Sperimentazioni di Ingegneria del Software

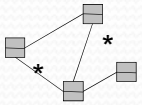
- Architettura logica e organizzazione in layers
- Introduzione al modello di progetto

Grazie a Simona Bernardi (Universita` di Saragoza, Spagna), che ha messo a disposizione il suo materiale

Verso la progettazione: elaborati

Modellazione del business

Modello di dominio



Requisiti

Modello dei casi d'uso

Specifiche supplementari

Visione

Glossario

Modello di progetto

La architettura logica dipende dai vincoli e requisiti non funzionali catturati nelle specifiche supplementari

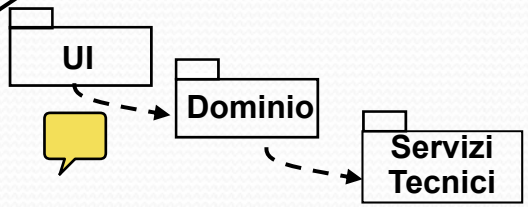


Diagramma de package dell'architettura logica (vista statica)

Progettazione

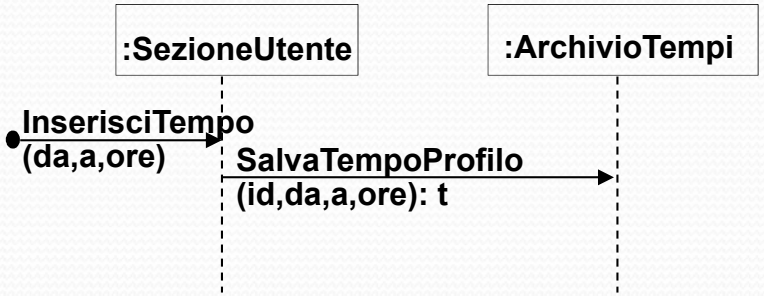


Diagramma di interazione (vista dinamica)

