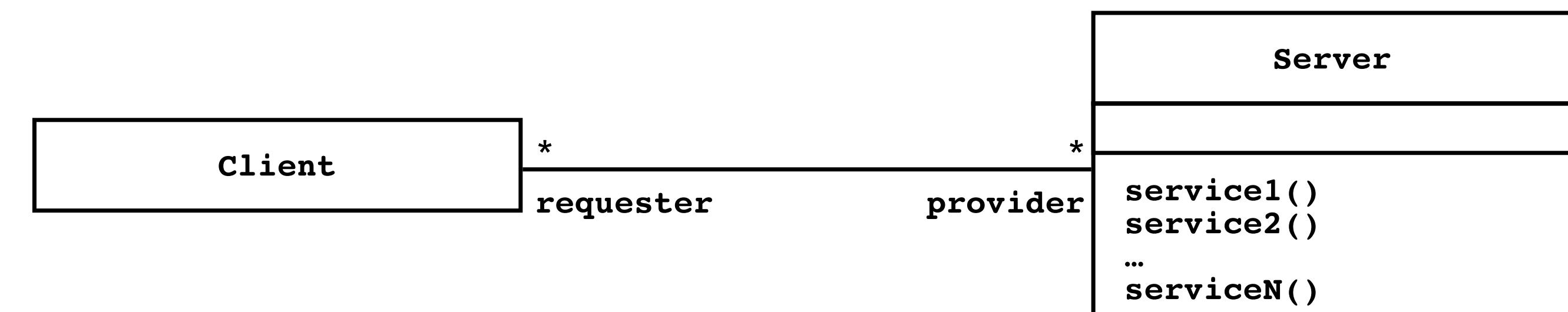


Architettura Client Server

- ~ Uno o più server forniscono servizi ad istanze di sottosistemi chiamati client
- ~ Il client chiama il server che fornisce un servizio e ritorna un risultato
 - ◆ il client conosce l'interfaccia (interface) del server
 - ◆ Il server non ha bisogno di conoscere l'interfaccia del client
- ~ La risposta in generale è immediata
- ~ L'utente interagisce solo con il client

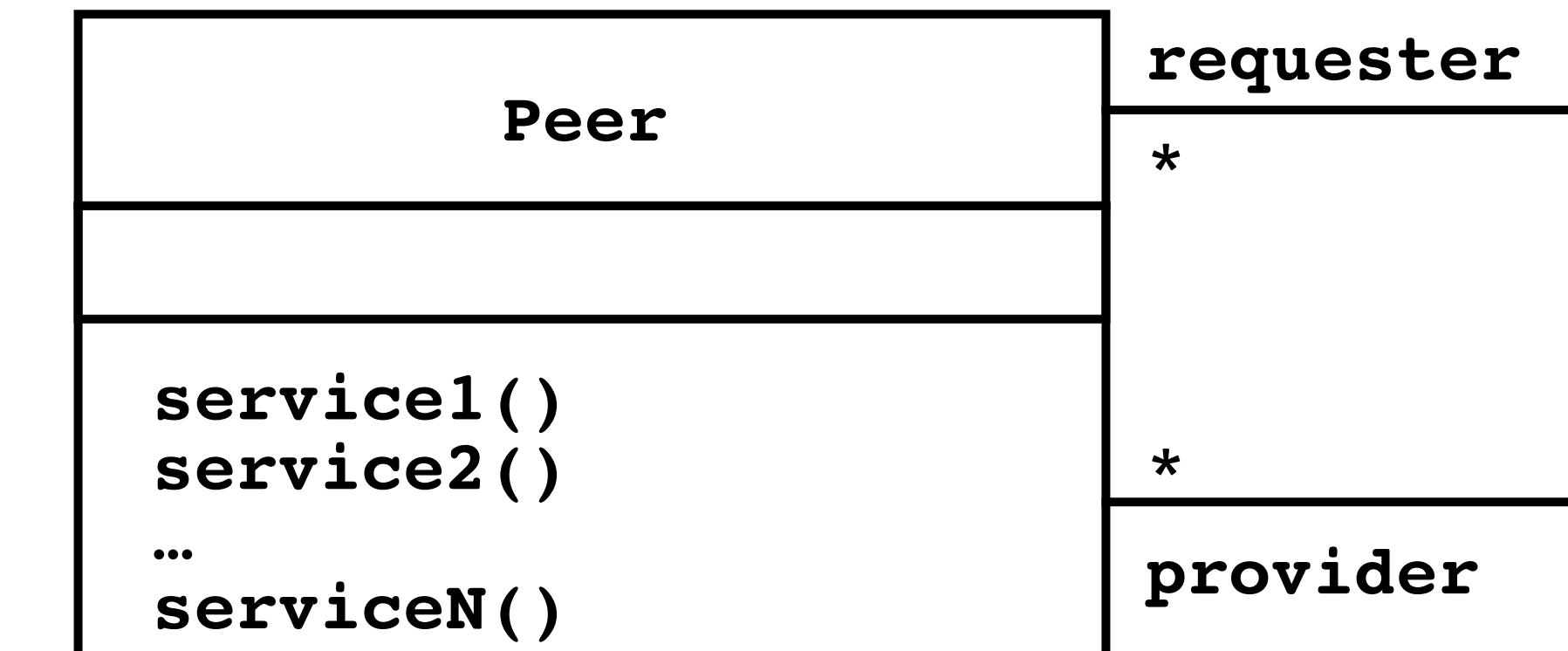
Architettura Client Server

- ~ Un sistema informatico con un database centrale è un esempio di architettura client/server
- ~ Le architetture sono adatte per sistemi distribuiti che gestiscono un grande numero di dati



Architettura Peer-to-peer

- ~ E' una generalizzazione della architettura client/server
- ~ I client possono essere server e vice-versa
- ~ Il flusso di controllo può provocare deadlock



Architettura a Layer

- ~ E' un altro modo per ottenere **separazione ed indipendenza** - permettono di localizzare le modifiche
 - ▶ primo modo MVC, per esempio posso cambiare una view o aggiungerne un'altra senza cambiare i dati nel model.
- ~ utilizzato per modellare l'interfacciamento di sottosistemi.
- ~ organizza il sistema in un insieme di livelli (o macchine astratte) ciascuno dei quali fornisce un insieme di servizi.
- ~ Le funzionalità del sistema sono organizzate in strati separati.
- ~ Ogni strato si basa sulle funzionalità e sui servizi offerti dallo strato immediatamente sottostante.
- ~ Supporta lo sviluppo incrementale di sottosistemi in diversi livelli.
 - ▶ uno strato più esteso può essere implementato senza modificare le sue interfacce e quindi senza influenzare il livello sottostante.
 - ▶ Quando l'interfaccia di un livello cambia, solo il livello adiacente viene influenzato.

