Sistemi Informativi:

Progetto del sistema

Argomenti

- concorrenza
- mapping HW/SW
- dati persistenti
- controllo accessi e risorse globali
- controllo SW
- condizioni al contorno

concorrenza (1)

- Importante nei **server**
- identificare i flussi di controllo ed i problemi di concorrenza (diagramma di attivita')
- obiettivo: prestazioni, tempo di risposta
- Thread:
  - E' un miniprocesso all'interno di un singolo programma, meno costoso di un processo vero e proprio. Gestito dal singolo programma e non dall'OS

concorrenza (2)

- due oggetti sono inerentemente concorrenti se possono ricevere eventi contemporaneamente senza interagire
- oggetti di questo tipo sono assegnati a thread differenti
- oggetti con attivita’ mutuamente esclusiva (alternativa) sono assegnati allo stesso thread (perche’?)

Quesiti sulla concorrenza

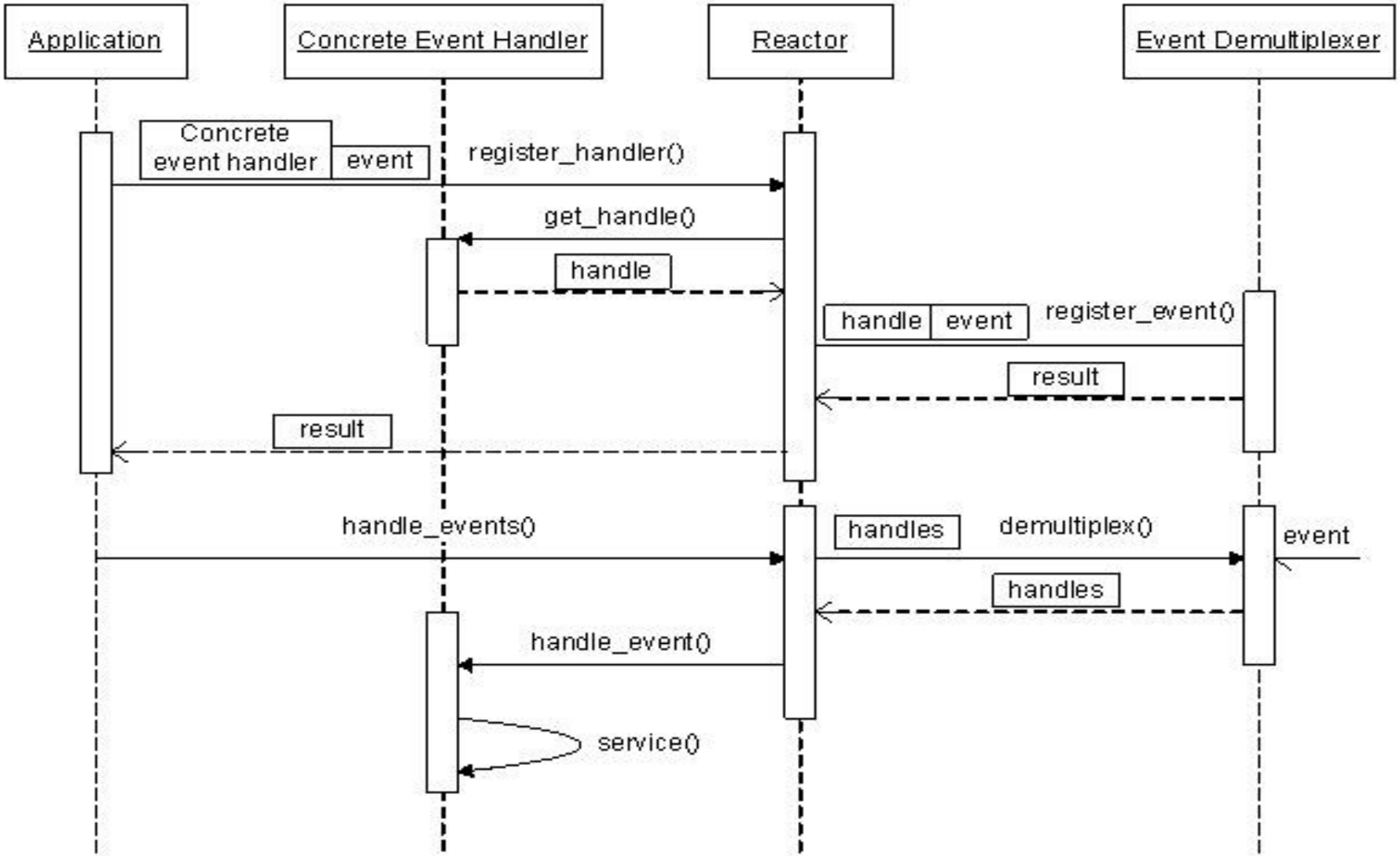
- quali oggetti sono indipendenti?
- quali thread di controllo sono identificabili?
- il sistema e’ multiutente?
- si puo’ decomporre una richiesta in un insieme di richieste parallele?

concorrenza

- implementazione fisica
  - hardware
    - se gli oggetti non interagiscono
- implementazione logica
  - software
    - se gli oggetti accedono a risorse condivise
- vantaggi/svantaggi?

Threads ed efficenza

- Un thread dell'OS e' costoso
- Utile quando l'attivita' potrebbe essere sospesa (es I/O)
- In certi casi se non abbiamo attivita' che bloccano la comnputazione e non si usano l thread si usa il pattern “Reactor”


Application
Concrete Event Handler
Reactor
Event Demultiplexer
Concrete event handler
event
register_handler()
get_handle()
handle
handle
event
register_event()
result
result
handle_events()
handles
demultiplex()
event
handles
handle_event()
service()

Reactor scala bene se..

- Mai bloccare in un thread reattivo
- No stato condiviso o semafori
- L' attivita' deve essere CPU bound
- IO asincrono o in un altro gruppo di thread; risultato spedito al reattore
- Si usano callback , neccessaria disciplina per uso sostenibile
- Piccole dosi di lavoro
- Su macchina con 8 cpu non si possono avere arrivo contemporaneo di 8 task lunghi perche' nulla partira' finche' almeno uno non termina
  - Spezzare in task piu' piccoli
- I task sono schedulati dall applicazione e non dall'OS, percio' possono essere piu piccoli e veloci.
- Chi scrive la app e' sicuro di schedulare meglio dell'OS che usa thread bloccanti.

