



Luca Cabibbo  
Architettura  
dei Sistemi  
Software

# Introduzione ai pattern architeturali

**dispensa asw320**  
ottobre 2023

*If the design, or some central part of it,  
does not map to the domain model,  
that model is of little value, and  
the correctness of the software is suspect.*

*Eric Evans*



## - Riferimenti

- ❑ Luca Cabibbo. **Architettura del Software: Strutture e Qualità**. Edizioni Efestò, 2021.
  - Capitolo 16, **Introduzione ai pattern architeturali**
- ❑ [POSA1] Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., and Stal, M. **Pattern-Oriented Software Architecture (Volume 1): A System of Patterns**. Wiley, 1996.
- ❑ [POSA4] Buschmann, F., Henney, K., and Schmidt, D.C. **Pattern-Oriented Software Architecture (Volume 4): A Pattern Language for Distributed Computing**. Wiley, 2007.
- ❑ [DDD] Evans, E. **Domain-Driven Design: Tackling Complexity in the Heart of Software**. Addison-Wesley, 2004.
- ❑ Larman, C. **Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development**. Prentice Hall, 2004.



## - Obiettivi e argomenti

### □ Obiettivi

- introdurre i pattern architetturali
- presentare i pattern architetturali POSA fondamentali più astratti

### □ Argomenti

- introduzione
- decomposizioni tecniche e di dominio
- Domain Model (POSA4)
- Domain Object (POSA4)
- discussione



## \* Introduzione

### □ Un *pattern software*

- la descrizione strutturata di una soluzione esemplare a un problema (software) ricorrente

### □ Un *pattern architetturale* (o *stile architetturale*)

- esprime uno schema per l'organizzazione strutturale fondamentale per i sistemi software
- guida l'organizzazione dell'architettura di un sistema software (o di un suo componente complesso)

### □ Un *pattern language* – *linguaggio di pattern*

- una famiglia di pattern correlati
- comprende anche una discussione delle relazioni tra pattern



## POSA1

- **Pattern-Oriented Software Architecture (Volume 1) [POSA1]**, pubblicato nel 1996, è il primo libro a presentare in modo sistematico un insieme di pattern architetturali
  - dal fango alla struttura – ad es., layers, pipes-and-filters, ...
    - per sostenere una decomposizione controllata del sistema complessivo
  - sistemi distribuiti – ad es., broker, ...
    - per fornire un'infrastruttura per applicazioni distribuite
  - sistemi interattivi – ad es., model-view-controller, ...
    - per strutturare sistemi software che prevedono un'interazione uomo-macchina
  - sistemi adattabili – ad es., reflection, microkernel, ...
    - per sostenere l'adattamento del sistema e la sua evoluzione