Architettura a Layer

Name	Layered architecture
Description	Organizes the system into layers with related functionality associated with each layer. A layer provides services to the layer above it so the lowest-level layers represent core services that are likely to be used throughout the system. See Figure 6.6.
Example	A layered model of a system for sharing copyright documents held in different libraries, as shown in Figure 6.7.
When used	Used when building new facilities on top of existing systems; when the development is spread across several teams with each team responsibility for a layer of functionality; when there is a requirement for multi-level security.
Advantages	Allows replacement of entire layers so long as the interface is maintained. Redundant facilities (e.g., authentication) can be provided in each layer to increase the dependability of the system.
Disadvantages	In practice, providing a clean separation between layers is often difficult and a high-level layer may have to interact directly with lower-level layers rather than through the layer immediately below it. Performance can be a problem because of multiple levels of interpretation of a service request as it is processed at each layer.

Architettura a Layer

User interface

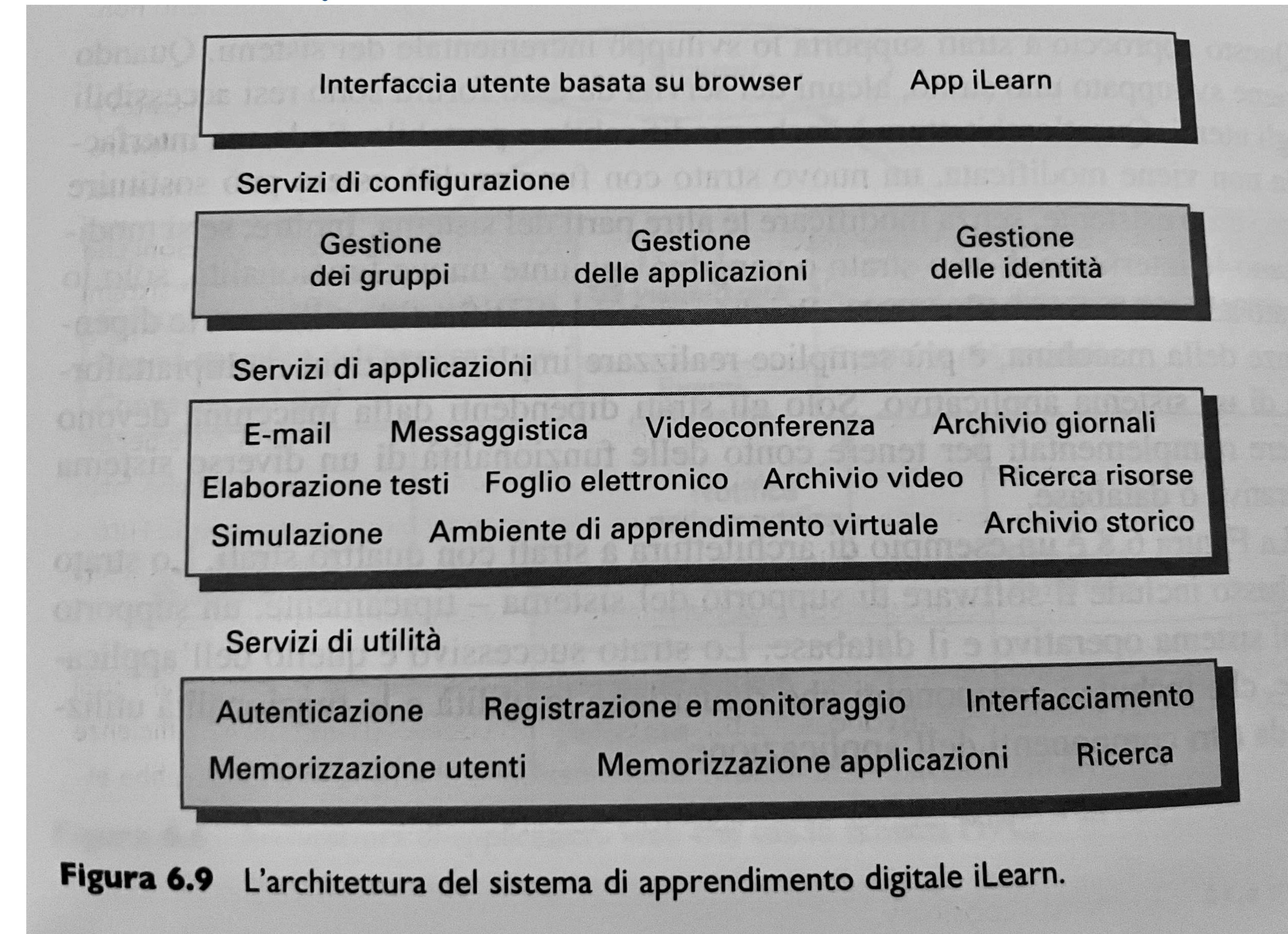
User interface management
Authentication and authorization

Core business logic/application functionality

System utilities

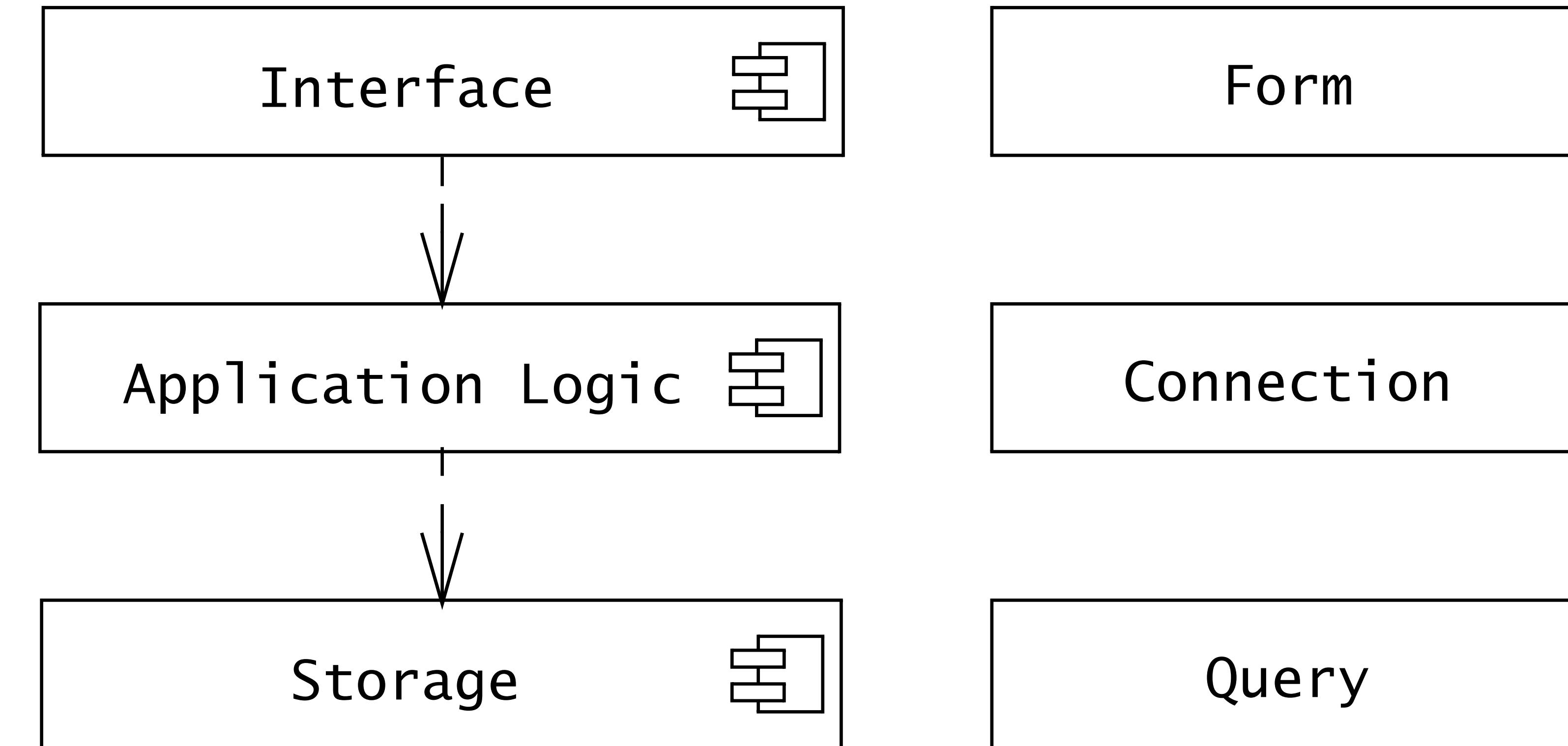
System support (OS, database etc.)