Domande x Reactor

- Qual e' l'unita' di lavoro?
- E' facile aggiungerne? Viene solo dall'esterno ? (es rete)
- E' facile spezzarlo?
- Facile assemblare il risultato?
- Facile spostare in un altro thread la parte bloccante ed elaborare comunque il risultato?

mapping HW/SW

- sistema realizzato in hardware o software?
- mapping degli oggetti: processore, memoria,Input/Output
- mapping associazioni/messaggi: connettivita’
- la difficolta’ deriva da vincoli imposti: usare uno specifico componente

mapping oggetti

- Processore:
  - richiede troppa cpu per questo processore?
  - si accelera distribuendo la computazione?
  - quanti processori/ memorie
- Memoria:
  - riusciamo a memorizzare (buffering) picchi di richieste?
- Input /Output
  - altro HW per far fronte al ritmo di produzione dati? (es stampante)
  - rapporto tra tempo di risposta e velocita' rete

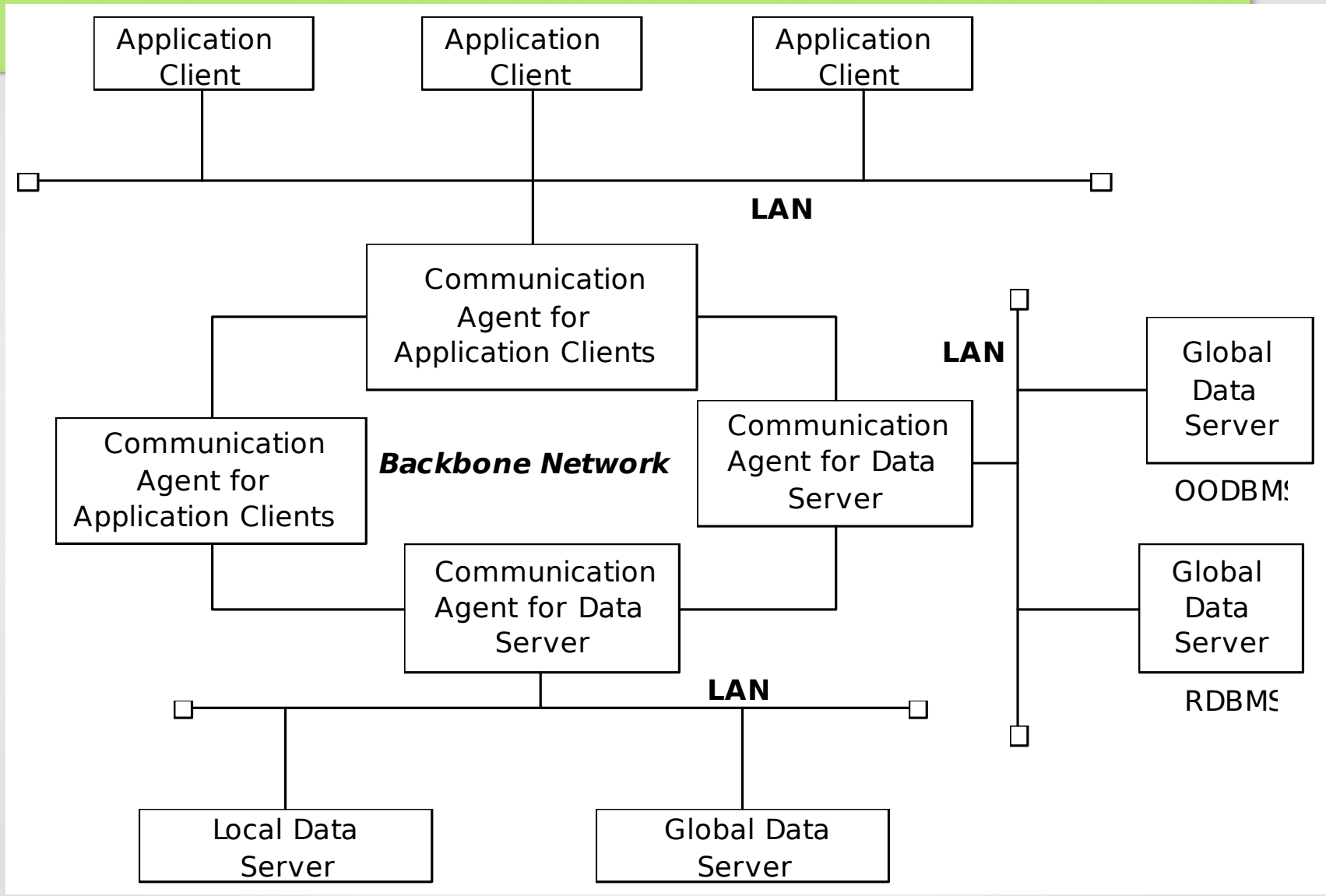
mapping: connettivita’

- Fisica:
  - quali associazioni del modello sono mappate su connessioni fisiche? (magari nessuna)
- Logica: tra sottosistemi

connettivita’ nei sistemi distribuiti

- mezzo di comunicazione? ethernet, wireless
- qualita’ di servizio? protocolli?
- interazione sincrona, asincrona?
- larghezza di banda necessaria alla trasmissione dei dati

connettivita’ nei sistemi distribuiti



mapping Hw/Sw

- topologia delle connessioni:
  - albero, stella , anello, matrice
- ci sono alcuni task da eseguire in luoghi specifici?
  - es. controllore aereo di bordo
- tempo di risposta

rappresentazione UML dei sottosistemi

- modellare struttura statica e dinamica:
  - diagramma statico dei **componenti** durante le fasi di progettazione e compilazione
  - diagramma dinamico di installazione (deployment) al momento dell'esecuzione
- esistenza in vita dei componenti
  - alcuni esistono solo durante la progettazione
  - alcuni fino alla compilazione (package)
  - altri solo all'esecuzione