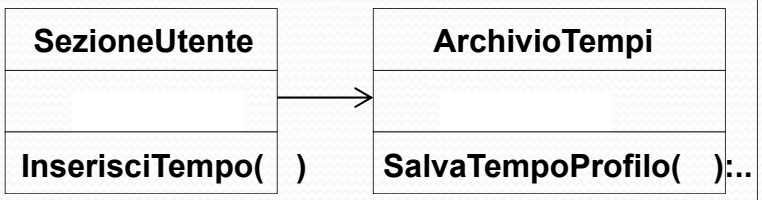
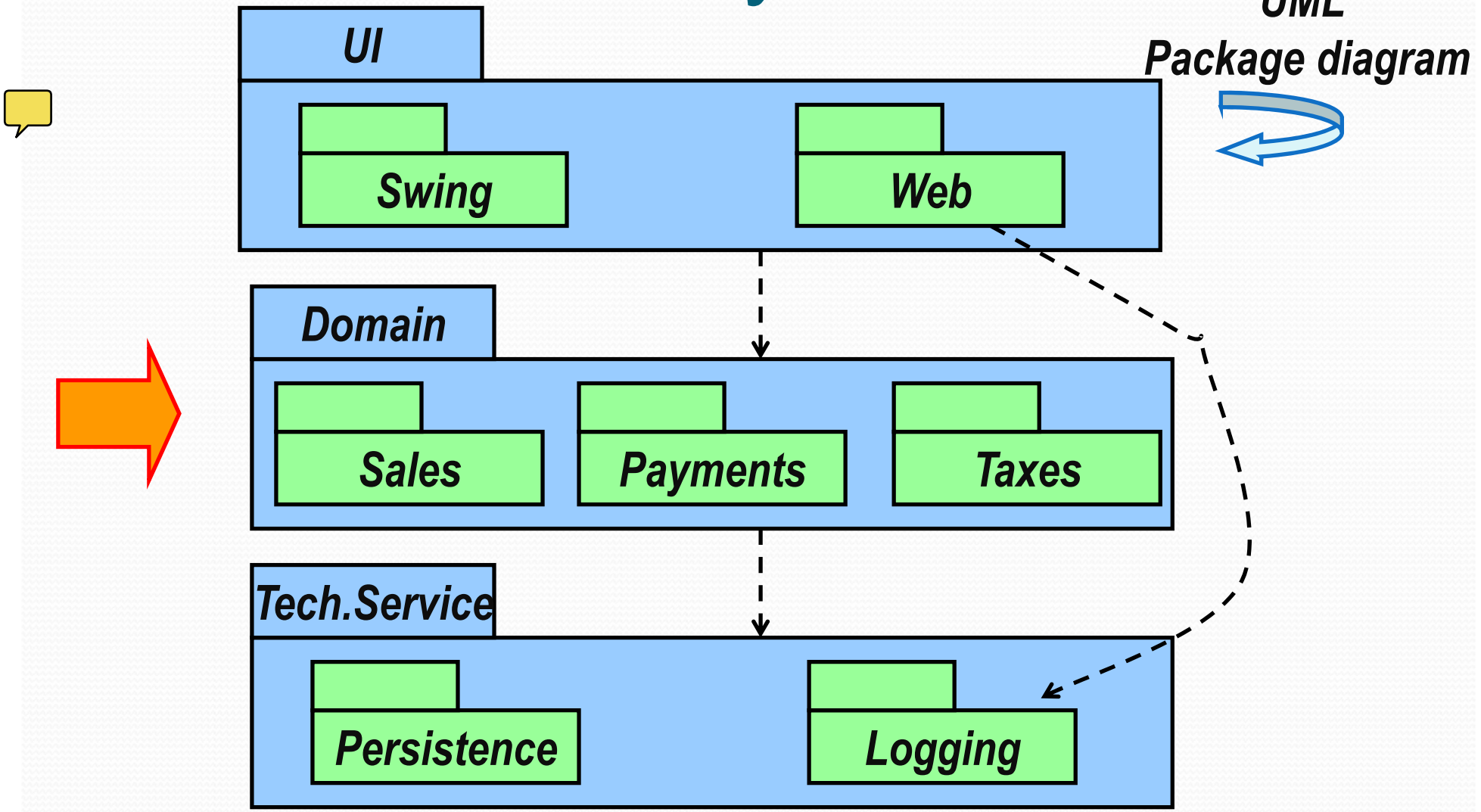


Diagramma di classe (vista statica)

Architettura logica e layers

- Macro-organizzazione delle classi software in packages (o namespaces), sottosistemi e layers
- Logica: non ci sono decisioni sullo sviluppo degli elementi software su nodi fisici (parte dell´architettura di sviluppo)
  - “Platform independent architecture”
- Layer: gruppo di classi sw, packages, sottosistemi con responsabilità condivisa su un aspetto importante del sistema
- Layers tipici
  - Interfaccia utente
  - Logica dell’applicazione e oggetti di dominio
  - Servizi tecnici