Esempio del iLearn system



Architettura Three-tier

- ~ Questa architettura organizza i sottosistemi in tre layers
 - ▶ interface layer: include tutti gli oggetti boundary che interagiscono con l'utente includendo windows, form e pagine web
 - ▶ Application logic layer: include tutti gli oggetti control ed entity realizzando così elaborazione vera e propria, il controllo sulle regole etc
 - ▶ Storage layer: realizza lo storage, il reperimento e le query sugli oggetti persistenti



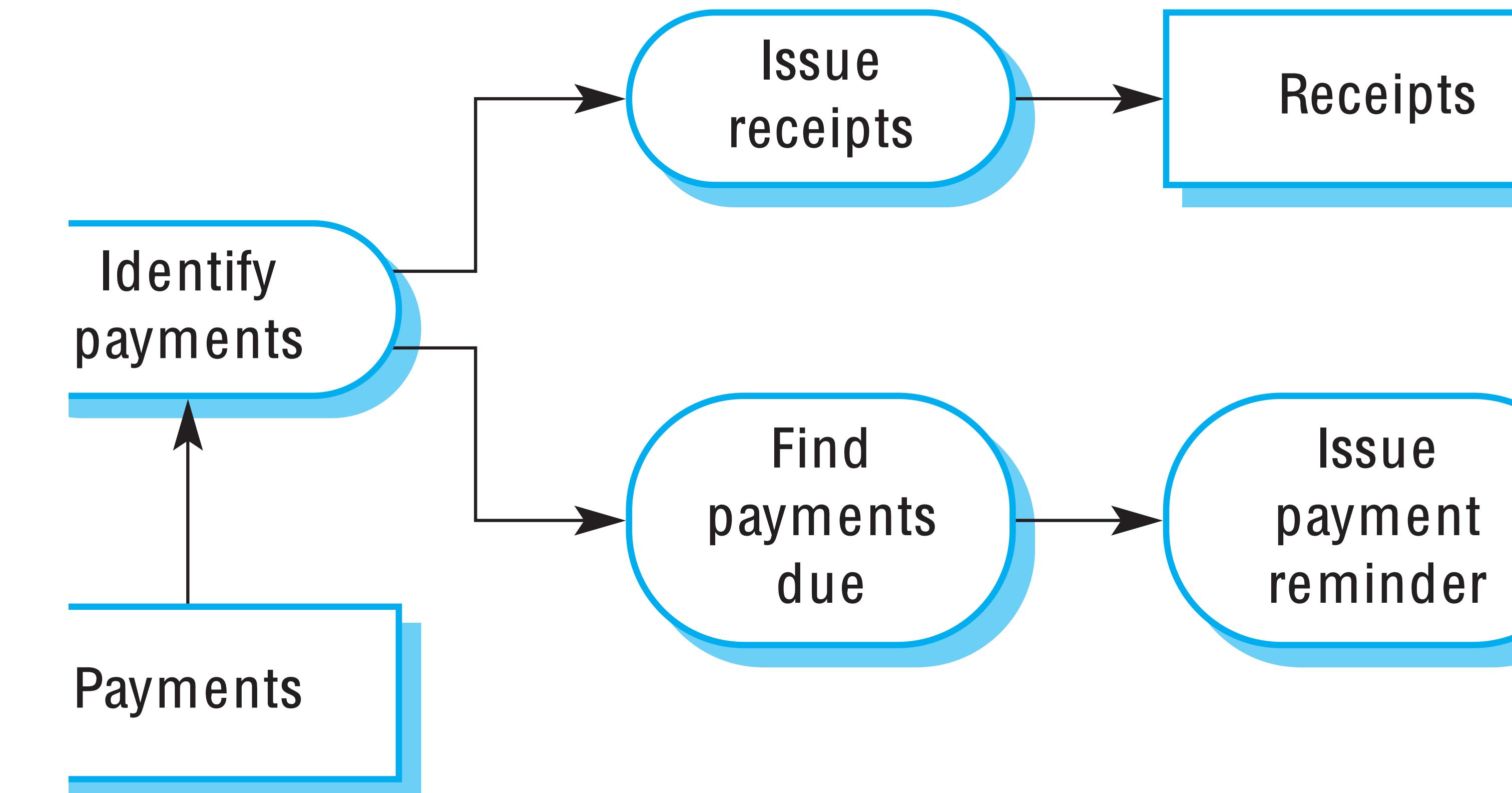
Architettura Pipe and Filter

- ~ Le trasformazioni funzionali elaborano i loro ingressi per produrre uscite.
- ~ Può essere definito come un modello di “tubo e filtro” (come in UNIX shell).
- ~ Le varianti di questo approccio sono molto comuni. Quando le trasformazioni sono sequenziali, si tratta di un modello sequenziale batch che viene ampiamente utilizzato nei sistemi di elaborazione dati.
- ~ Non proprio adatto per sistemi interattivi.

Architettura Pipe and Filter

Name	Pipe and filter
Description	The processing of the data in a system is organized so that each processing component (filter) is discrete and carries out one type of data transformation. The data flows (as in a pipe) from one component to another for processing.
Example	Figure 6.13 is an example of a pipe and filter system used for processing invoices.
When used	Commonly used in data processing applications (both batch- and transaction-based) where inputs are processed in separate stages to generate related outputs.
Advantages	Easy to understand and supports transformation reuse. Workflow style matches the structure of many business processes. Evolution by adding transformations is straightforward. Can be implemented as either a sequential or concurrent system.
Disadvantages	The format for data transfer has to be agreed upon between communicating transformations. Each transformation must parse its input and unparse its output to the agreed form. This increases system overhead and may mean that it is impossible to reuse functional transformations that use incompatible data structures.