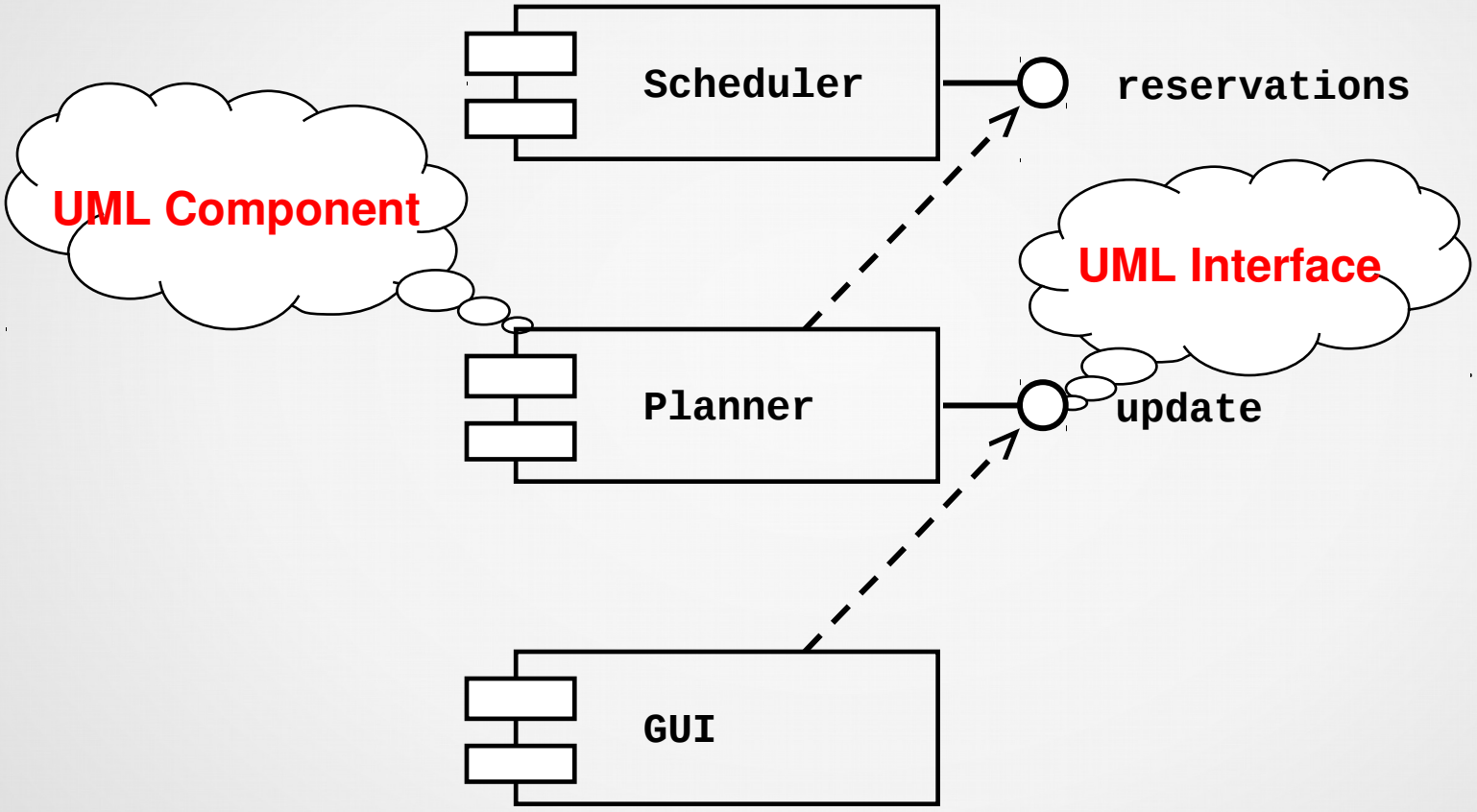
Componente

- Un pezzo fisico del sistema
- 5 tipi In UML
  - Modulo eseguibile direttamente
  - Libreria statica o dinamica
  - Tabella
  - File
  - Documento testuale

Diagramma dei Componenti

- un grafo di componenti connesso da relazioni di dipendenza
- mostra le dipendenze tra sorgenti, oggetti, eseguibili
- dipendenze come frecce tratteggiate tra utente e fornitore
- dipendenze specifiche al tipo di linguaggio

Diagramma dei Componenti (UML 1)



Caratteristiche di un Componente

- Mai installato parzialmente
- Autonomo e documentato da poter essere usato da un altro ente
- Non ha stato (indistinguibile da una copia)
- Sostituibile
- Funzione definita ed internamente coeso
- Puo’ essere annidato in altri componenti
- Pensate ai componenti di una vettura

Diagramma dei componenti (Uml 1)

