



Università  
Ca' Foscari  
Venezia

# Progettazione Architettuale



# Progettazione Architeturale

- Obiettivo: stabilire la struttura globale di un sistema software
- Descriveremo diversi tipi di modello di architettura, e come l'architettura di un singolo sistema può essere modellata in diversi modi



# Processo di progettazione architettonica

- Strutturazione del sistema
  - Il sistema viene diviso in sottosistemi, e vengono identificato come i sottosistemi comunicano
- Modellazione del controllo
  - Viene stabilito un modello delle relazioni di controllo tra le differenti parti del sistema
- Scomposizione modulare
  - I sottosistemi vengono a loro volta divisi in moduli



# Sottosistemi e moduli

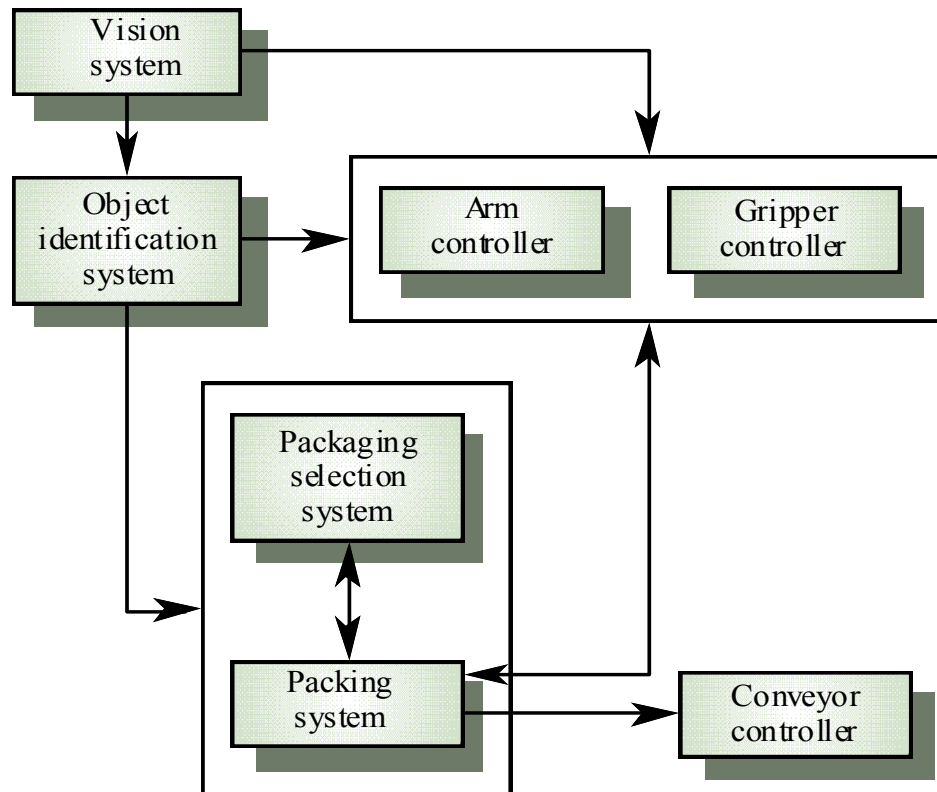
- Un sottosistema è un sistema di per sè: le sue operazioni sono indipendenti dai servizi offerti dagli altri sotto-sistemi
- Un modulo è una componente del sistema che offre servizi ad altre componenti, ma che non può essere considerato un sistema a se stante



# Strutturare un sistema

- Significa scomporre il sistema in sottosistemi
- L'architettura viene espressa di solito con un diagramma a blocchi, che presenta una visione d'insieme della struttura del sistema
- Si possono sviluppare modelli più specifici, che mostrano come i sottosistemi siano distribuiti e condividono dati, e che mostrano le interfacce tra i vari sottosistemi

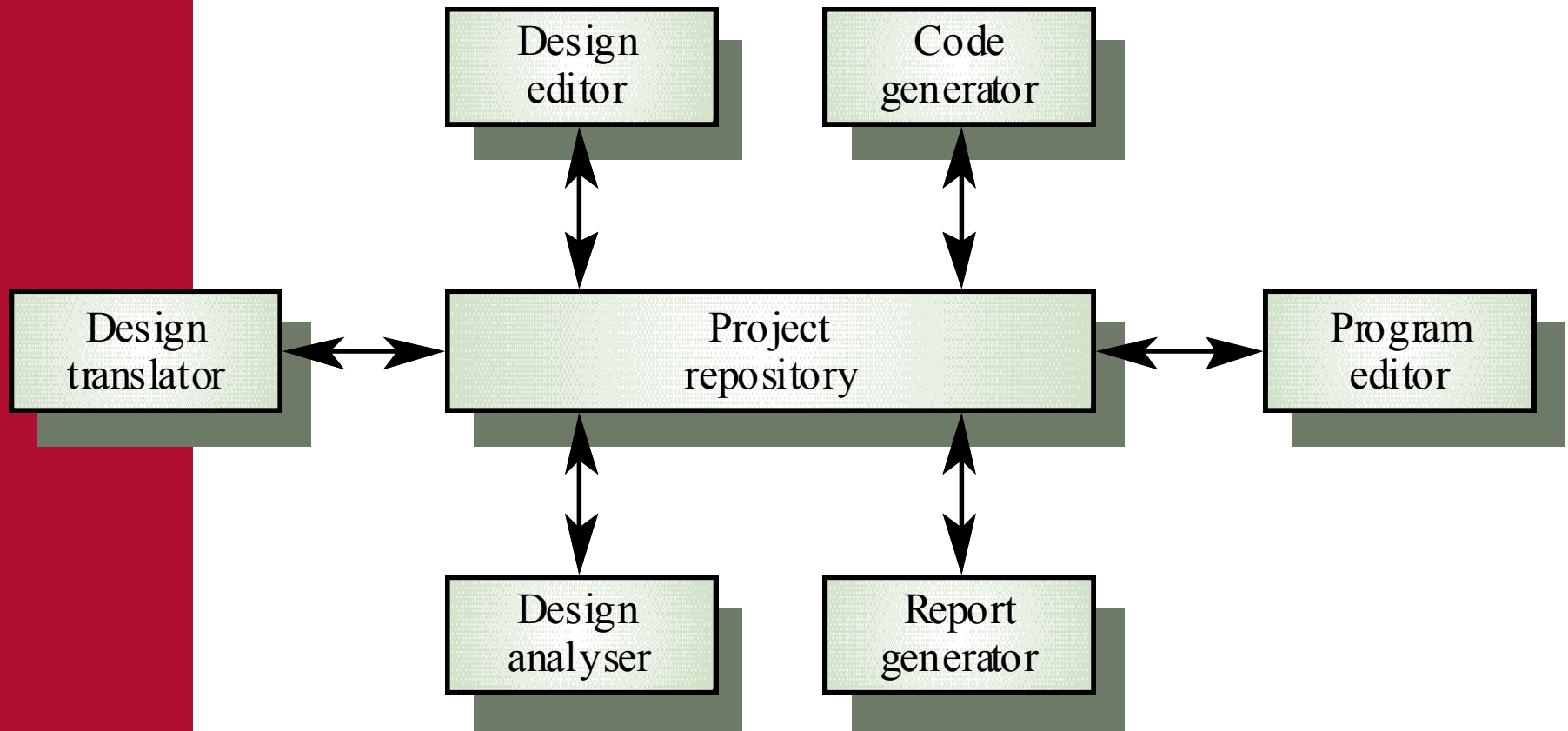
# Sistema di controllo di un robot di impacchettamento