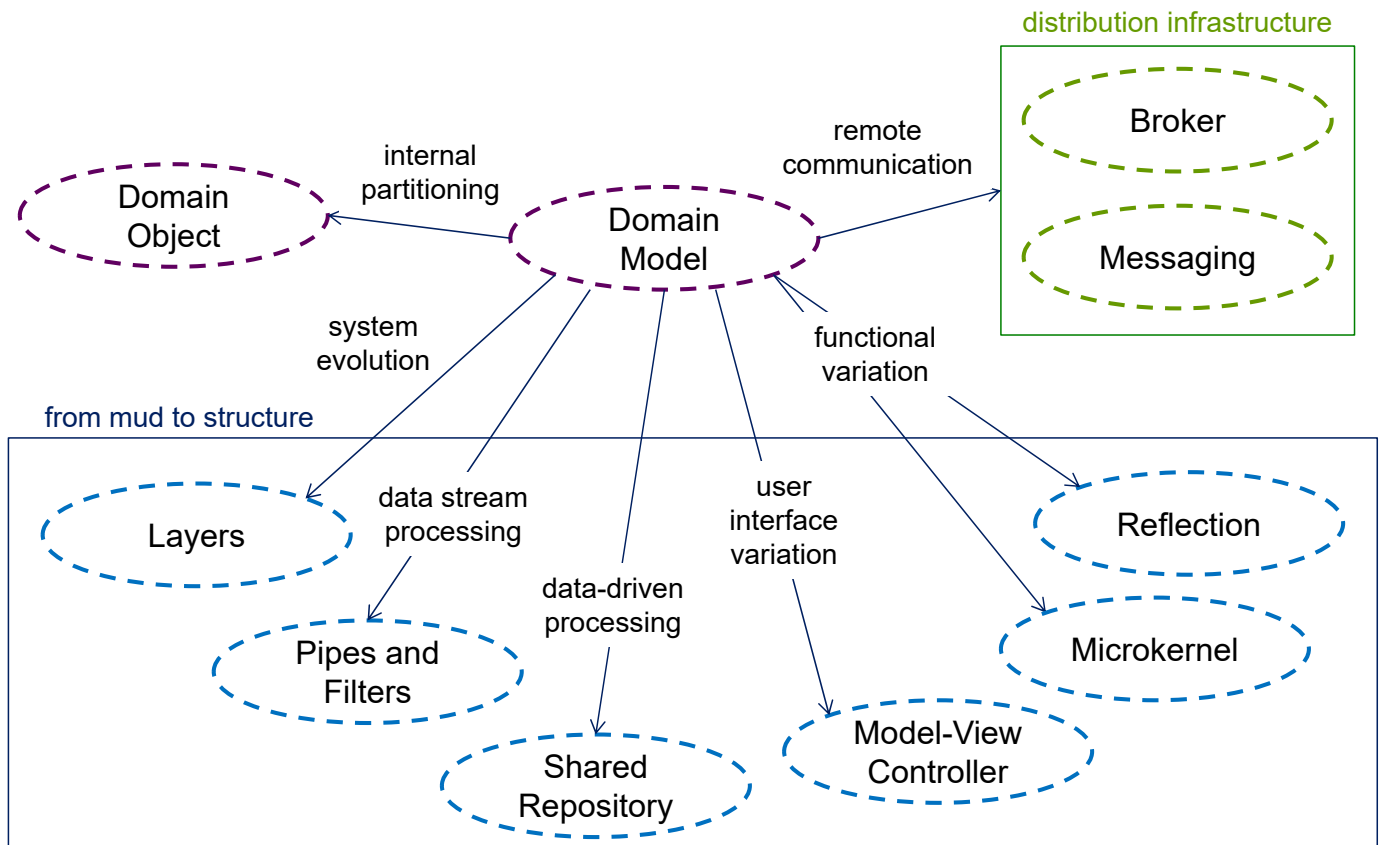


## POSA4

- **Pattern-Oriented Software Architecture (Volume 4) [POSA4]**, pubblicato nel 2007
  - definisce un linguaggio di pattern per sistemi distribuiti
  - rivisita e correla numerosi pattern architetturali definiti in precedenza in questo ambito
    - dai volumi precedenti della serie POSA
    - da **Patterns of Enterprise Application Architecture** di Martin Fowler
    - da **Enterprise Integration Patterns** di Hohpe e Woolf
    - ...
- Esistono anche altri pattern e altri cataloghi



# I principali pattern architetturali POA



7

Introduzione ai pattern architetturali

Luca Cabibbo ASW



## Premessa: l'antipattern Big Ball of Mud

- **Big Ball of Mud** ("grossa palla di fango") è il principale **antipattern** architetturale
  - una "Grossa Palla di Fango" è una giungla di "spaghetti code", strutturata a caso, scomposta, sciatta, che si tiene insieme con nastro isolante e filo da imballo
  - questi sistemi mostrano segni inequivocabili di crescita irregolare e di riparazioni ripetute e basate su espedienti
  - le informazioni sono condivise in modo promiscuo tra elementi distanti del sistema, spesso ad un punto tale che quasi tutte le informazioni importanti sono globali o duplicate
  - la struttura complessiva del sistema potrebbe non essere mai stata ben definita – se lo fosse, essa potrebbe essere corrosa e non più riconoscibile
  - i programmatori con un briciolo di sensibilità architetturale fuggono da questi acquitrini: solo coloro che sono indifferenti all'architettura e, forse, si sentono a proprio agio con l'inerzia del compito quotidiano di rattoppare i buchi in queste dighe fallite, si accontentano di lavorare su tali sistemi