Architettura Pipe and Filter - esempio di un sistema di pagamento





System Design ed UML

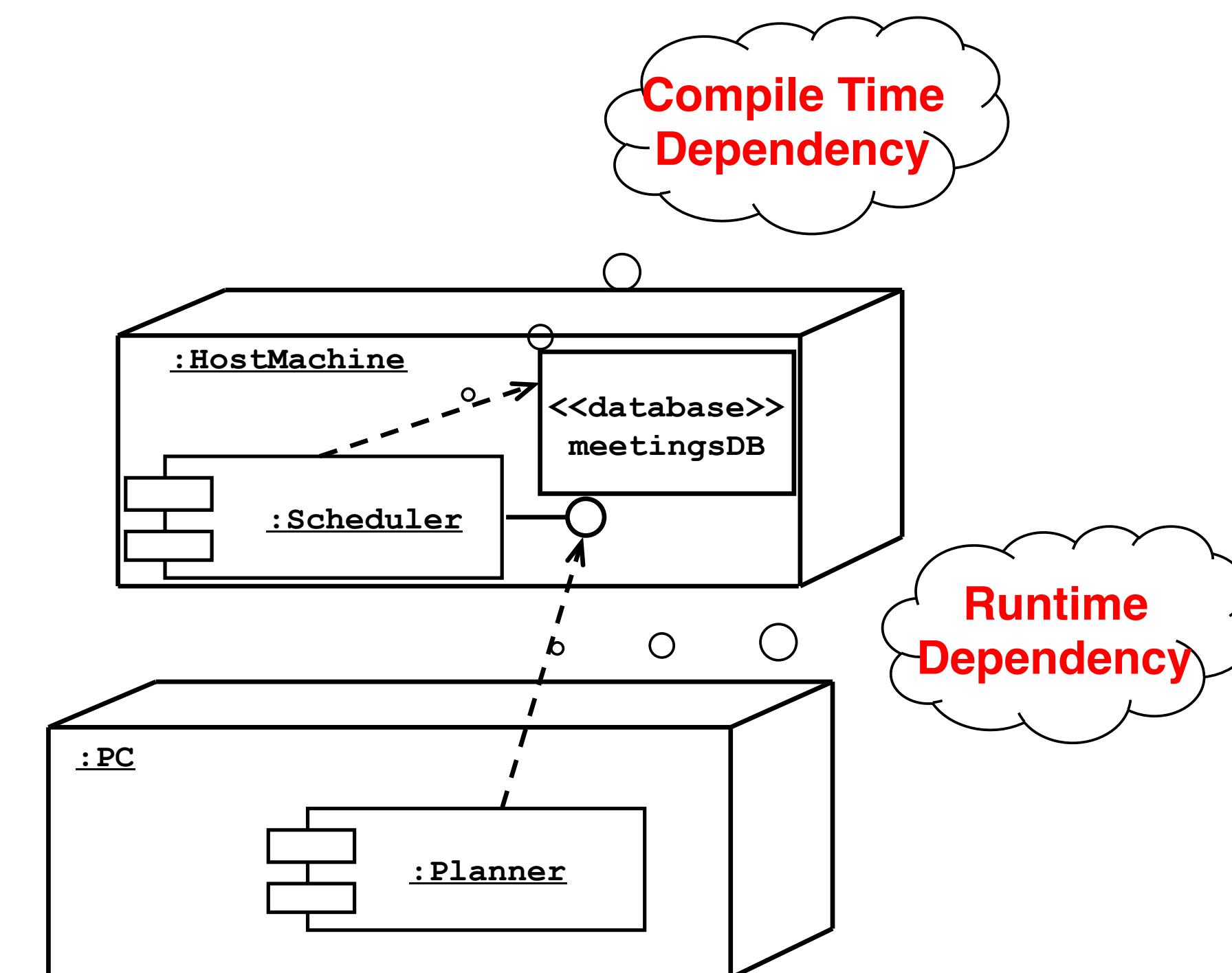
UML deployment diagram

- ~ E' usato per rappresentare le relazioni tra componenti runtime e nodi hardware
- ~ Rappresenta la allocazione dei componenti in diversi nodi e le dipendenze tra i componenti
- ~ I componenti sono entità auto contenute che forniscono servizi ad altri componenti o attori
- ~ Un nodo è qualsiasi cosa possa mandare in esecuzione del software e può appartenere a due categorie
 - ▶ Un dispositivo - hardware, può essere un computer o un componente hardware più semplice collegato al sistema
 - ▶ Ambiente di esecuzione - pezzo di software che può contenere ed eseguire un altro software