Architettura a layer di PoS



Mapping package→ codice

// ---UI Layer

com.mycompany.nextgen.ui.swing

com.mycompany.nextgen.ui.web

//---Domain Layer

com.mycompany.nextgen.domain.sales

com.mycompany.nextgen.domain.payments

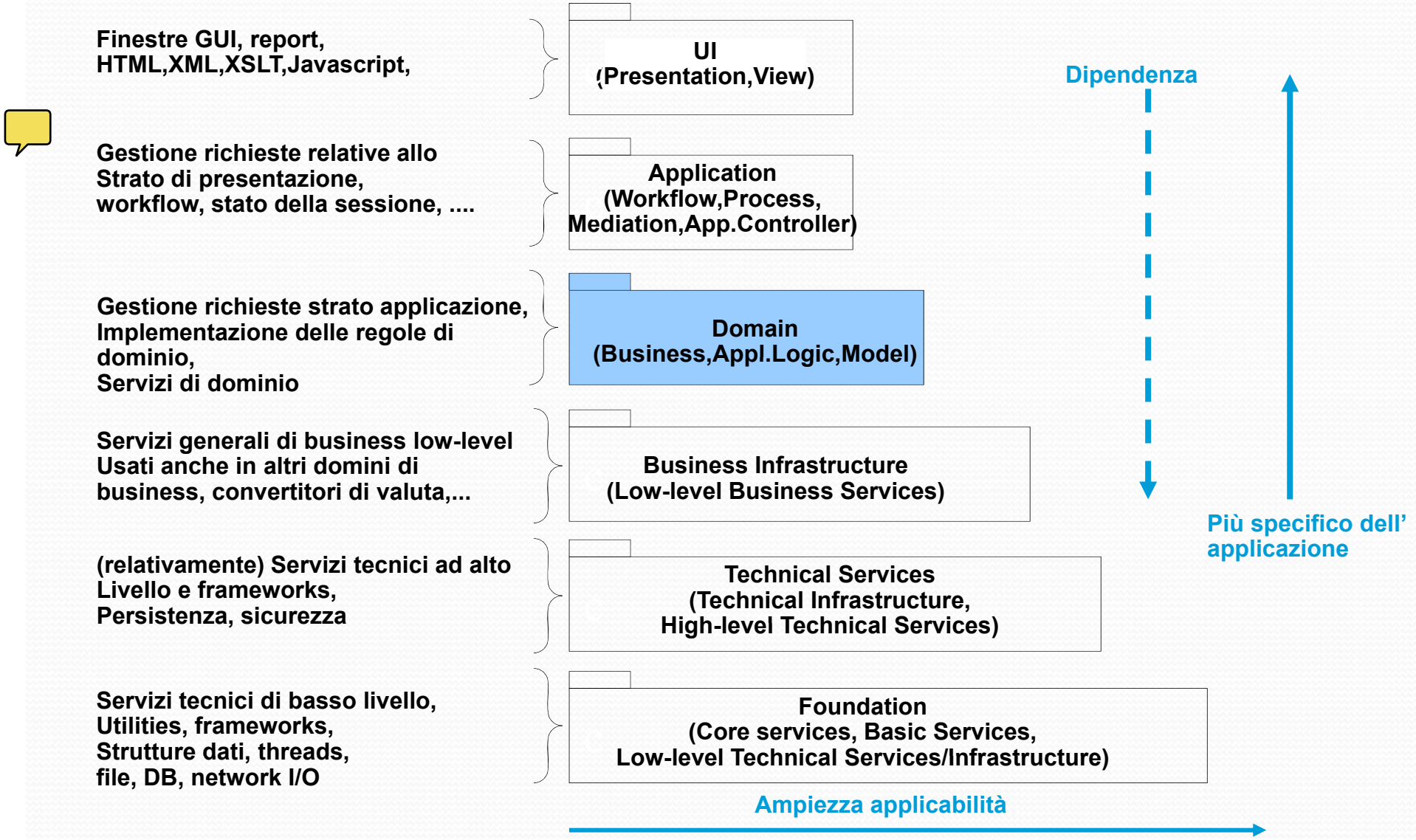
com.mycompany.nextgen.domain.taxes

//--- Tecnical services

com.mycompany.service.persistence

.....