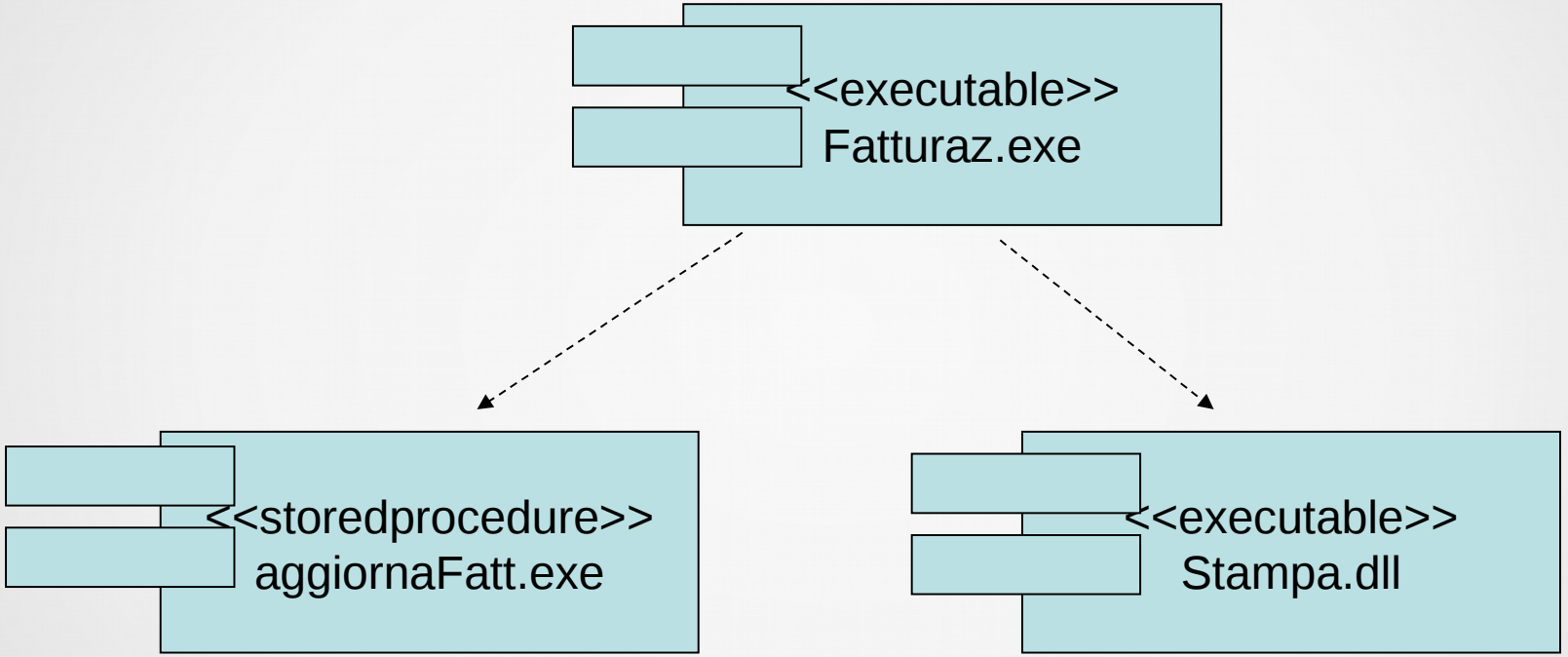
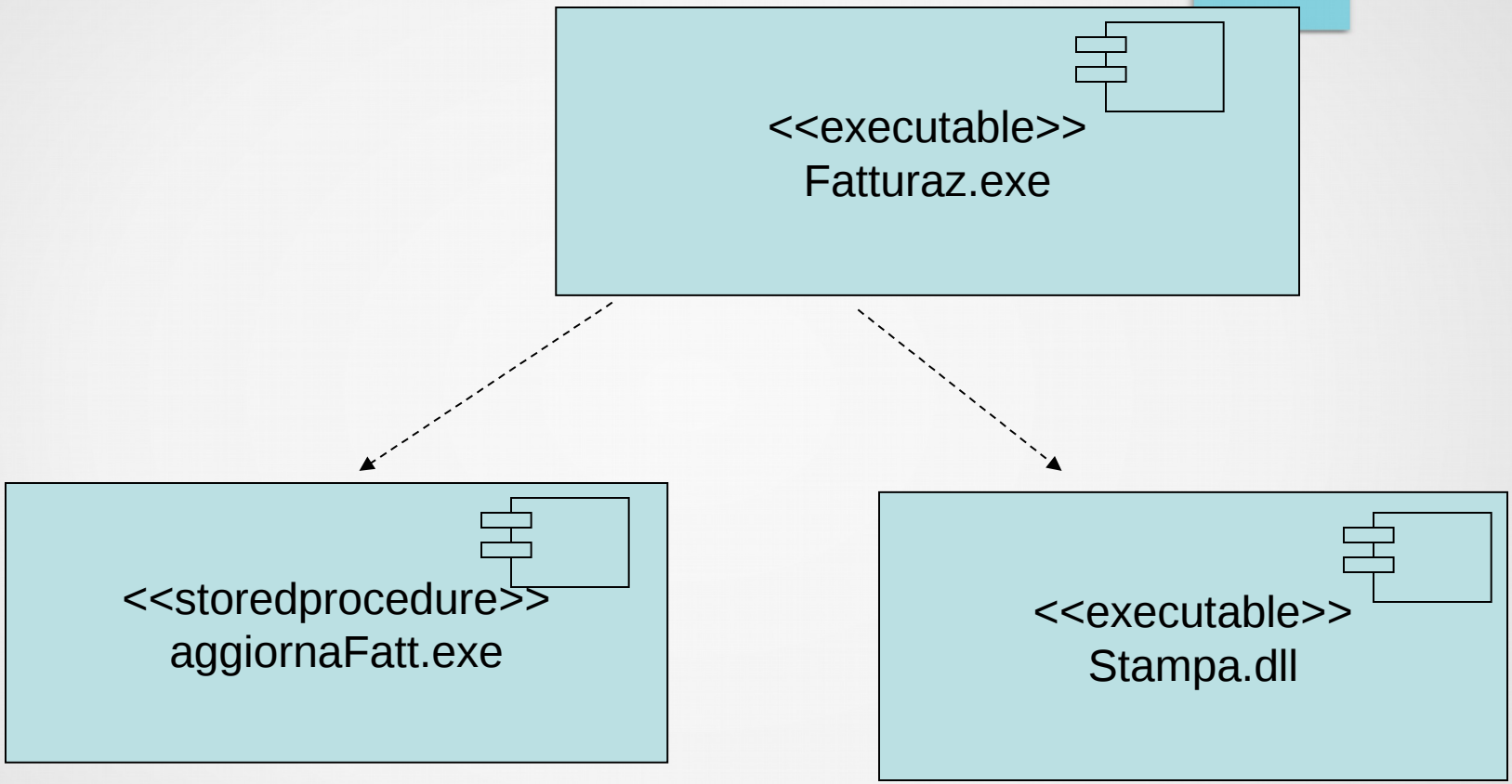


Diagramma dei componenti (UML 2)



Componente e package

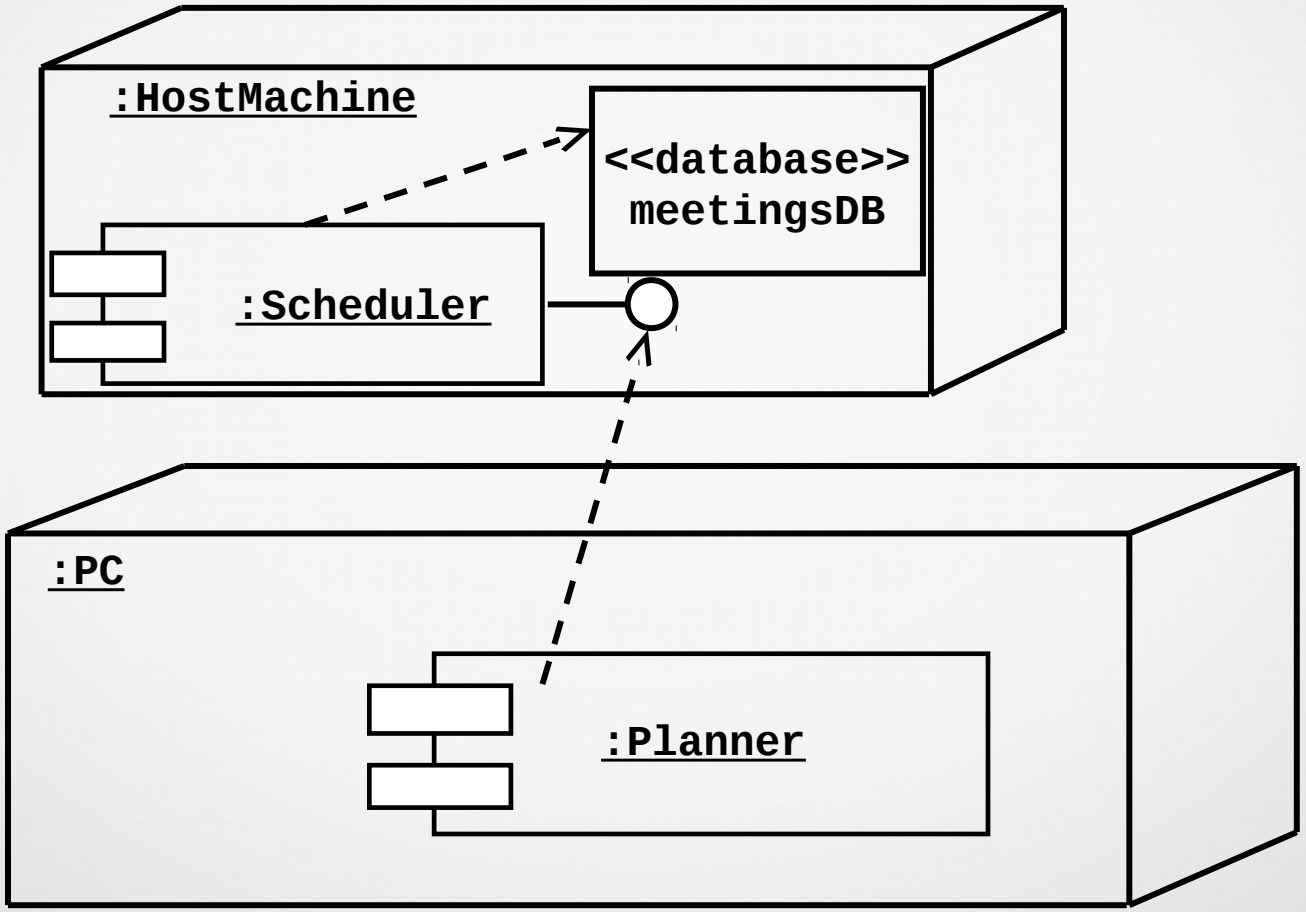
- Il package raggruppa in modo logico i componenti: aiuta a definire i componenti
- I componenti contribuiscono di solito all'esecuzione di una singola funzionalita' dell'applicazione
- In realta' anche se i componenti sono delle unita' sostituibili , molto spesso le loro interdipendenze obbligano a sostituire un intero pezzo del sistema