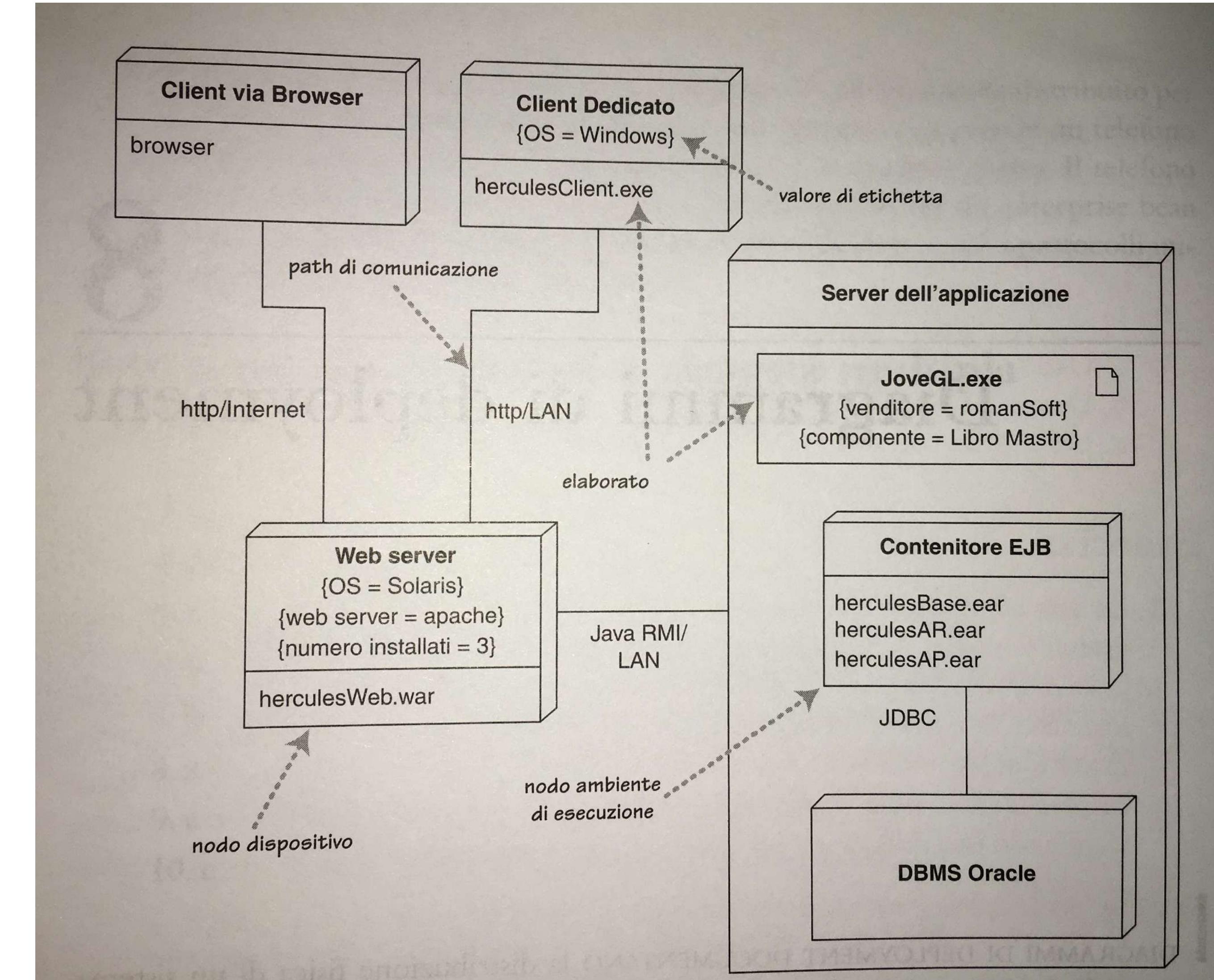
UML deployment diagram

~ I nodi contengono elaborati - manifestazione fisiche del software

- ◆ Tipicamente si tratta di file
- ◆ File jar o file eseguibili, anche file dati o di configurazione, documenti HTML etc.



UML deployment diagram



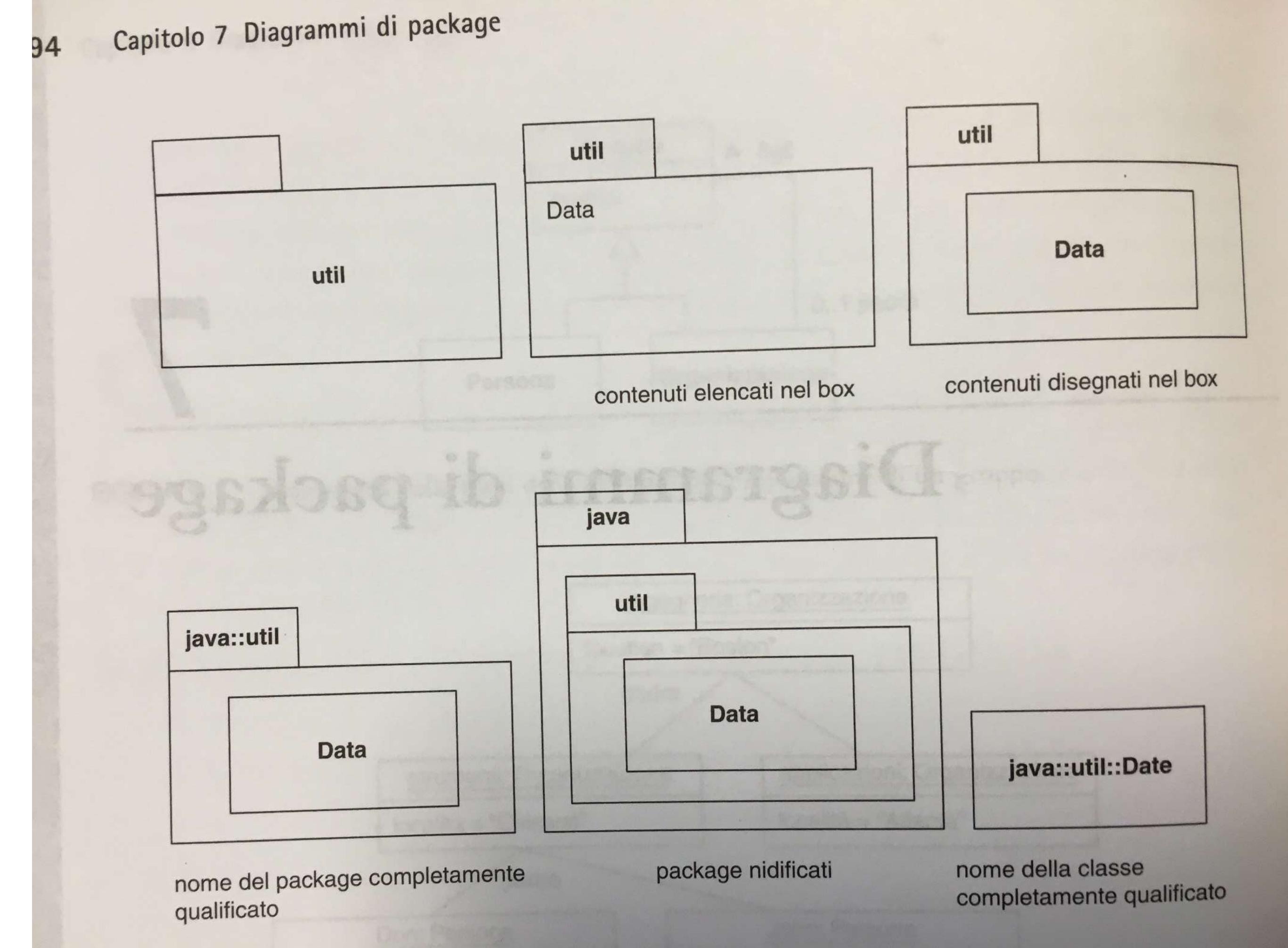
Identificare i sottosistemi

~ Trovare i sottosistemi durante il sistema design è simile a trovare gli object durante l'analisi.