Layers pattern

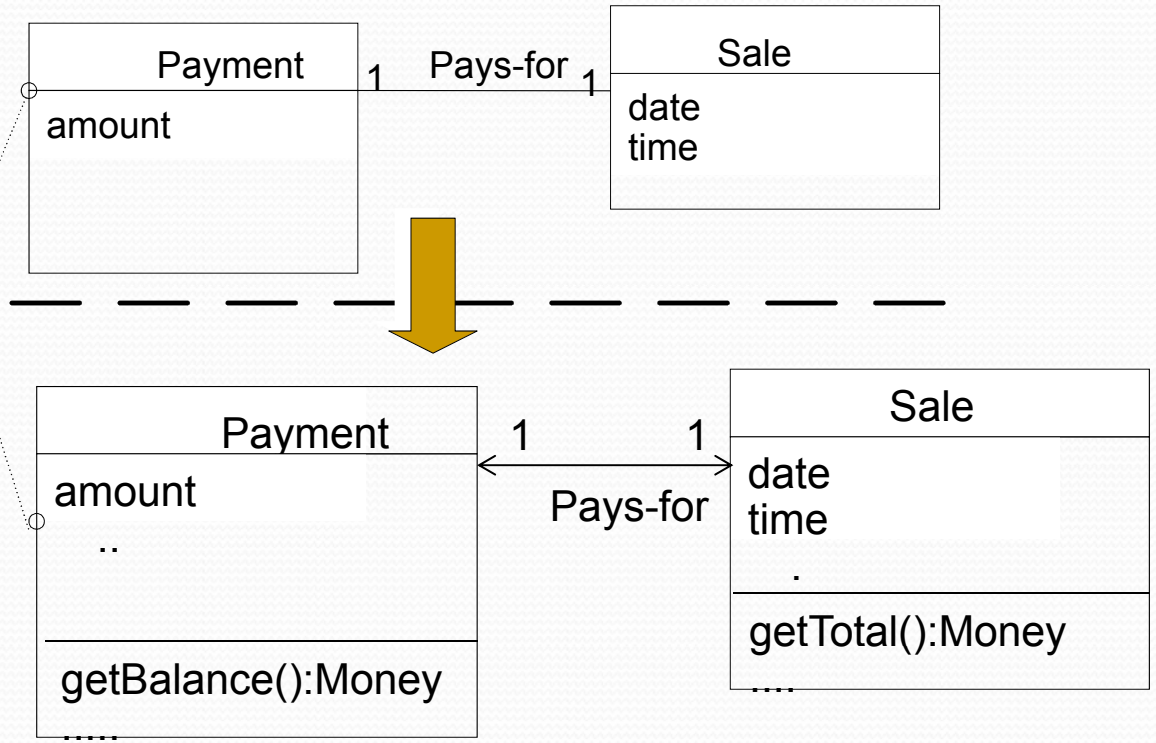


Relazione tra lo strato di dominio ed il modello di dominio

Modello di dominio UP

Rappresentazione concetti del dominio

Un Payment nel modello di dominio è un concetto, un Payment nel modello di progetto è una classe sw. Non sono la stessa cosa, ma il primo ha ispirato il nome e def.del secondo.