# Modelli di gestione dei dati

- I sottosistemi possono scambiarsi dei dati. Questo può essere fatto in due modi:
  - I dati condivisi sono mantenuti in un database centrale (o deposito) al quale possono accedere tutti i sottosistemi
  - Ogni sottosistema mantiene un proprio database e passa i dati esplicitamente agli altri sottosistemi
- Quando devono essere condivise grosse quantità di dati, viene usualmente scelto il modello di deposito (repository model)

# Architettura di un toolset CASE





# Caratteristiche del modello repository

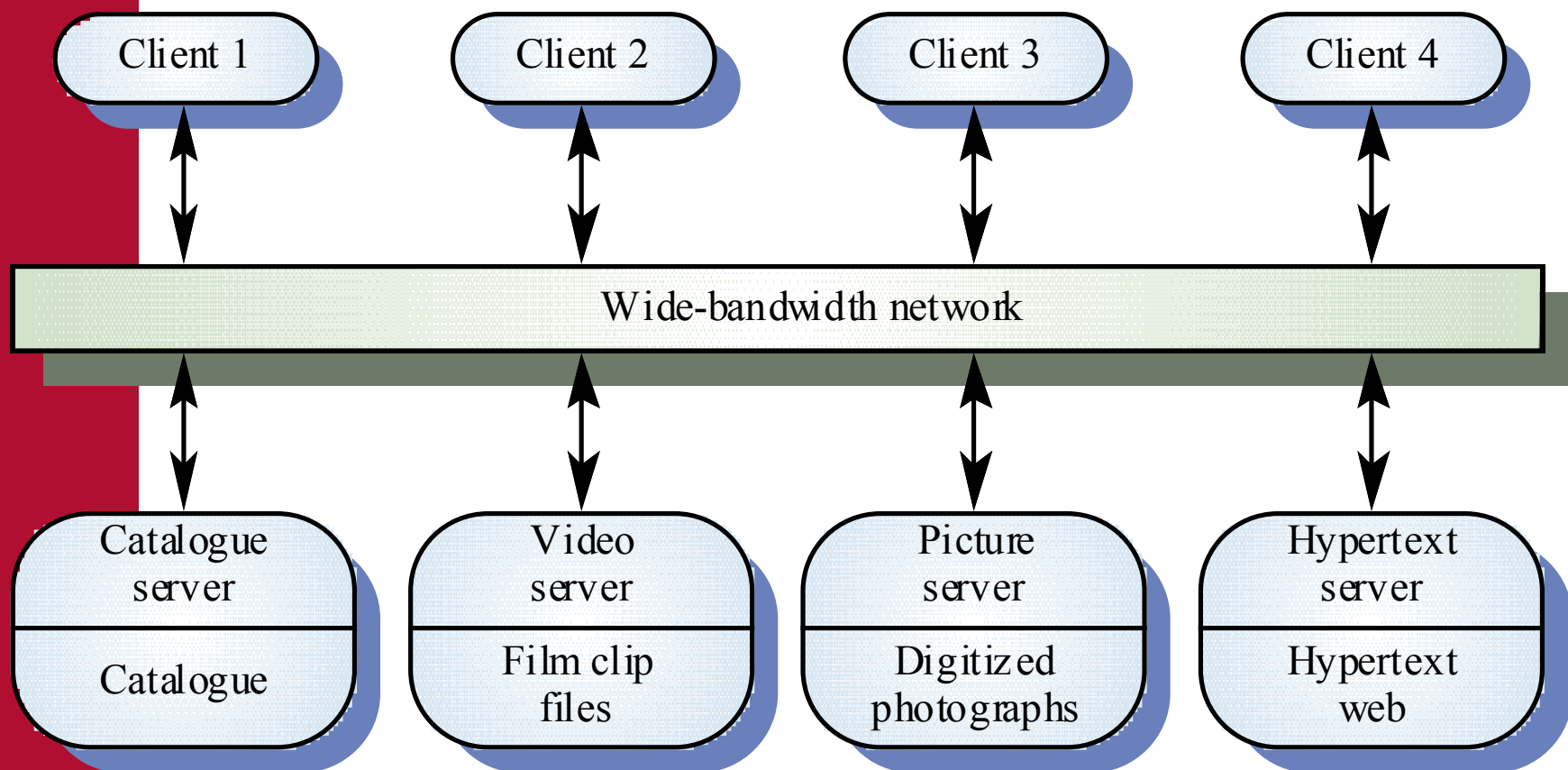
- Vantaggi
  - Modo efficiente di condividere grandi quantità di dati
  - I sottosistemi possono disinteressarsi a come i dati vengono prodotti, alle operazioni di backup, sicurezza ecc.
- Svantaggi
  - I sottosistemi devono concordare su un modello dei dati, che inevitabilmente è un compromesso tra esigenze diverse
  - L'evoluzione dei dati è difficoltosa e dispendiosa
  - Non c'è spazio per politiche specifiche di gestione dei dati
  - La distribuzione dei dati non è efficiente



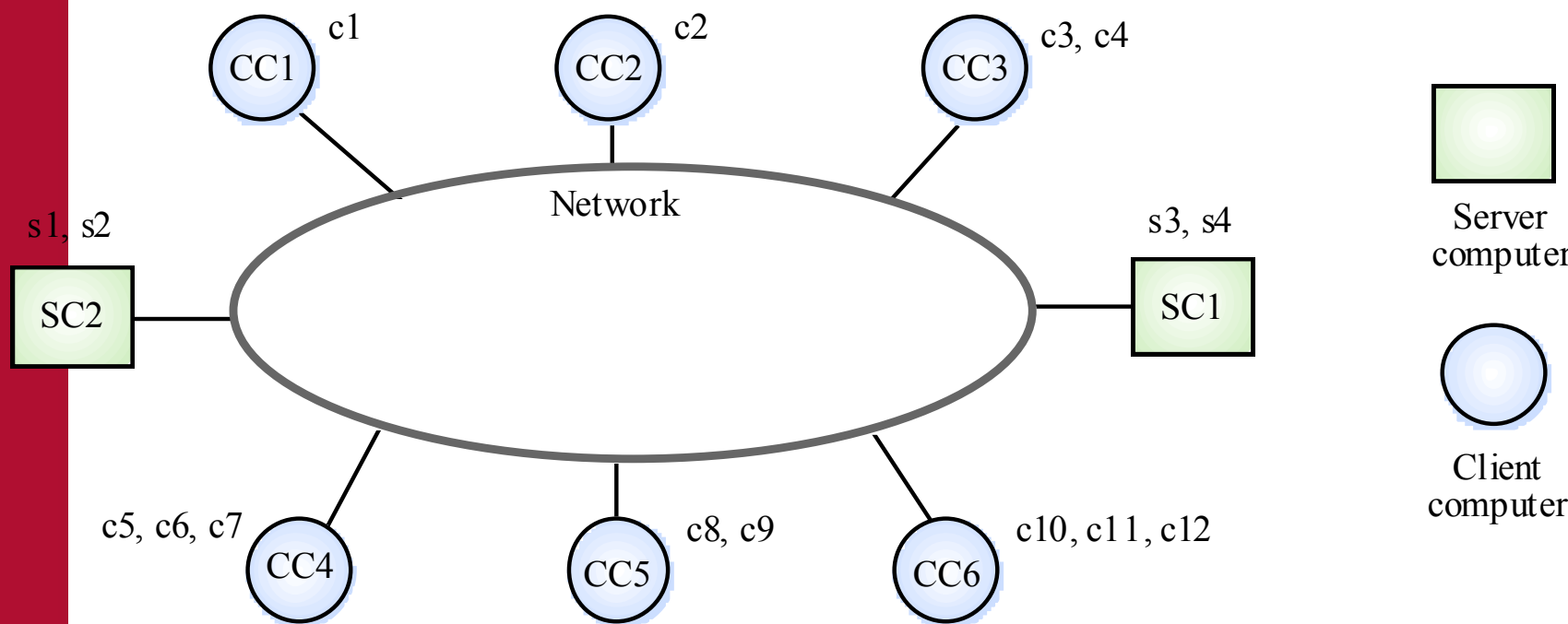
# Modello “client-server”