Heuristics for grouping objects into subsystems

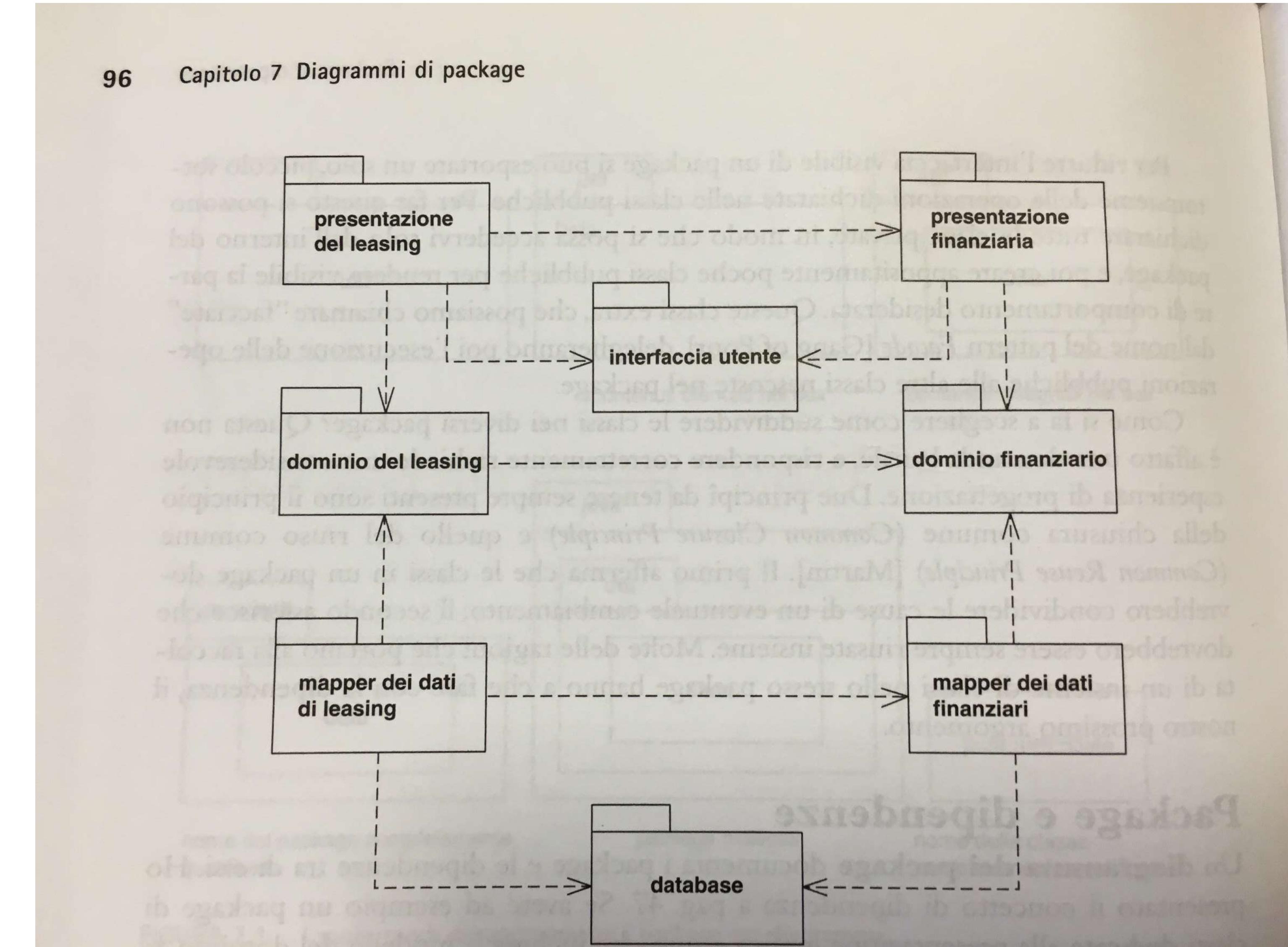
- Assign objects identified in one use case into the same subsystem.
- Create a dedicated subsystem for objects used for moving data among subsystems.
- Minimize the number of associations crossing subsystem boundaries.
- All objects in the same subsystem should be functionally related.

Modellare i sottosistemi



Modellare i sottosistemi

96 Capitolo 7 Diagrammi di package



Component diagram

- ~ Un component diagram mostra i vari componenti in un sistema e le loro dipendenze
- ~ Un componente rappresenta il modulo fisico del codice
- ~ Di solito un componente coincide con il package
- ~ Le dipendenze tra componenti mostrano come i cambiamenti ad un componente possono causare cambiamenti ad altri componenti