Diagramma di installazione(deployment)

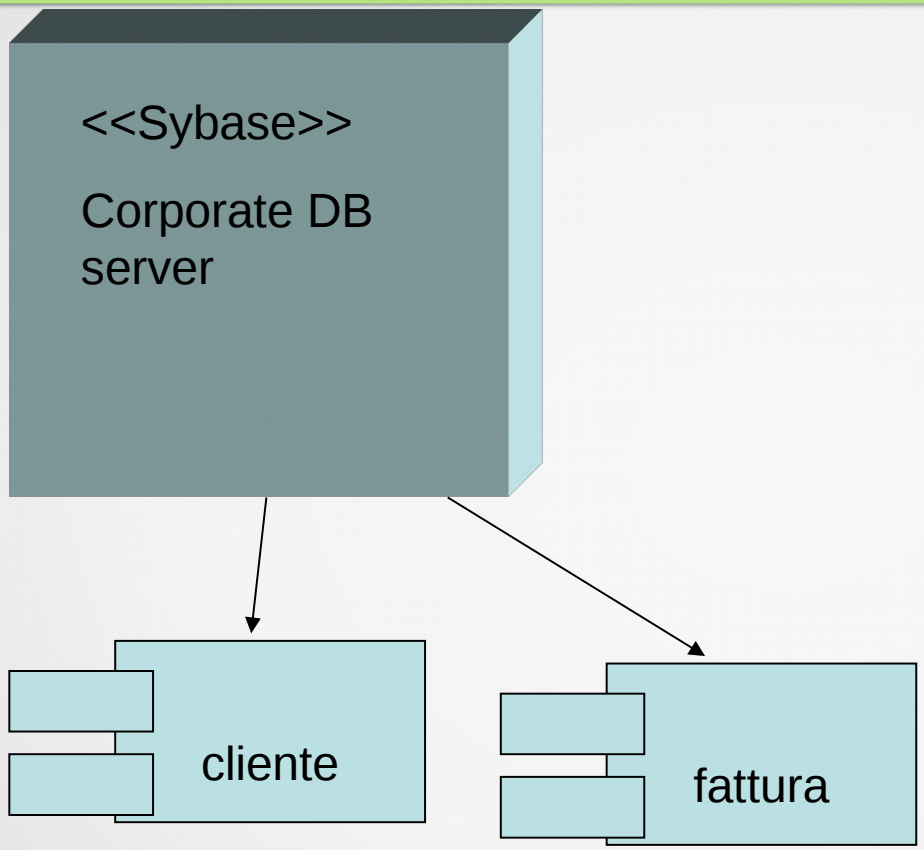
- rappresenta il sistema dopo le seguenti decisioni:
  - decomposizione in sottosistemi
  - concorrenza
  - mapping HW/SW
- I nodi, disegnati come cubi, sono delle risorse computazionali
- I nodi sono il luogo dove i componenti vengono eseguiti

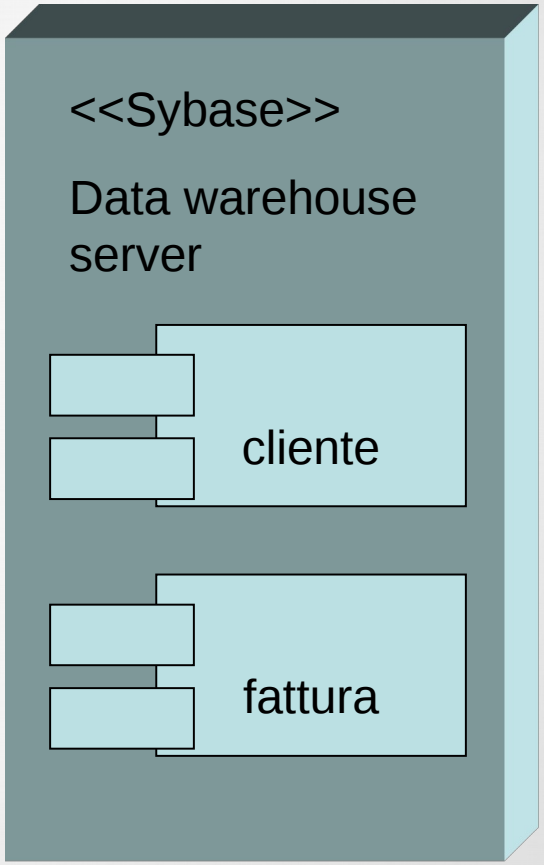
Diagramma di deployment

Diagramma di installazione



Nodi e componenti





Gestione risorse globali

- controllo dell’accesso
- differenti diritti di accesso per le varie classi di attori
- meccanismi di protezione contro accessi non autorizzati

domande su risorse globali

- occorre autenticazione per accedere al sistema? schema:
  - nome e pin, biglietto di ingresso..
- interfaccia di autenticazione
- occorre un server globale dei nomi?
- rendere noto il server al sistema (pubblicazione):
  - fase di compilazione, esecuzione?
  - per indirizzo, per nome?