- Modello per sistemi distribuiti, che mostra come dati e processi sono distribuiti su un insieme di componenti
- Insieme di servers autonomi che offrono servizi specifici, ad es. di stampa, di gestione dei dati...
- Insieme di clienti che richiedono questi servizi
- Rete che permette ai clienti di accedere ai server

# Emeroteca

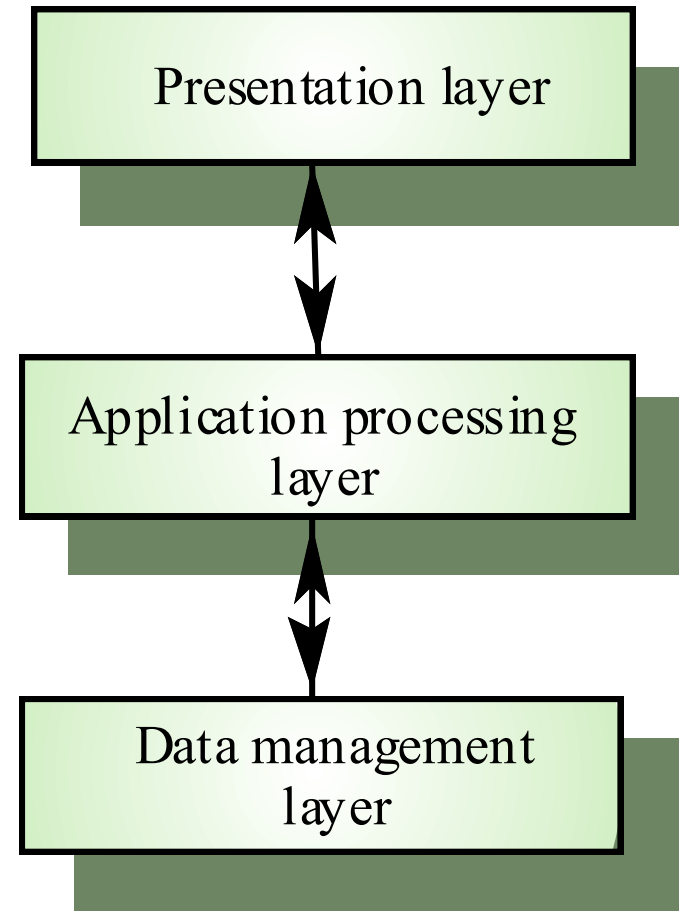


# Computers in una rete client-server

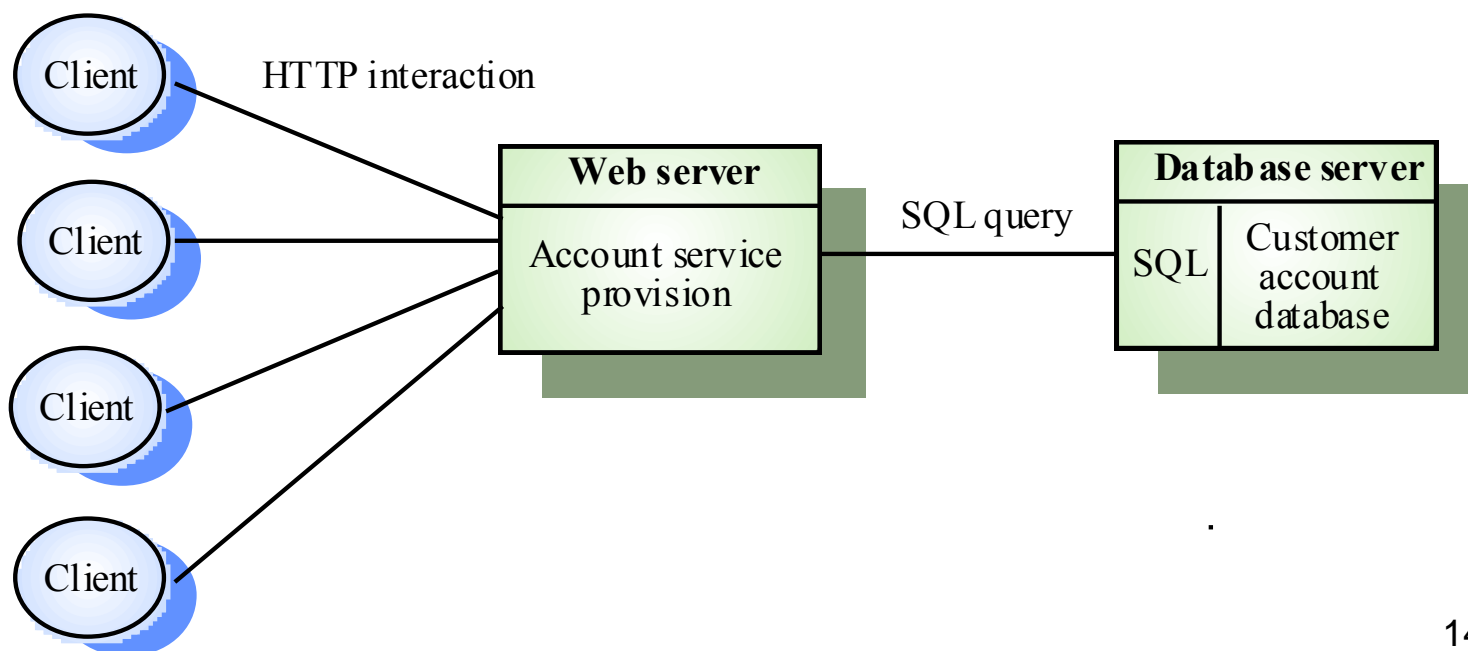
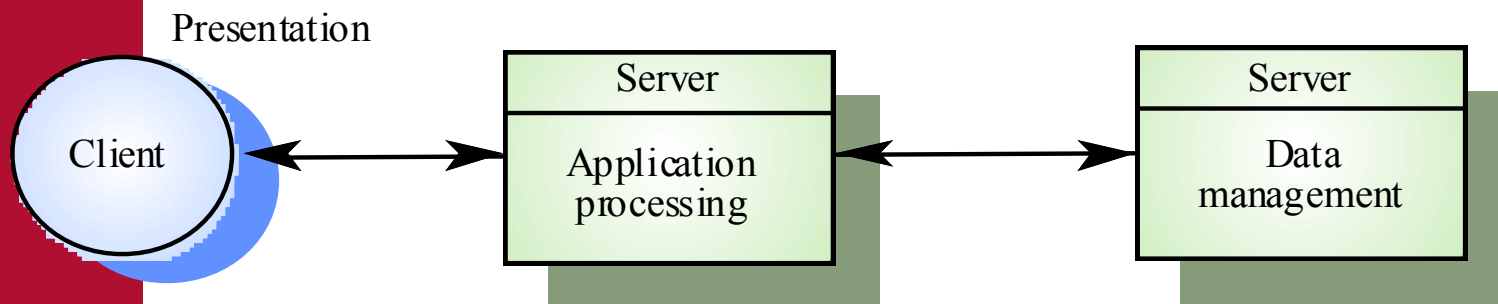