Diagrammi dei componenti

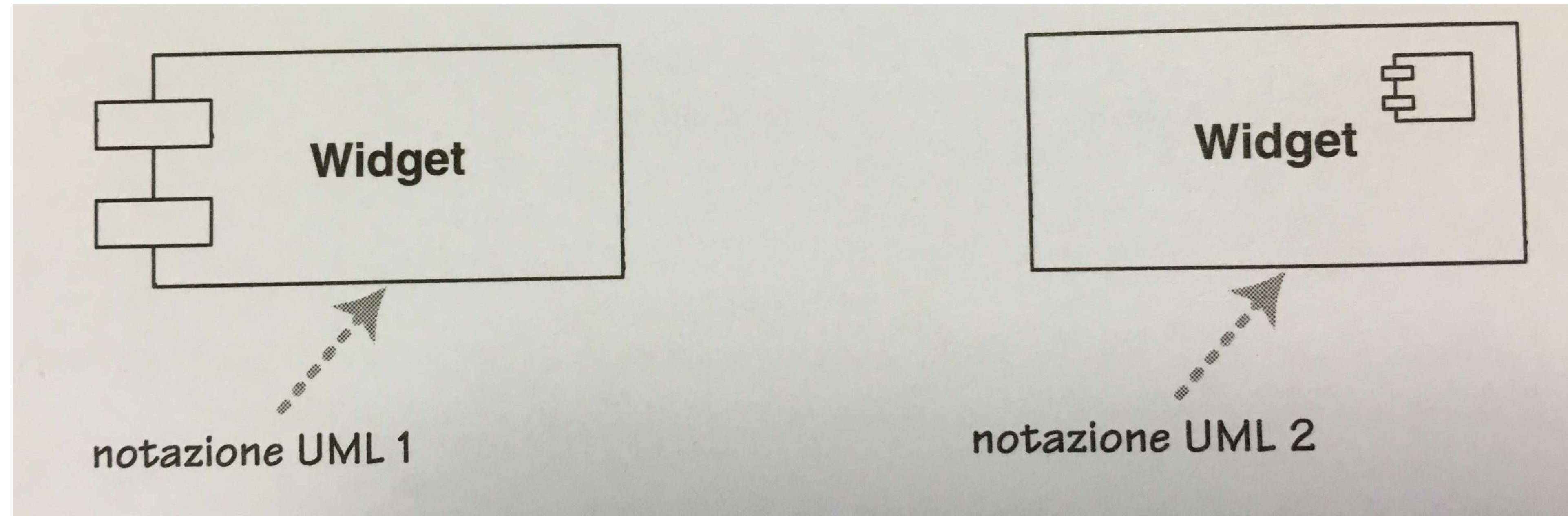
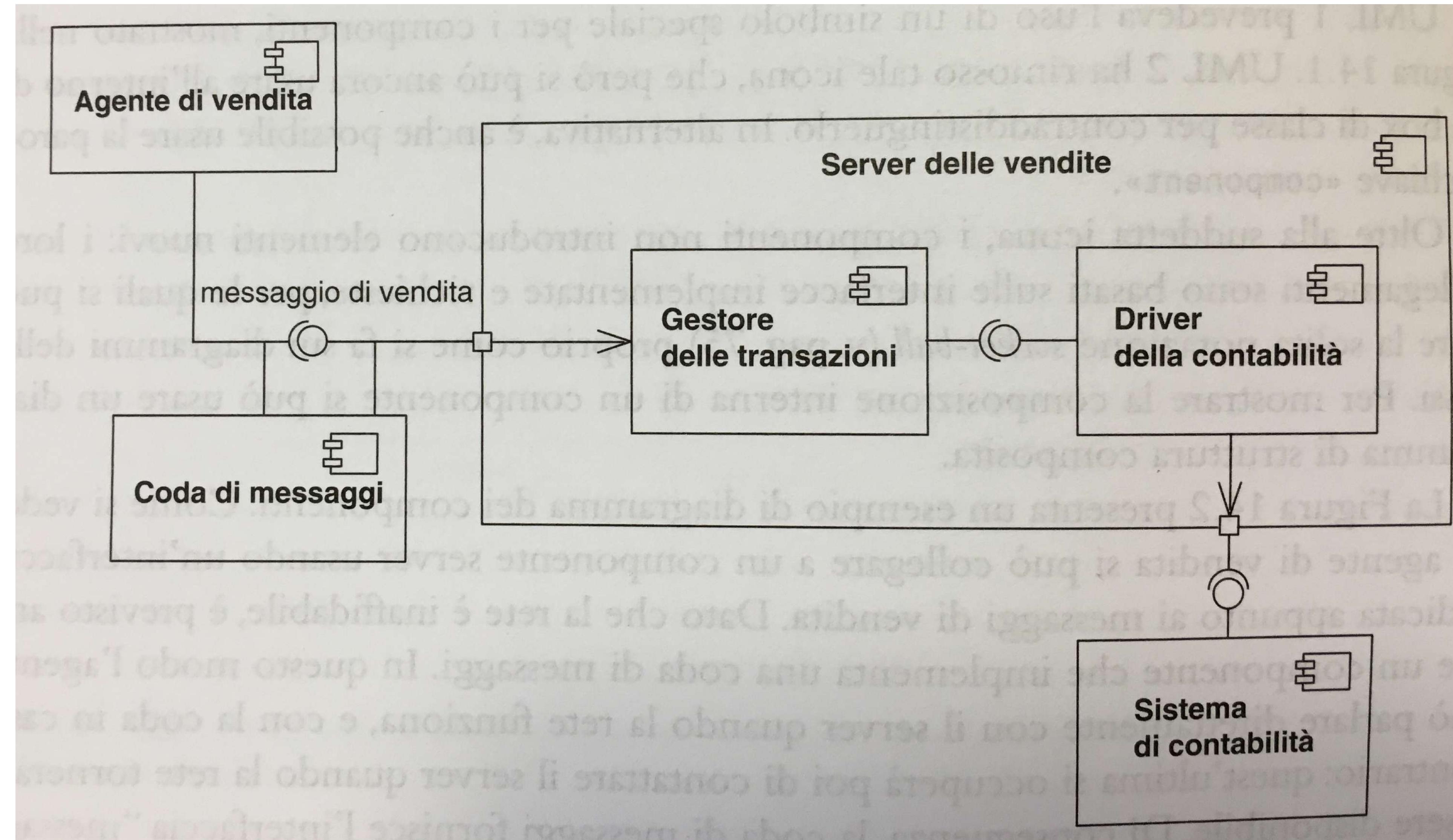
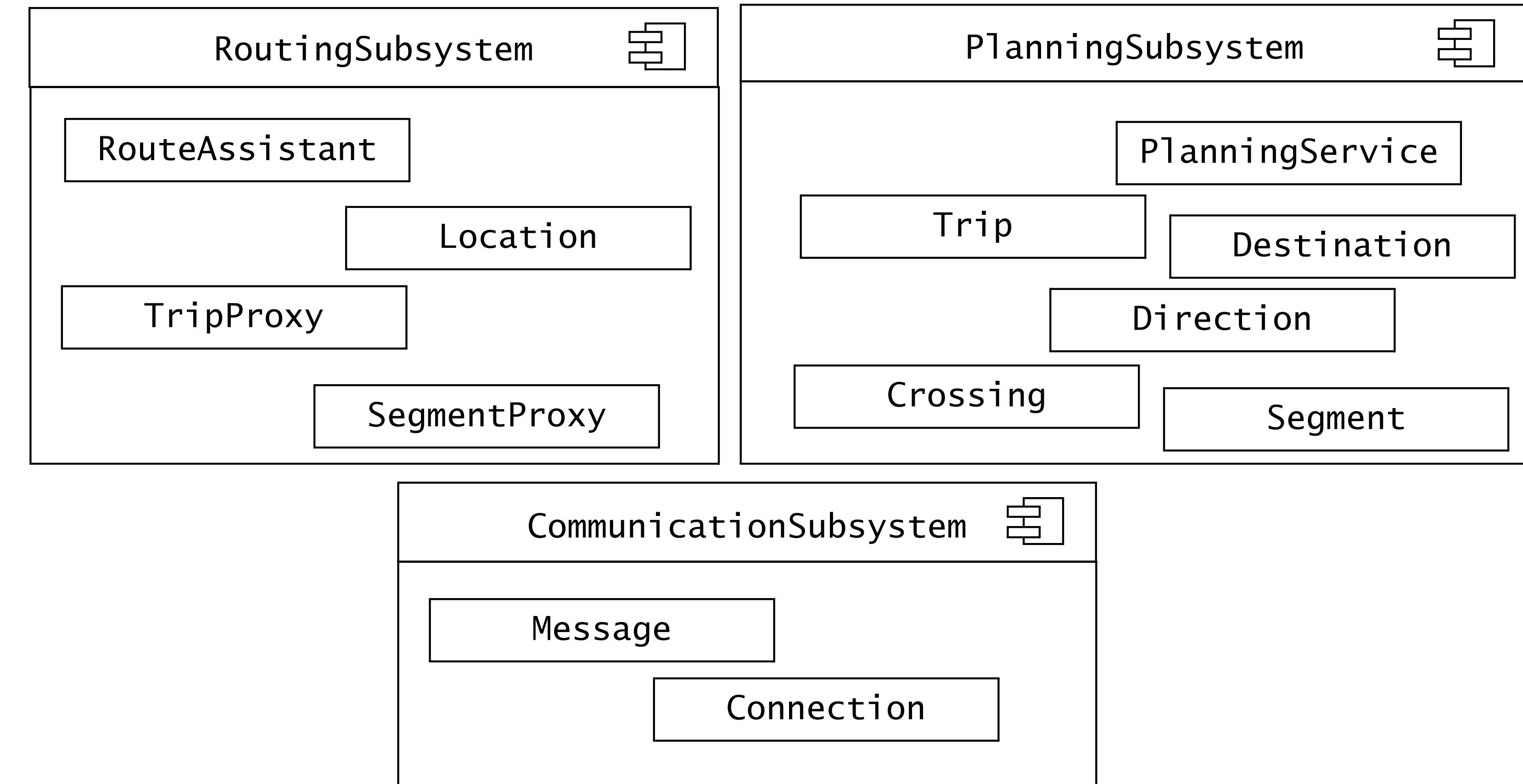
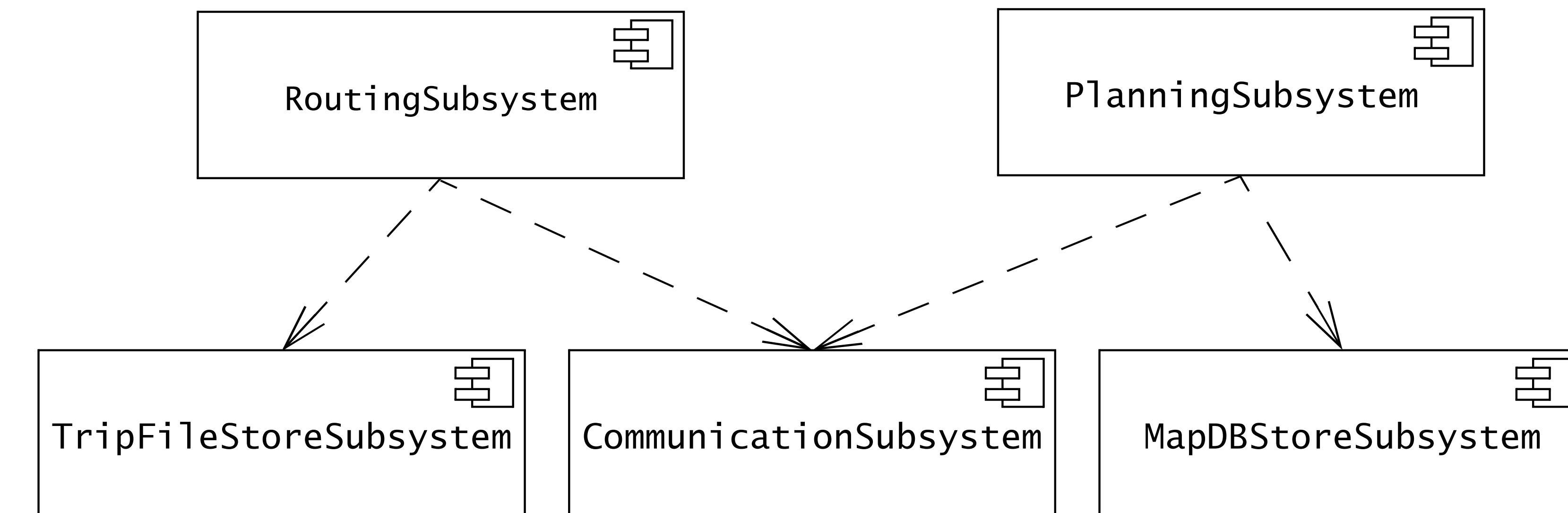
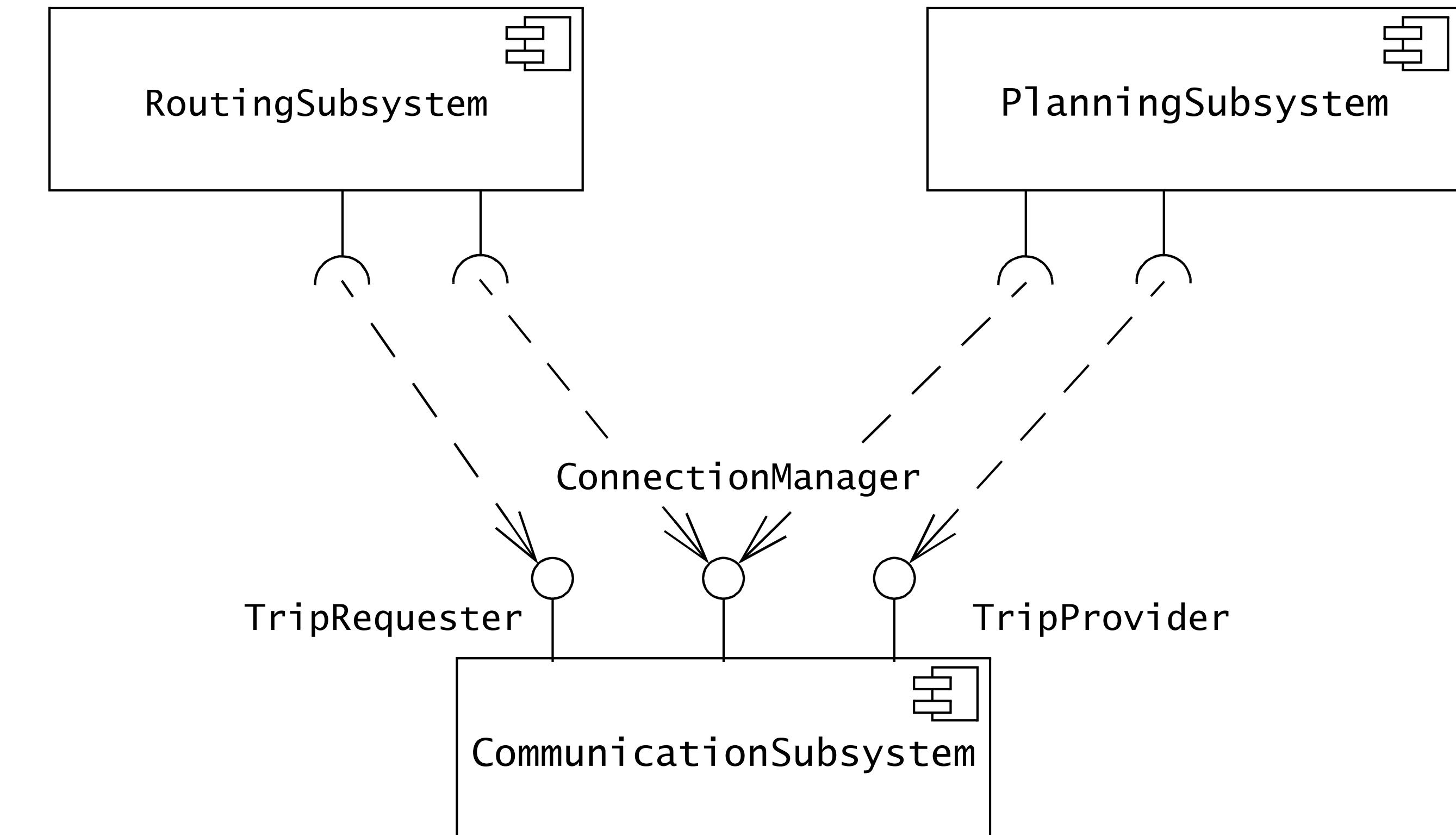