8

Introduzione ai pattern architetturali

Luca Cabibbo ASW



# Benefici nell'uso dei pattern architetturali

- Possibili usi dei pattern architetturali
  - soluzione di progetto per il sistema in discussione
  - base per l'adattamento o per un nuovo pattern architetturale
  - ispirazione per una soluzione correlata
- Ogni sistema è basato su un pattern architetturale dominante
  - un sistema o una struttura potrebbe usare anche degli ulteriori pattern architetturali "secondari"
- Benefici nel basare un'architettura su un pattern riconoscibile
  - selezione di una soluzione provata e ben compresa
  - più facile comprendere l'architettura e le sue caratteristiche



## - Osservazione sulla notazione

- I pattern architetturali propongono dei criteri di decomposizione di un sistema in elementi architetturali
  - questi elementi possono essere mostrati usando un linguaggio di modellazione a oggetti – ad es., UML
    - ogni elemento ha un nome o riferimento, un'interfaccia pubblica e un'implementazione privata
    - le interazioni tra elementi possono essere mostrate mediante lo scambio di messaggi
  - una notazione a oggetti è spesso adeguata – ma va interpretata in modo "architetturale"



## - Pattern e tattiche architeturali

- Un pattern architeturale è un “pacchetto di decisioni di progetto” [SAP]
  - talvolta è utile analizzare un pattern architeturale per comprendere le tattiche che applica e le qualità a cui si riferiscono
  - è anche importante comprendere come realizzare delle opportune *strategie architeturali* – basate sull'applicazione congiunta di pattern e tattiche architeturali



## - Portata di un pattern architeturale

- La *portata (scope)* di un pattern architeturale è la dimensione complessiva massima sostenuta da quel pattern
  - ad es., un sottosistema o un'intera applicazione software, un'applicazione per uso personale o un'applicazione web, una famiglia di prodotti software o l'insieme di tutti i sistemi software utilizzati da un'organizzazione

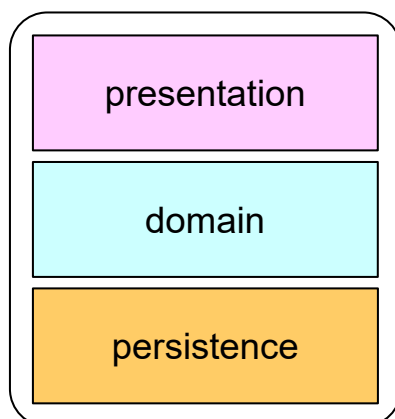


## \* Decomposizioni tecniche e di dominio

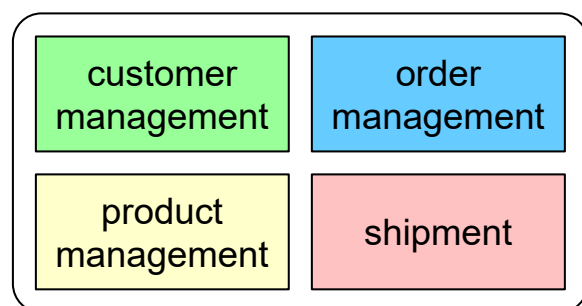
- ❑ Molti pattern architetturali hanno lo scopo di decomporre un sistema software in un insieme di elementi architetturali
  - questa decomposizione può essere
    - una *decomposizione tecnica*
      - fa riferimento a capacità e aspetti tecnici
      - ciascun elemento rappresenta un interesse tecnico distinto
    - una *decomposizione di dominio*
      - fa riferimento al dominio e a capacità di business
      - gli elementi rappresentano attività o compiti che l'organizzazione a cui è destinato il sistema software svolge (con lo scopo di generare valore di business)