# Strati funzionali nell'architettura client-server (three tier architecture)

- **Presentation layer**
  - Si preoccupa di presentare i risultati della computazione agli utenti del sistema e di gestire gli input da parte degli utenti.
- **Application processing layer**
  - Si preoccupa di offrire le funzionalità specifiche dell'applicazione
- **Data management layer**
  - Si preoccupa di gestire la comunicazione con il DBMS



# Three-tier architecture – struttura ed esempio





# Caratteristiche del modello client-server

- Vantaggi
  - La distribuzione dei dati è semplice
  - Fa uso effettivo del sistema di rete (hardware economico)
  - E' facile aggiungere nuovi server o fare l'upgrade dei server esistenti
- Svantaggi
  - Non c'è un modello dei dati condiviso da tutti i sottosistemi, quindi i sottosistemi possono usare organizzazioni diverse dei dati e lo scambio dei dati può risultare inefficiente
  - Gestione di dati ridondante in ogni server
  - Non c'è un registro centrale dei nomi e dei servizi: può risultare difficile sapere quali dati/servizi sono disponibili



# Modello “macchina astratta”

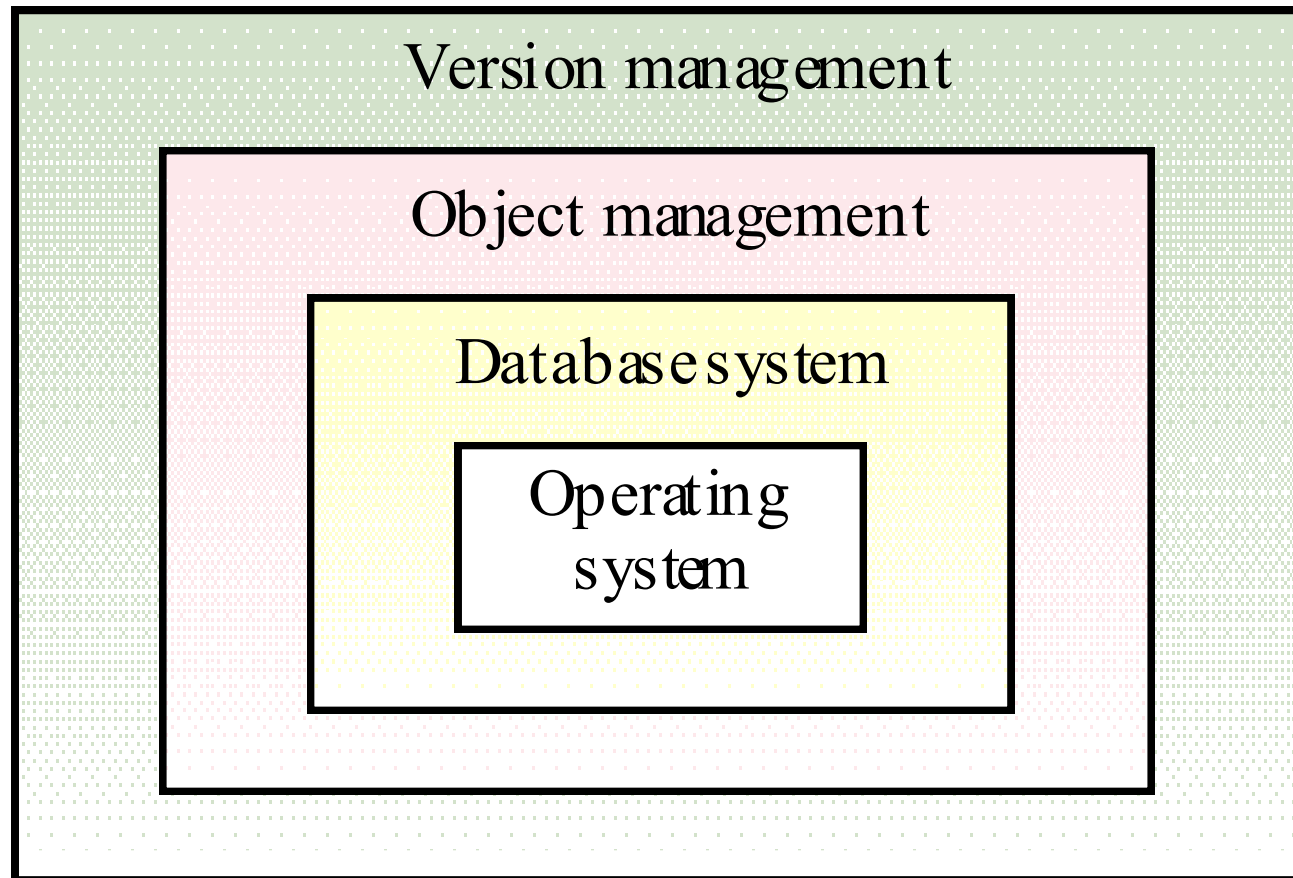
- Usato per modellare l'interfaccia tra sottosistemi
- Organizza il sistema in un insieme di livelli (o macchine astratte) ognuno dei quali offre un insieme di servizi
- Supporta lo sviluppo incrementale di sottosistemi a livelli diversi: quando l'interfaccia di un livello cambia, ne è affetto solo il livello adiacente