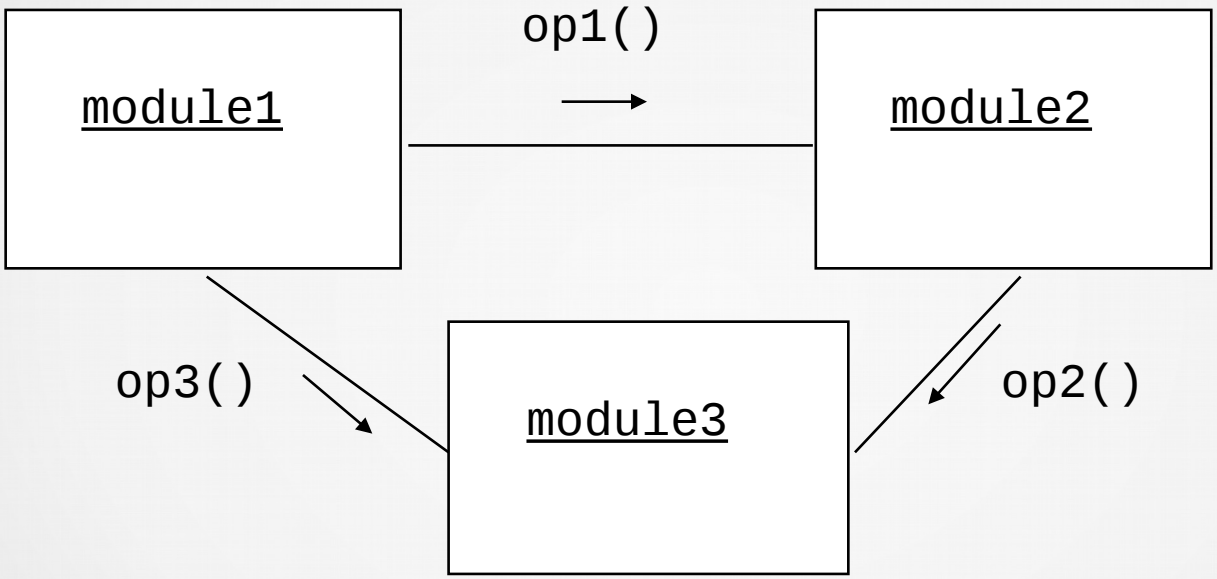
Controllo SW (1)

- implicito (non procedurale o dichiarativo)
  - sistemi a regole (intelligenza artificiale)
  - programmazione logica
- esplicito (linguaggi procedurali)
  - controllo centralizzato
    - procedurale main program chiama gli altri
    - facile da costruire

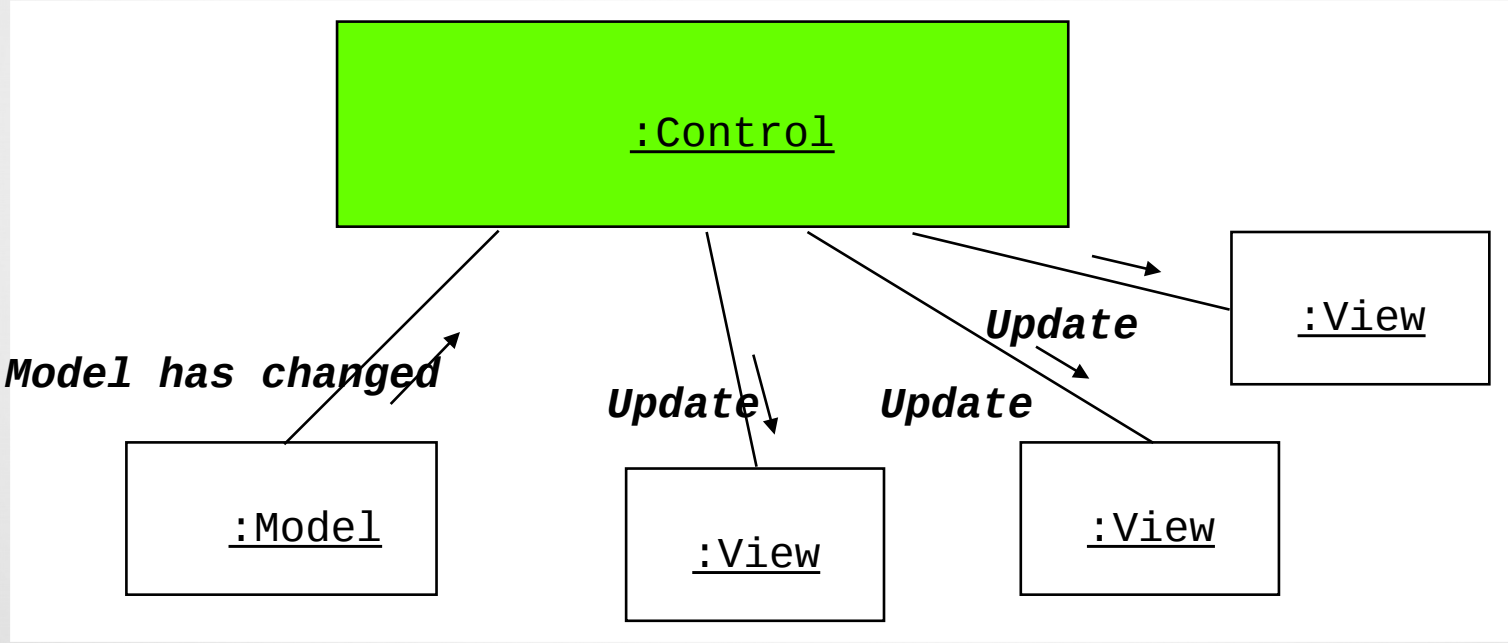
Controllo SW (2)

- ad eventi
  - ogni evento causa esecuzione di una funzione
  - utile per interfacce utente
- decentralizzato :diversi oggetti controllano il sistema
  - maggior costo comunicazione
  - sistemi ad agenti