## Decomposizioni tecniche e di dominio



technical partitioning

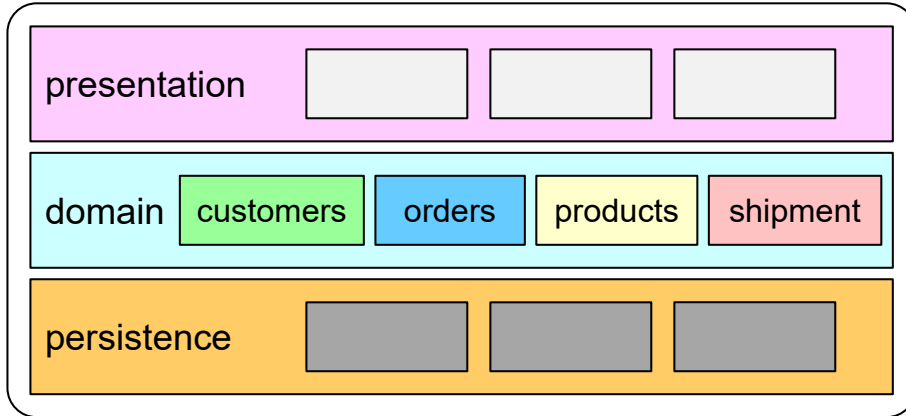


domain partitioning



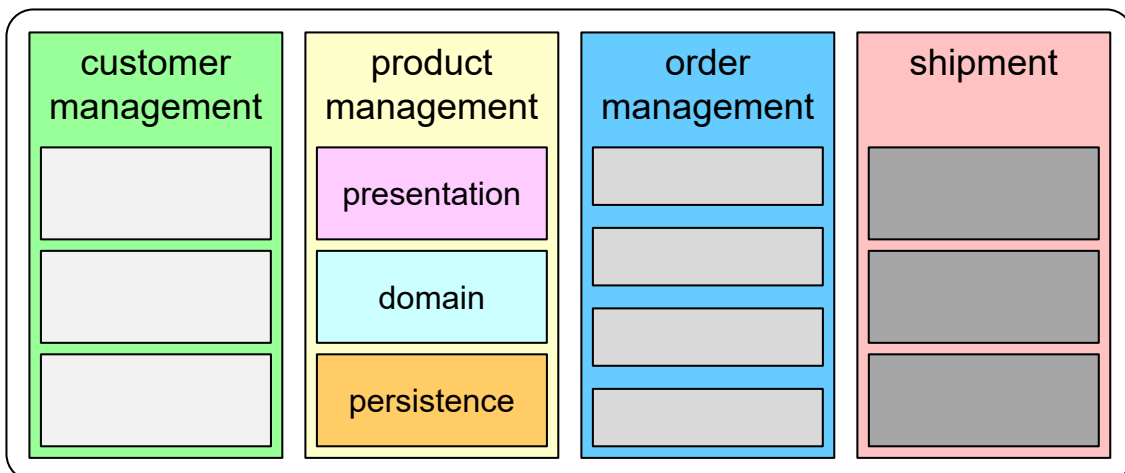
# Decomposizioni multiple

- In molti sistemi sono utili più decomposizioni successive



# Decomposizioni multiple

- In molti sistemi sono utili più decomposizioni successive







## Decomposizione di primo livello

- Nell'architettura di un sistema software, la *decomposizione di primo livello* del sistema è spesso particolarmente rilevante
  - perché spesso descrive come gli elementi software
    - vengono rilasciati come processi ai nodi ed eseguiti a runtime
    - vengono assegnati ai team di sviluppo



## \* Domain Model (POSA4)

- Il pattern architetturale **Domain Model** [POSA4]
  - il più astratto – la “radice” della gerarchia dei pattern architetturali POSA
  - fornisce una chiave di lettura comune per gli altri pattern architetturali



# Domain Model

## □ Contesto

- l'inizio della progettazione di un sistema software
- è necessaria una struttura iniziale per il software

## □ Problema

- i requisiti identificati descrivono le funzionalità e le qualità desiderate per il sistema da sviluppare – ma non sono direttamente utili per guidare il suo sviluppo
- è necessaria un'"intuizione" precisa e ragionata del dominio applicativo del sistema – per evitare che questo diventi "una grossa palla di fango"



# Domain Model

## □ Soluzione

- crea, usando un metodo appropriato, un modello di dominio (*domain model*) che definisce e limita le responsabilità di business del sistema
  - gli elementi nel modello sono astrazioni significative nel dominio applicativo
  - i loro ruoli e le loro interazioni riflettono il flusso di lavoro nel dominio
- usa il modello di dominio come fondamenta per l'architettura software del sistema



## Domain Model

- Un *modello di dominio* è una qualche rappresentazione del dominio di interesse per il sistema
  - inoltre, il “modello di dominio” va creato usando “un metodo appropriato”
    - il “tipo” di modello di dominio da creare e il “metodo appropriato” dipendono fortemente dalle caratteristiche del sistema in discussione
  - si veda anche il pattern Domain Object



## Domain Model

- Alcune possibili interpretazioni concrete per “modello di dominio”
  - un modello delle funzionalità del sistema
  - una decomposizione delle informazioni del dominio
  - una decomposizione comportamentale
  - talvolta, una decomposizione tecnica
  - ...