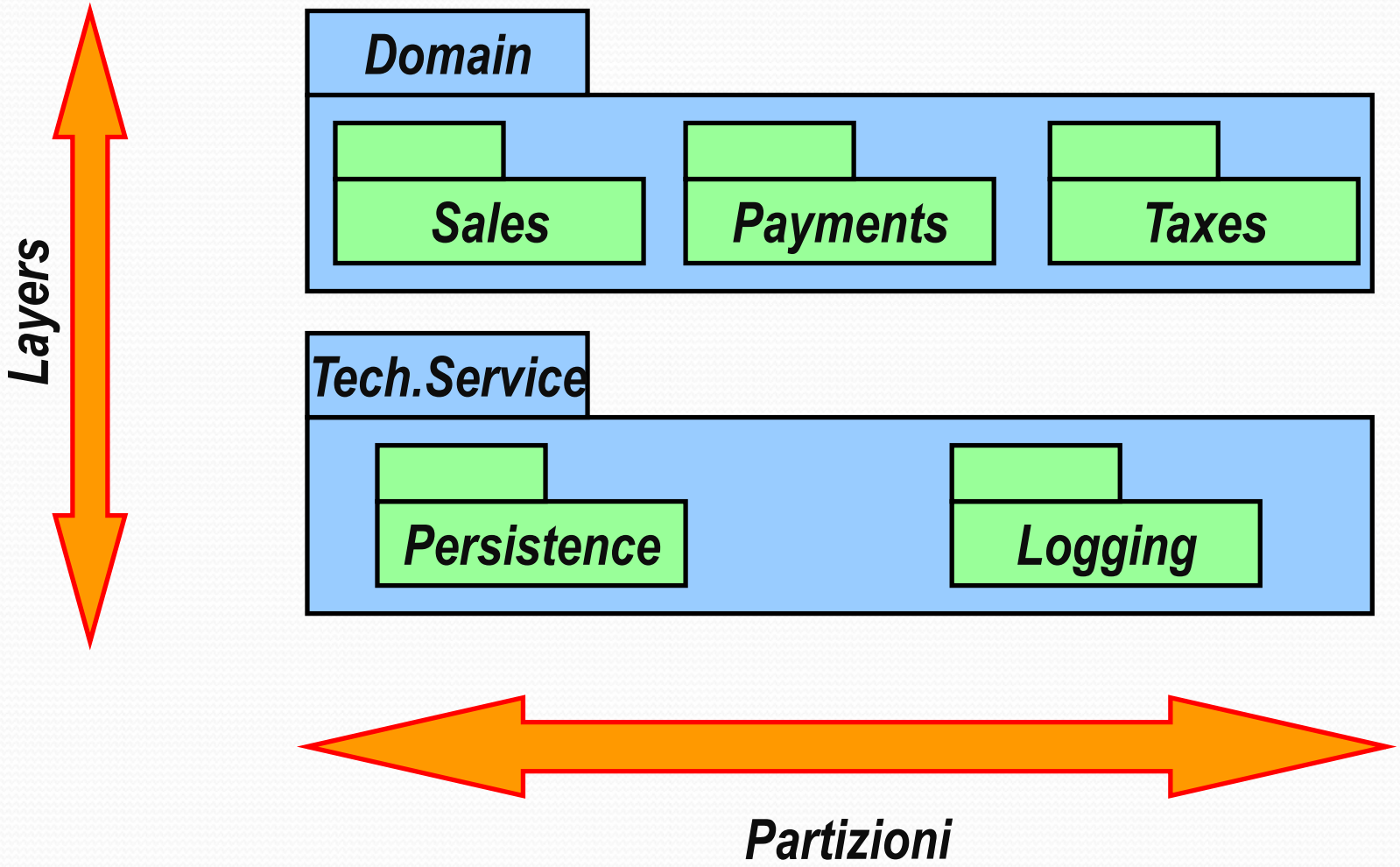


Strato di dominio dell'architettura sw nel modello di progetto UP

L'analista si ispira agli oggetti del modello di dominio per creare

Le classi software nel modello di progetto

Layers e partizioni

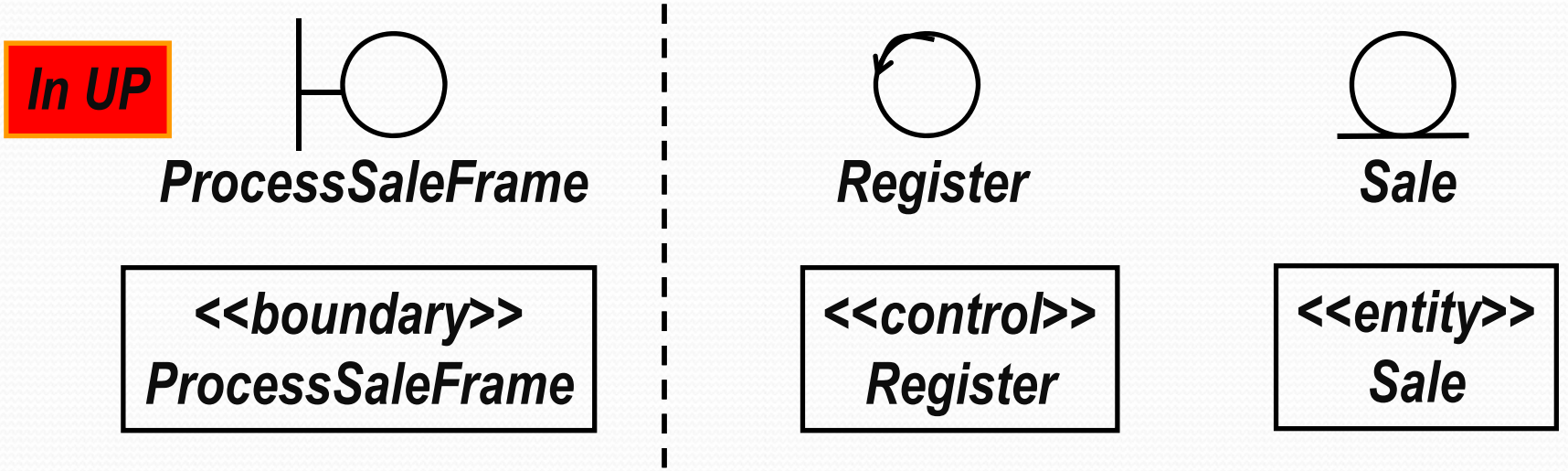


Principio di separazione modello/vista - I

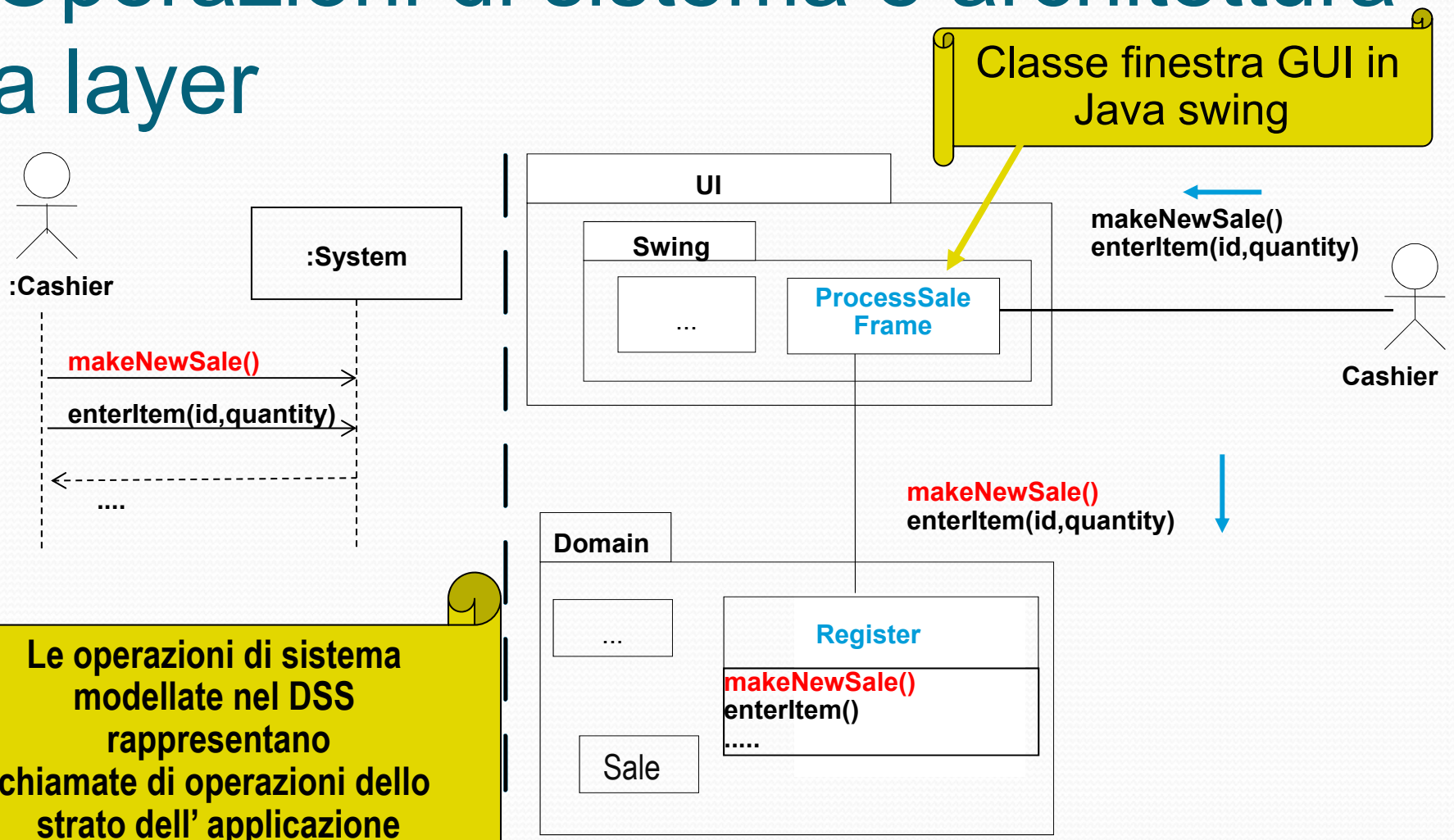
- Non relazionare o accoppiare oggetti non UI con oggetti UI
  - Le finestre appartengono ad una applicazione in particolare, mentre gli oggetti non UI possono venire riutilizzati in nuove applicazioni od essere relazionati a nuove interfacce
- Non incapsulare la logica dell' applicazione in metodi di oggetti UI
  - Gli oggetti UI inizializzano elementi UI, ricevono eventi UI e delegano le richieste della logica dell'applicazione agli oggetti non UI (oggetti di dominio)
- Modello=Strato di dominio, Vista= Strato UI