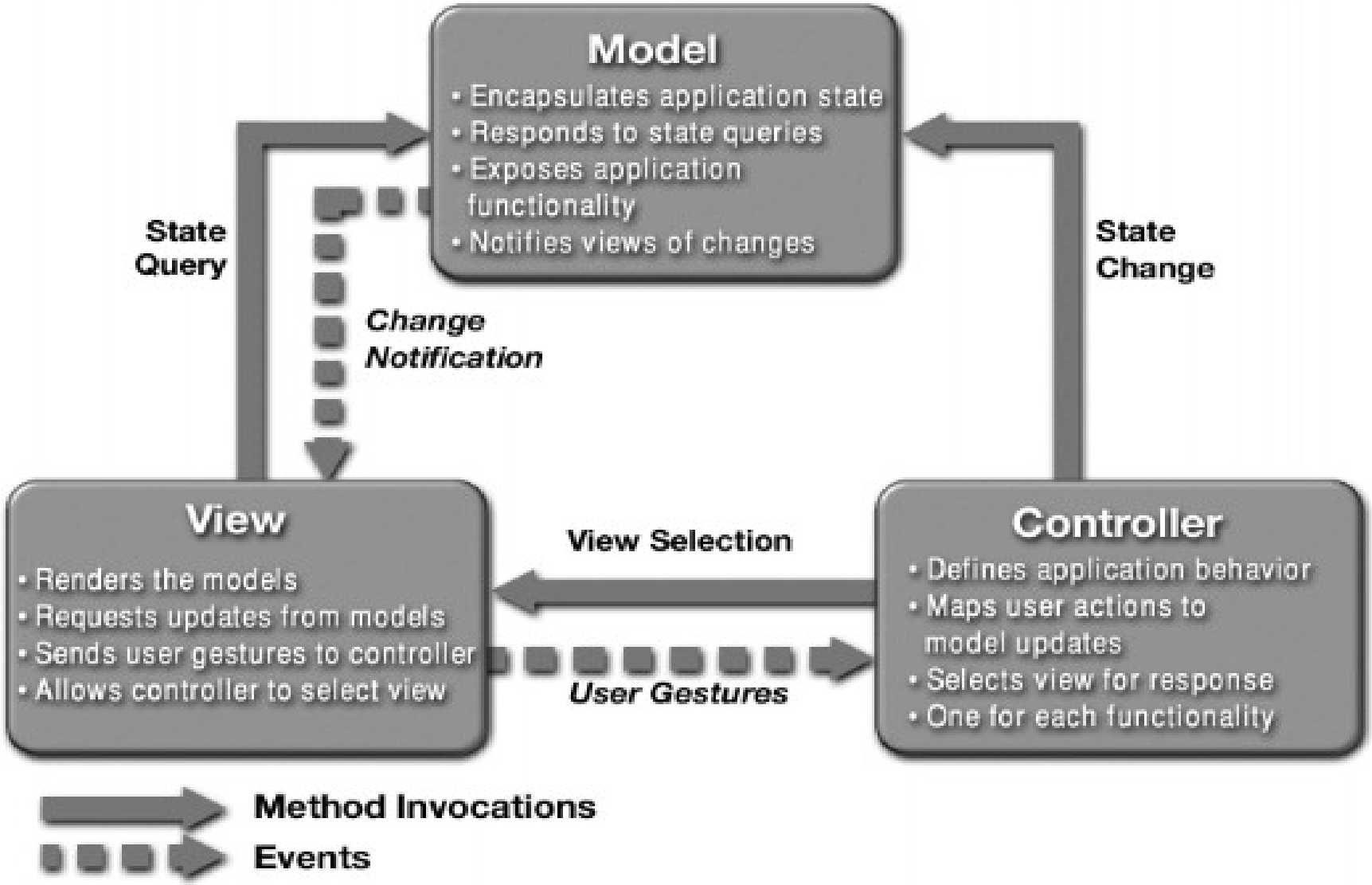
Controllo procedurale



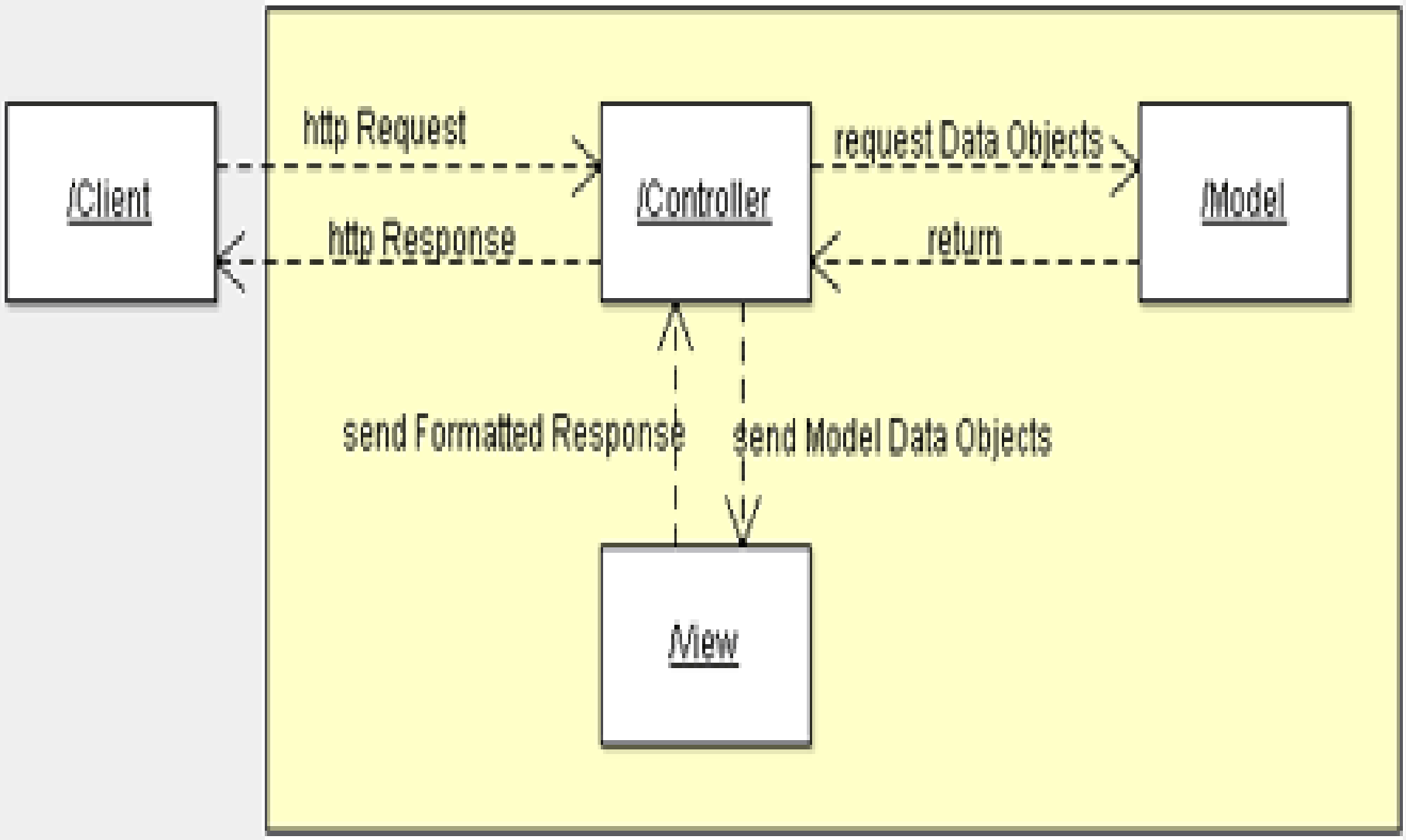
Controllo ad eventi: Smalltalk-80 (MVC)