Università  
Ca' Foscari  
Venezia

# Sistema di gestione delle versioni





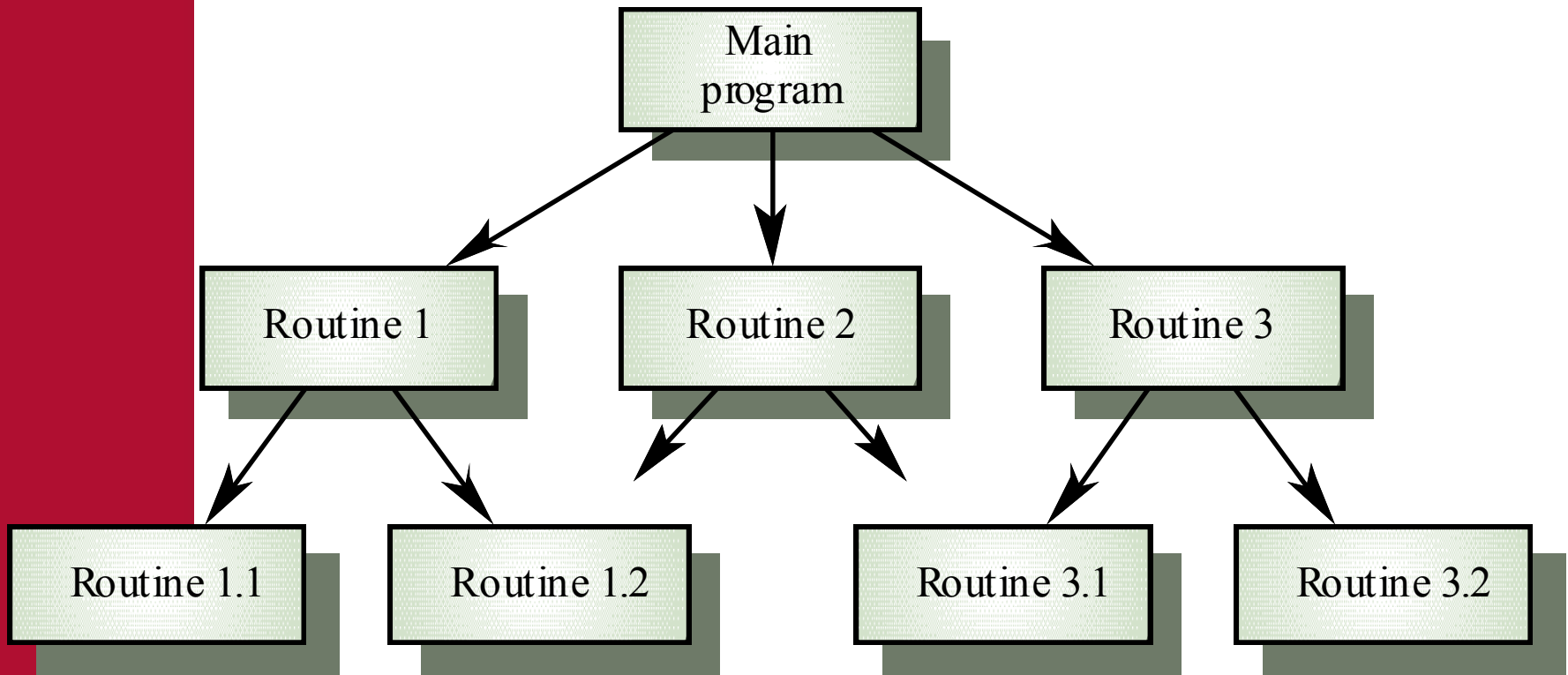
# Riassumendo (1)

- Modelli di struttura di un sistema:
  - Repository
  - Client-server
  - Macchina astratta

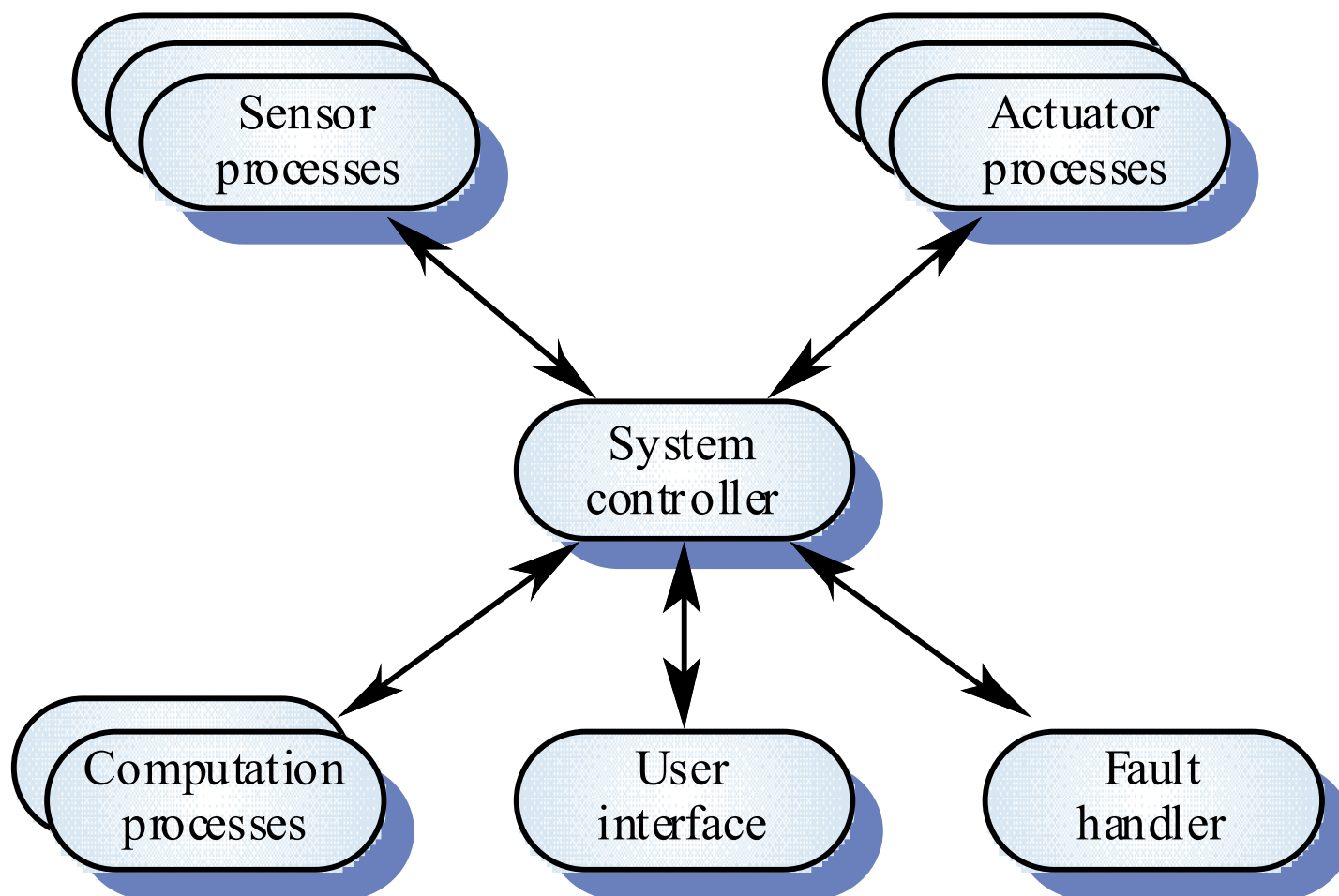
# Modelli di controllo

- Controllo **centralizzato**
  - Un sottosistema ha il controllo globale e dà inizio e fine agli altri sottosistemi
  - Modello “**call-return**”: gerarchia di procedure, in cui il controllo viene gestito top-down. Applicabile a sistemi sequenziali.
  - Modello “**manager**”: una componente del sistema controlla l'interruzione, l'inizio e il coordinamento degli altri processi. Applicabile a sistemi concorrenti. Può essere implementato in sistemi sequenziali con un comando “case”.
- Controllo **basato sugli eventi**
  - Ogni sottosistema può rispondere ad eventi generati dall'esterno da altri sottosistemi o dall'ambiente esterno al sistema