## Domain Model e DDD

- Il pattern Domain Model [POSA4] deriva dal pattern *Model-Driven Design* del libro *Domain-Driven Design* [DDD] di Eric Evans
  - se il progetto, o qualche sua parte importante, non è in corrispondenza con il modello di dominio, allora il valore del modello è scarso, e la correttezza del progetto è sospetta – se invece c'è una corrispondenza ma è complessa, allora la realizzazione o la manutenzione del sistema sarà problematica
  - pertanto, progetta una porzione del sistema software in modo che rifletta (in modo letterale) il modello di dominio, con una corrispondenza ovvia
  - rivisita continuamente il modello e il progetto, in modo che entrambi riflettano una profonda comprensione del dominio



## Discussione

- Il pattern Domain Model – diversamente da altri pattern architetturali – non si concentra né su tipi di elementi né su tipi di relazioni tra elementi
  - Domain Model fornisce un suggerimento generale relativo alla scelta degli elementi dell'architettura e delle relazioni tra di essi
  - questo suggerimento generale è utile guidare l'applicazione degli altri pattern architetturali

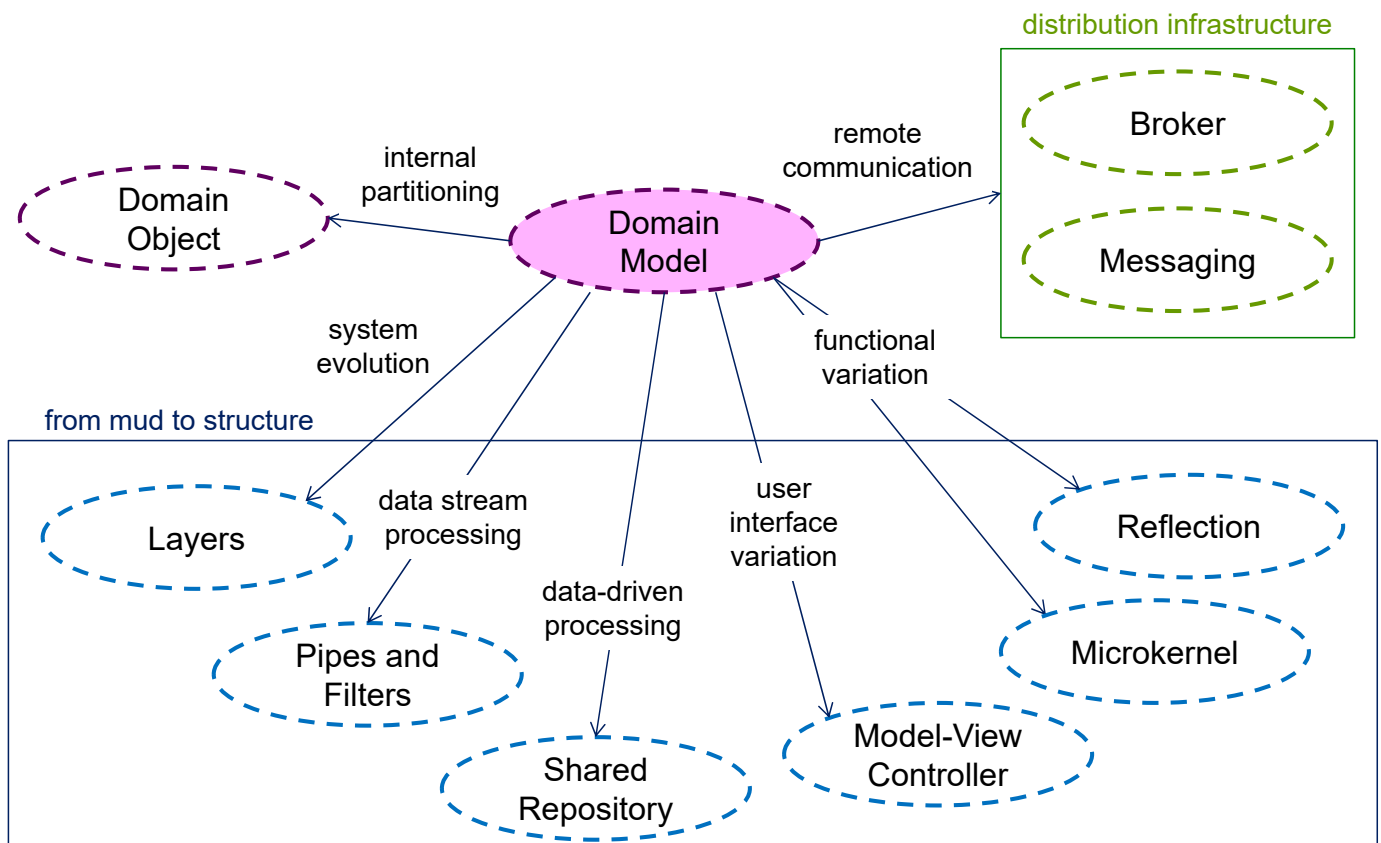


# Discussione

- Un modello di dominio sostiene il passo iniziale nella definizione di un'architettura software
  - aiuta a identificare gli elementi software, e a definire le corrispondenze tra requisiti ed entità software
  - sostiene la comunicazione tra le parti interessate al sistema
- Ciascuna entità del dominio, auto-contenuta e coesa, può essere rappresentata da un *Domain Object* separato
- Gli altri pattern architetturali, più concreti, aiutano
  - a organizzare e collegare gli elementi del modello di dominio – ma anche a raggrupparli e a mantenerli separati – per sostenere gli interessi e le qualità del sistema



# Domain Model e altri pattern [POSA]





## \* Domain Object (POSA4)

- Il pattern architetturale **Domain Object** [POSA4]
  - guida la decomposizione di un intero sistema (o di un elemento architetturale grande) nella realizzazione di un Domain Model
  - si focalizza sugli aspetti tecnici e di qualità del sistema software
  - si basa sui principi di modularità e di separazione degli interessi