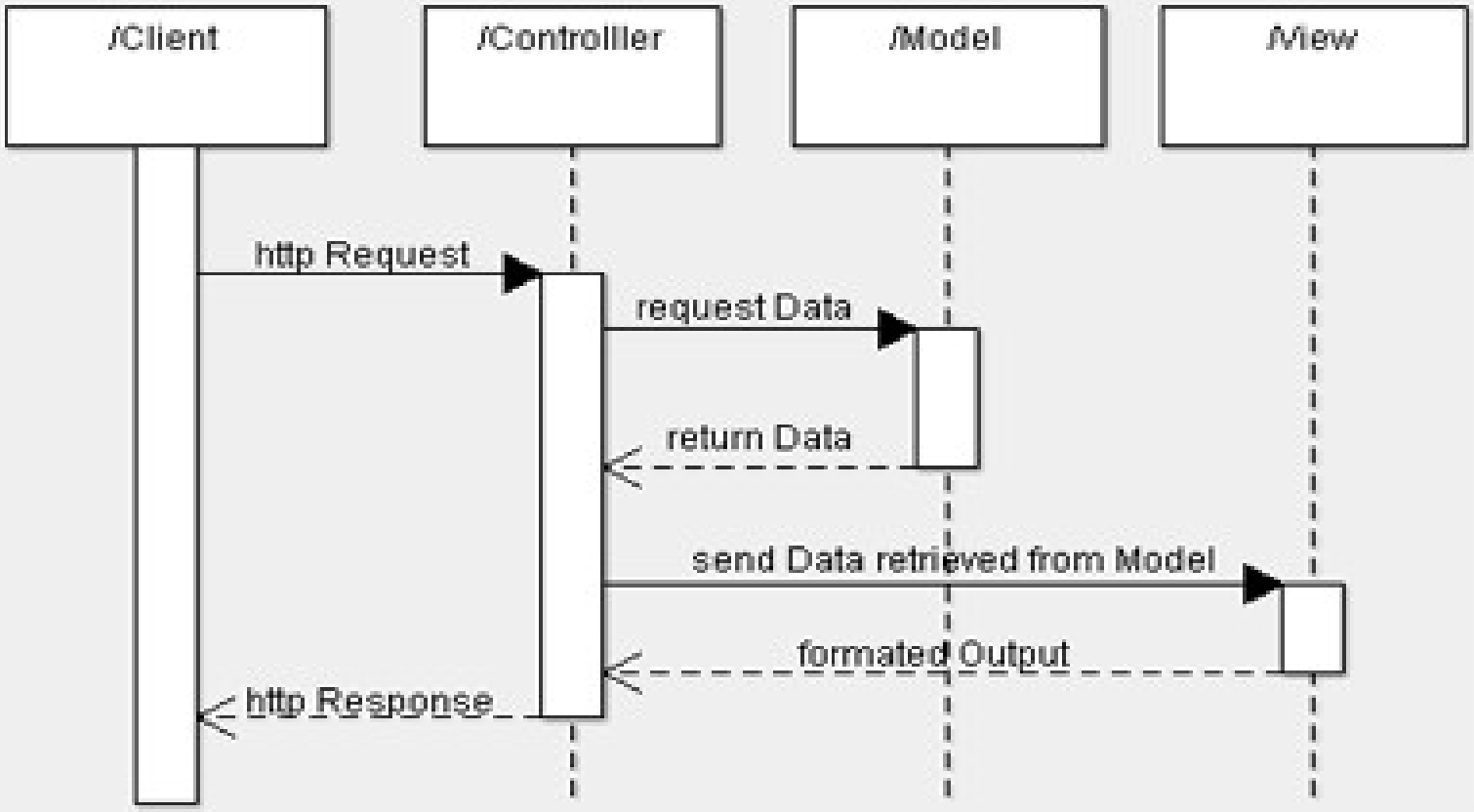
(MVC)


Model
• Encapsulates application state
• Responds to state queries
• Exposes application functionality
• Notifies views of changes
State Query
State Change
Change Notification
View
• Renders the models
• Requests updates from models
• Sends user gestures to controller
• Allows controller to select view
View Selection
User Gestures
Controller
• Defines application behavior
• Maps user actions to model updates
• Selects view for response
• One for each functionality
Method Invocations
Events

(MVC) http e browser



(MVC) http e browser



Controllo centralizzato o no?

- centralizzato
  - un oggetto controlla tutto
  - facile modifica della struttura
  - possibile congestione nel traffico
- decentralizzato
  - responsabilita’ distribuita
  - piu’ complicato
  - adatto ai sistemi ad oggetti

Condizioni al contorno

- importante considerare le condizioni di inizio e termine del funzionamento del sistema:
- casi d’uso di partenza (startup)
- liberazione delle risorse alla terminazione
- Indisponibilita’ del sistema
  - errori interni, stabili, intermittenti, imprevedibili
  - errori esterni (mancanza di corrente..)

Quesiti per condizioni di contorno

- accensione: come si inizializza il sistema?
  - a che dati accedere
  - che servizi registrare
  - che interfaccia mostrare all'utente
- termine
  - possono i sottosistemi terminare indipendentemente?, notificano gli altri ?
  - come comunicare le modifiche locali
  - ordine di spegnimento
- Malfunzionamenti
  - che fare? ci sono soluzioni di backup?
  - dopo un malfunzionamento, si riparte come in una normale accensione?

esercizio

- con che diagramma/scatola modellare i seguenti elementi in UML?
  - Mario Rossi
  - un insieme di operazioni per verificare le date
  - un modulo che le realizza
  - un insieme di impiegati
  - una richiesta di “prelievo” al bancomat
  - il server su cui gira il sistema di contabilita’

Riassumendo

- identificare la concorrenza
- mappare HW/SW
- risorse globali
- controllo SW
- condizioni al contorno

ognuna di queste attivita’ riesamina aspetti diversi del sistema,; al loro termine le interfacce possono essere definite