Diagramma dei componenti









System Design Document – SDD

System Design Document

1. Introduction
 - 1.1 Purpose of the system
 - 1.2 Design goals
 - 1.3 Definitions, acronyms, and abbreviations
 - 1.4 References
 - 1.5 Overview
2. Current software architecture
3. Proposed software architecture
 - 3.1 Overview
 - 3.2 Subsystem decomposition
 - 3.3 Hardware/software mapping
 - 3.4 Persistent data management
 - 3.5 Access control and security
 - 3.6 Global software control
 - 3.7 Boundary conditions
4. Subsystem services
- Glossary

I componenti non sono una tecnologia, anche se chi si occupa di tecnologia sembra avere difficoltà a capirlo. I componenti hanno a che fare con la modalità con cui gli utenti vogliono porsi in relazione al software. Vogliono poter comprare software un pezzo per volta, e aggiornarlo così come si aggiorna uno stereo. Vogliono che i pezzi nuovi funzionino perfettamente insieme ai vecchi, e vogliono aggiornare il sistema seguendo il proprio ritmo, e non quello dei produttori. Vogliono essere in grado di mescolare senza problemi pezzi sviluppati da produttori diversi. Questo è un requisito molto ragionevole: è solo difficile da soddisfare.

Ralph Johnson, <http://www.c2.com/cgi/wiki?DoComponentsExist>