# Modello Call-return



# Modello Manager



# Modello basato sugli eventi

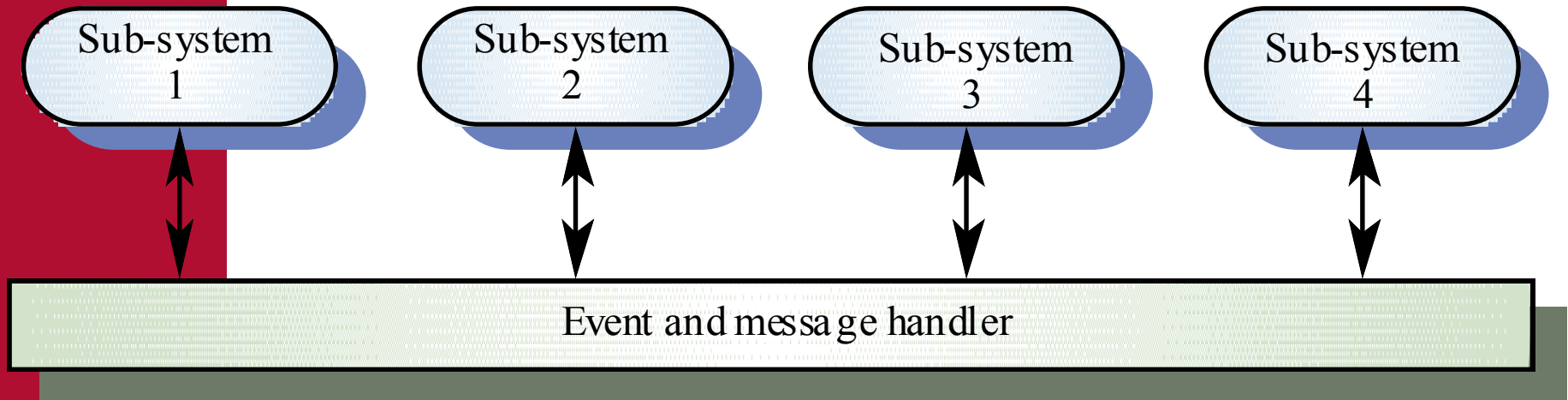
- Guidato da eventi generati dall'esterno, dove la temporizzazione degli eventi è fuori dal controllo dei sottosistemi che gestiscono l'evento
- Due modelli “event-driven” principali
  - Modello “broadcast”: un evento è trasmesso a tutti i sottosistemi. Ogni evento è trasmesso a tutti i sottosistemi. Ogni sottosistema in grado di gestire l'evento può farlo.
  - Modello “interrupt-driven”. Usato nei sistemi real-time dove le interruzioni sono individuate da ogni gestore di interruzioni e passate a qualche altra componente per processarle



# Modello broadcast

- Efficace per integrare sottosistemi su diversi computer in rete
- Ogni sottosistema si dichiara “interessato” a degli eventi specifici. Quando questi occorrono, il controllo viene trasferito ai sottosistemi che possono gestirlo
- Le politiche di controllo non sono interne all'evento o al gestore del messaggio. E' tutto lasciato ai sottosistemi. Tuttavia i sottosistemi non sanno se e quando l'evento sarà gestito.

# Broadcasting selettivo

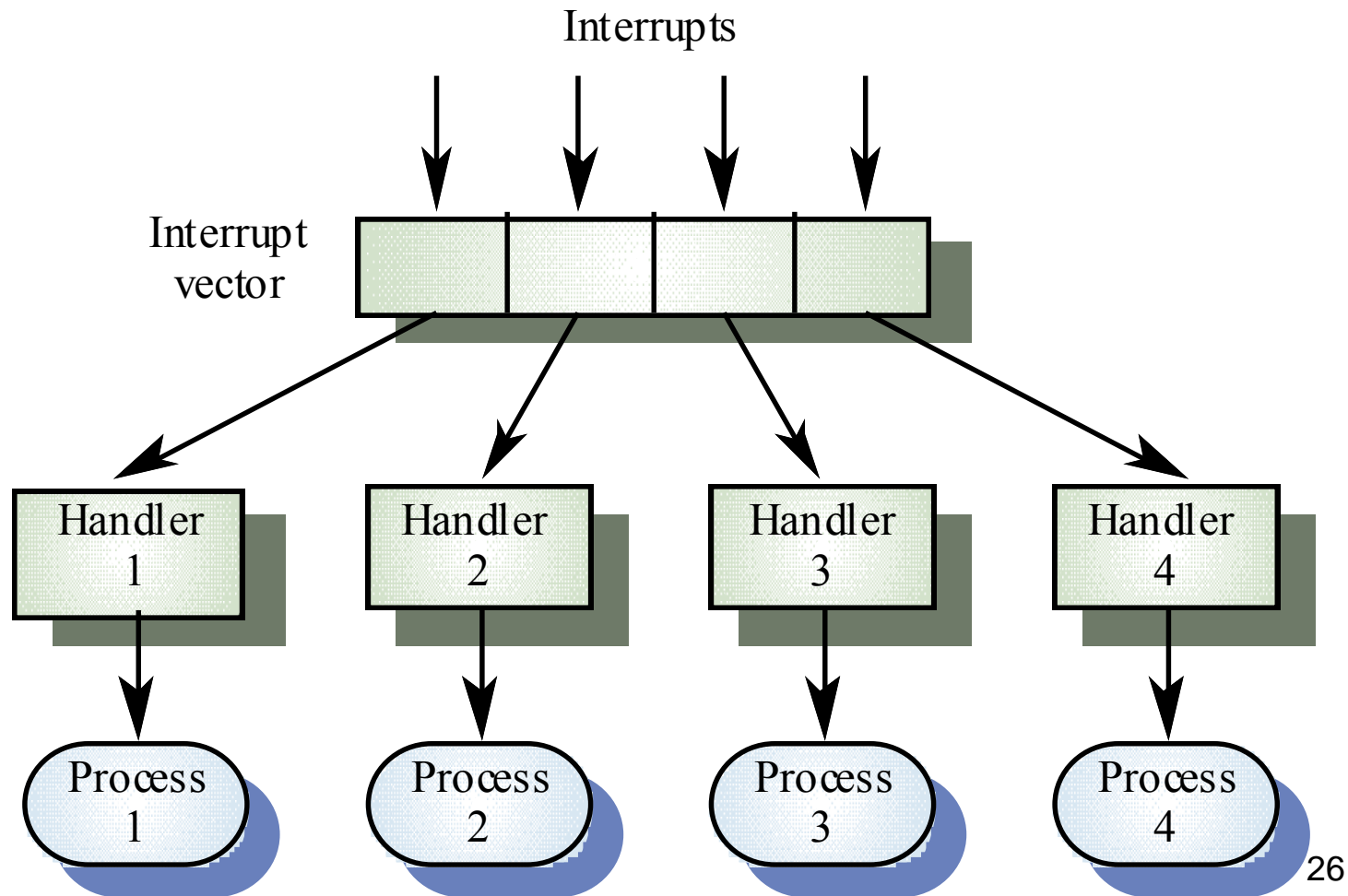




# Sistemi “interrupt-driven”

- Usati in sistemi real-time, dove la velocità di risposta ad un evento è essenziale
- Ci sono tipi di interruzione noti, con un gestore definito in corrispondenza ad ogni tipo
- Ogni tipo è associato a una locazione di memoria e uno switch hardware lo trasferisce al suo gestore
- Consentono rapidità di risposta, ma sono complicati da programmare e difficili da validare

# Controllo interrupt-driven





# Riassumendo (2)

- Modelli di controllo centralizzato
  - Call-return
  - Manager
- Modelli di controllo basato su eventi
  - Broadcast
  - Interrupt-driven