## Domain Object

- Contesto
  - decomposizione di un sistema software (o di un suo elemento architetturale) sulla base di Domain Model
- Problema
  - è necessario decomporre un sistema (o un elemento) software, identificandone le parti e le loro relazioni e collaborazioni
  - la decomposizione
    - deve sostenere sia le funzionalità che le qualità del sistema
    - deve evitare una complessità strutturale elevata
    - deve sostenere lo sviluppo e l'evoluzione indipendente delle parti che costituiscono il sistema



# Domain Object

## □ Soluzione

- incapsula ciascuna **responsabilità funzionale** distinta di un'applicazione in un **oggetto di dominio** auto-contenuto
  - per ogni oggetto di dominio, separa la sua interfaccia dalla sua implementazione – e fa collaborare gli oggetti di dominio solo sulla base delle loro interfacce
  - progetta gli oggetti di dominio in modo che siano coesi e debolmente accoppiati



# Domain Object e Domain Model

- Domain Object va applicato congiuntamente a Domain Model
  - suggeriscono di decomporre un sistema in “oggetti di dominio”, guidati da un opportuno “modello del dominio”
  - ciascun “oggetto di dominio” viene usato per rappresentare un elemento del “modello di dominio”
- Questa decomposizione può essere guidata – ad esempio
  - dai casi d'uso
  - dalle informazioni del dominio
  - dalle capacità di business
  - da interessi tecnici



# Domain Object

## □ Discussione

- il pattern Domain Object è basato sull'applicazione dei principi di modularità e di separazione degli interessi
  - la **modularità** è relativa a incapsulamento, coesione e accoppiamento
  - la **separazione degli interessi** è relativa alla separazione delle diverse responsabilità funzionali tra gli oggetti di dominio



# Domain Object

## □ Discussione

- un aspetto fondamentale della soluzione suggerita da Domain Object è che la decomposizione in **elementi** deve essere relativa a **responsabilità funzionali**
  - ma come gestire gli interessi di qualità (**non funzionali**) che il sistema deve possedere? infatti, la decomposizione del sistema non può essere indipendente da queste qualità
  - intuizione: ...