Principio di separazione modello/vista - II

- E' il principio di base del pattern Model-View-Controller (Smalltalk-80)
- A livello dell'architettura sw, il pattern MVC relaziona gli oggetti di dominio (model), gli oggetti GUI (view) e gli oggetti che gestiscono gli eventi (controller)



Operazioni di sistema e architettura a layer



Le operazioni di sistema modellate nel DSS rappresentano chiamate di operazioni dello strato dell' applicazione o dominio attraverso lo strato UI

Principio di separazione modello/vista - III

- Permette definire modelli con coesione, focalizzandosi nei processi di dominio piuttosto che sull'interfaccia utente
- Permette separare lo sviluppo dello strato di dominio e dello strato UI
- Minimizza l'impatto del cambio di requisiti (nel dominio) nello strato UI
- Permette di collegare nuove viste allo stesso strato di dominio
- Permette di avere viste concorrenti e multiple dello stesso modello di dominio
- Permette di eseguire lo strato di dominio in maniera indipendente dallo strato UI

Sperimentazioni di Ingegneria del Software

- Architettura logica e organizzazione in layers
- Introduzione al modello di progetto

Modellazione agile in progettazione

- Modellare per capire e non per documentare
- Modellazione in gruppo (con altri)
- Creare viste diverse in parallelo
  - Dinamiche
  - Statiche

Modelli dinamici/statici

- Modelli dinamici (es., diagrammi di interazione UML)
  - Rappresentano il comportamento del sistema, la collaborazione tra oggetti sw per realizzare (uno/più scenari di) un caso d'uso, i metodi di classi sw
- Modelli statici (es., diagrammi di classe UML)
  - Servono per definire i package, i nomi delle classi, gli attributi, le firme di operazioni
- Sono tra loro relazionati
  - Per tale ragione si consiglia di crearli in parallelo

Relazioni tra sequence e class diagram