# Scomposizione modulare

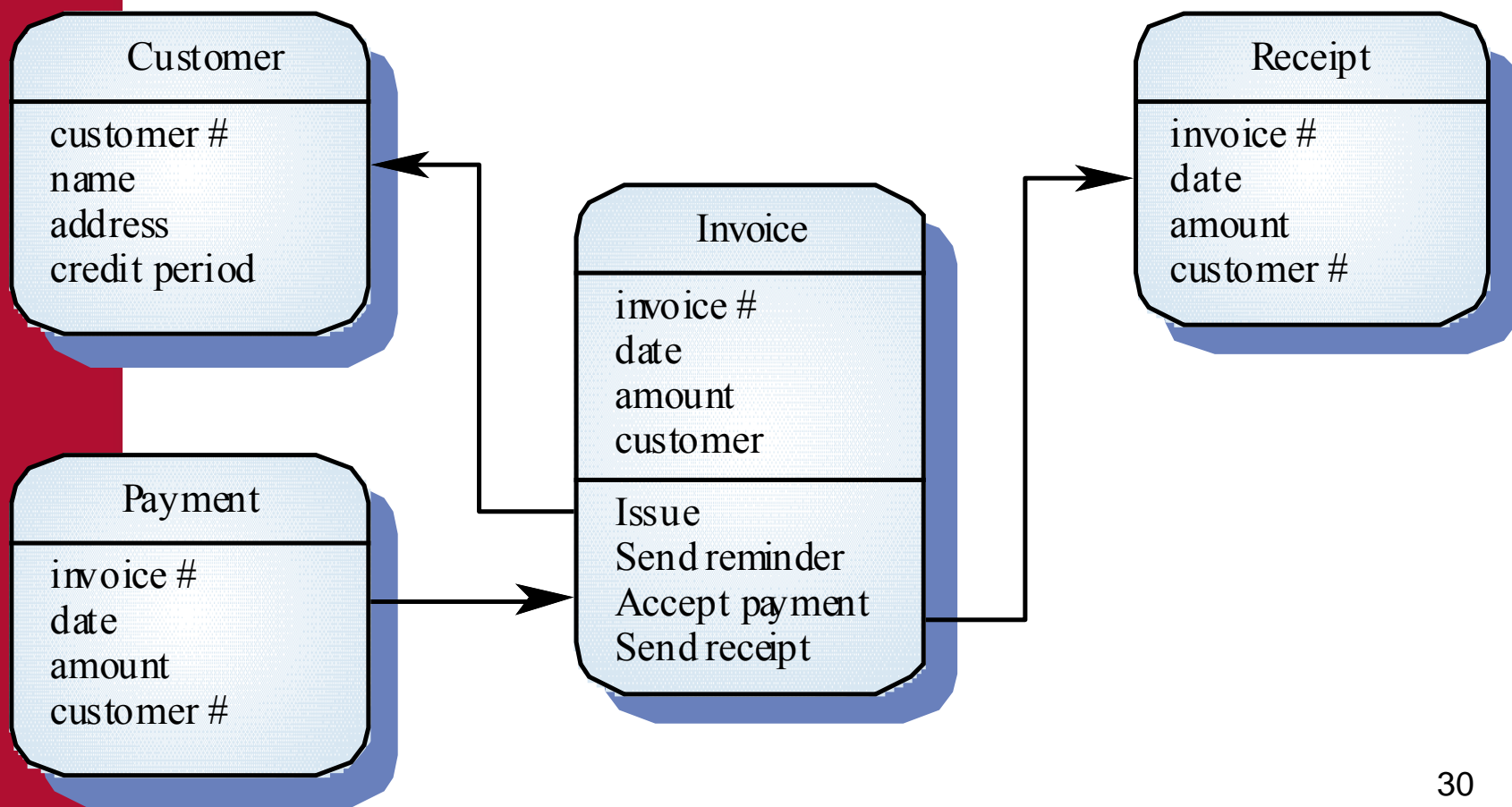
- E' un ulteriore livello strutturale: i sottosistemi sono scomposti in moduli
- Due modelli di scomposizione:
  - Un modello a **oggetti**, dove il sistema è scomposto in oggetti che interagiscono
  - Un modello **data-flow**, dove il sistema è scomposto in moduli funzionali che trasformano inputs in outputs (modello pipeline)
- Se possibile, ogni decisione rispetto alla concorrenza dovrebbe essere posticipata fino all'implementazione dei moduli



# Modelli a oggetti

- Consiste nello strutturare il sotto-sistema in un insieme di oggetti con accoppiamento lasco e interfacce ben definite
- La scomposizione orientata ad oggetti significa identificare le classi degli oggetti, gli attributi (campi) e le operazioni (metodi)
- Quando vengono implementati, gli oggetti sono istanze di queste classi e qualche modello di controllo è usato per coordinarne i metodi

# Sistema di smaltimento fatture

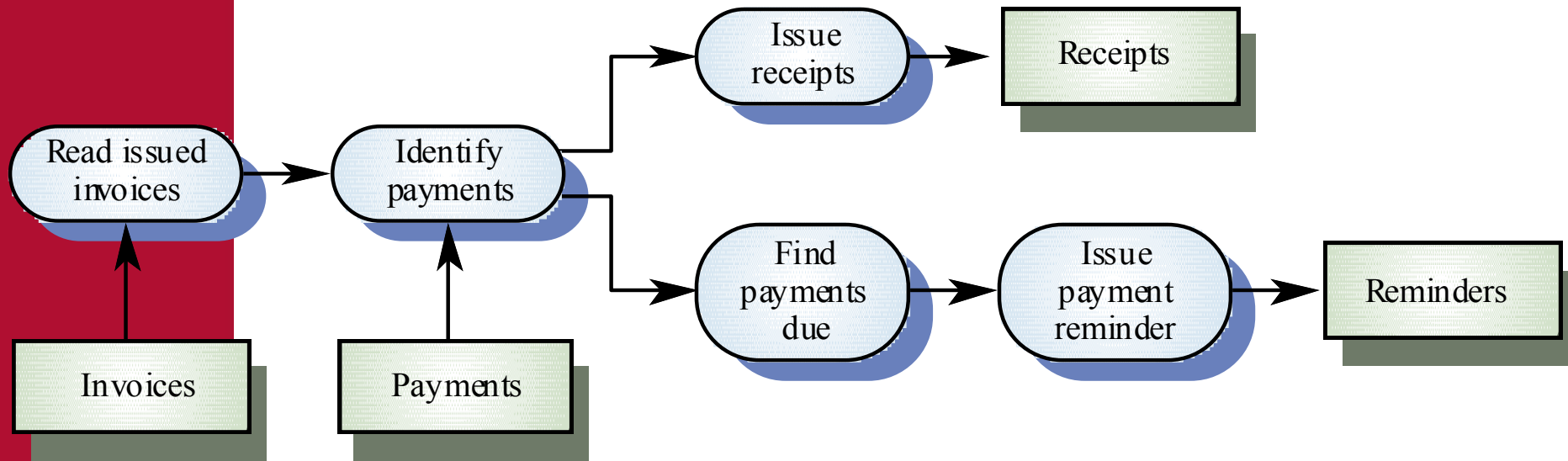




# Modelli “data-flow”

- Trasformazioni funzionali che producono un output a partire da un input
- Possono riferirsi ad un modello a “pipe” e a filtri (come la shell Unix)
- Quando le trasformazioni sono sequenziali, è un modello batch sequenziale molto usato nei sistemi di gestione dati
- Il modello “data-flow” non si adatta bene a sistemi interattivi