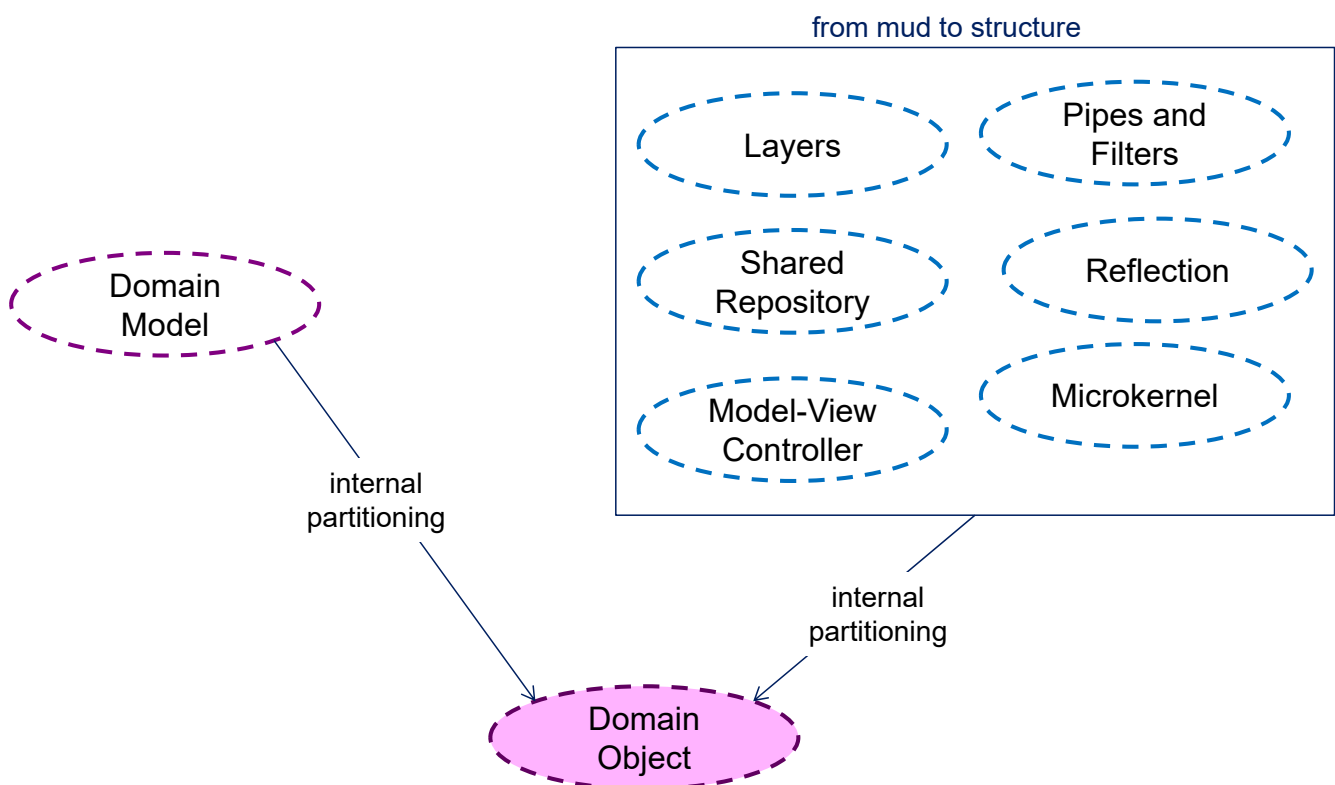
# Domain Object

## Discussione

- esistono diverse tipologie specifiche di oggetti di dominio – nel contesto di pattern architetturali più specifici
  - ad es., oggetti distribuiti, componenti e servizi
  - ciascuno pattern architetturale fornisce criteri propri per la scelta degli oggetti di dominio, delle loro responsabilità e delle loro relazioni e interazioni
  - in ogni caso, è necessario separare responsabilità differenti e incapsularle in modo che possano evolvere in modo indipendente
- l'uso di interfacce esplicite abilita l'accesso remoto agli oggetti di dominio



# Domain Object e altri pattern [POSA]





## \* Discussione

- I pattern architetturali guidano la decomposizione architetturale “fondamentale” di un sistema – o di un componente di un sistema
  - ci sono due tipi principali di decomposizioni – tecniche e di dominio
  - abbiamo illustrato due pattern architetturali POSA fondamentali
    - che suggeriscono di
      - identificare gli elementi dell’architettura (“oggetti di dominio”) con riferimento a un opportuno modello del dominio applicativo del sistema (“modello di dominio”)
      - assegnare responsabilità funzionali agli elementi dell’architettura – e le responsabilità per le qualità a...
    - questi suggerimenti generali sono utili nell’applicazione degli altri pattern architetturali