# Sistema smaltimento fatture

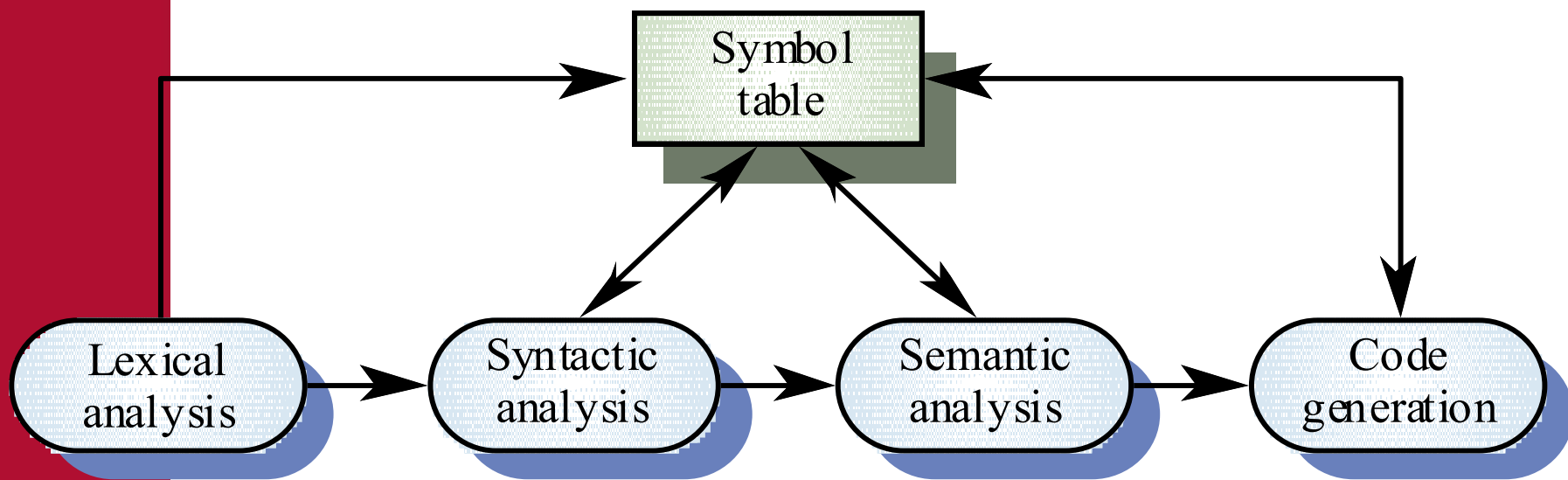




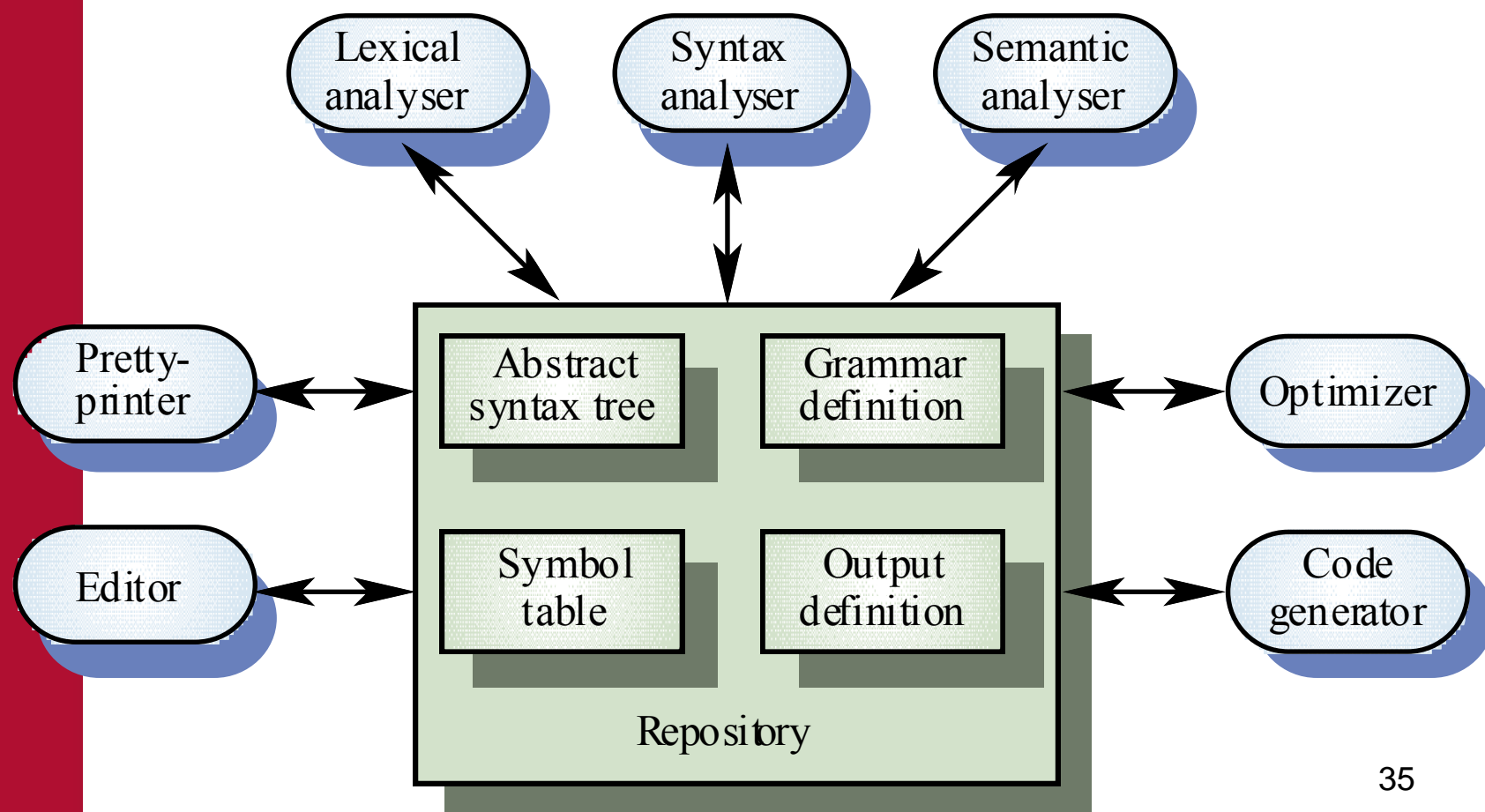
# Architetture “domain-specific”

- Modelli di architettura di sistema che sono specifici rispetto a qualche dominio di applicazione
- Due tipi di modelli “domain-specific”:
  - Modelli generici: sono astrazioni da un certo insieme di sistemi reali e contengono le caratteristiche principali di questi sistemi  
Di solito sono modelli bottom-up
  - Modelli di riferimento: sono modelli idealizzati, più astratti. Offrono informazione su quella classe di sistemi e permettono di confrontare diverse architetture  
Di solito sono modelli top-down

# Modelli generici: un compilatore



# Modelli generici: un ambiente di sviluppo software





# Architetture di riferimento

- Modelli di riferimento: sono ottenuti studiando il dominio di applicazione piuttosto che i sistemi esistenti
- Possono essere usati come base per implementare sistemi o per confrontare diversi sistemi.
- Hanno il ruolo di “standard” rispetto ai quali valutare un sistema
- Esempio: il modello OSI è un modello di riferimento per i sistemi di comunicazione



Università  
Ca' Foscari  
Venezia

# Modello di riferimento OSI

