



09 . Architettura logica e organizzazione in layer

Sviluppo di Applicazioni Software

Ferruccio Damiani

a.a. 2023/24

Università degli Studi di Torino - Dipartimento di Informatica

Attenzione!




©2024 Copyright for this slides by Ferruccio Damiani. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

<https://creativecommons.org/licenses/by/4.0/>

Si noti che

questi lucidi sono basati sul libro di testo del corso “C. Larman, *Applicare UML e i Pattern*, Pearson, 2016” e sul materiale fornito da Matteo Baldoni, Viviana Bono, Claudia Picardi e Gianluca Torta dell'Università degli Studi di Torino.

Table of contents

- 
1. Architettura logica
 2. Strato del dominio
 3. Principio di separazione Modello-Vista



Architettura logica

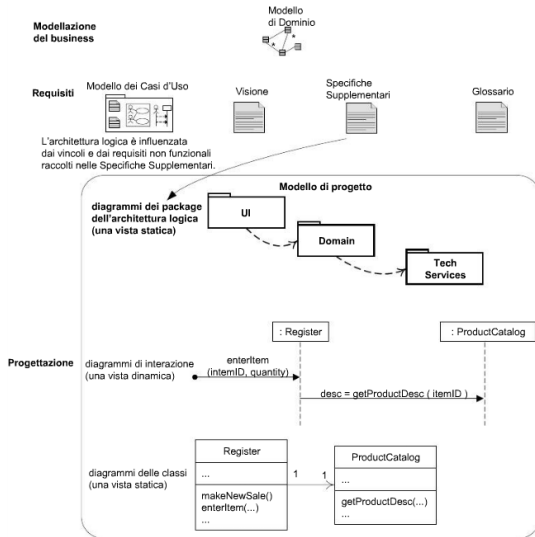


Architettura

La progettazione di un tipico sistema orientato agli oggetti è basata su diversi strati architetturali, come uno strato dell'interfaccia utente, uno strato della logica applicativa (o “del dominio”) e così via.

L'architettura logica può essere illustrata sotto forma di diagrammi dei package di UML.

Alcune relazioni tra gli elaborati di UP



Architettura logica

L'architettura logica di un sistema software è la macro-organizzazione su larga scala delle classi software in package (o namespace), sottoinsiemi e strati.

È chiamata architettura logica poiché **non** vengono prese decisioni su come questi elementi siano distribuiti sui processi o sui diversi computer fisici di una rete (queste ultime decisioni fanno parte dell'architettura di deployment): *platform independent architecture*.

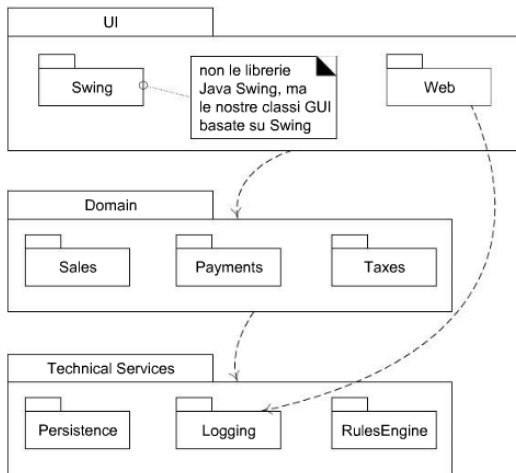
Layer o strato

gruppo di classi software, packages, sottosistemi con responsabilità condivisa su un aspetto importante del sistema.

Gli strati di un'applicazione software comprendono normalmente:

- **User interface** (*interfaccia utente o presentazione*), oggetti software per gestire l'interazione con l'utente e la gestione degli eventi
- **Application logic** o **domain objects** (*logica applicativa o oggetti del dominio*), oggetti software che rappresentano concetti di dominio
- **Technical services** (*servizi tecnici*), oggetti e sottosistemi d'uso generale che forniscono servizi tecnici di supporto.

Esempio dell'architettura per POS NextGen in diagrammi di package UML



/** Strato UI **/

```
com.mycompany.nextgen.ui.swing  
com.mycompany.nextgen.ui.web
```

/** Strato DOMAIN **/

```
// package specifici del progetto NextGen  
com.mycompany.nextgen.domain.sales  
com.mycompany.nextgen.domain.payments
```

/** Strato TECHNICAL SERVICES **/

```
// il nostro strato di persistenza (accesso alla base di dati)  
com.mycompany.service.persistence
```

```
// terze parti  
org.apache.log4j  
org.apache.soap.rpc
```

/** Strato FOUNDATION **/

```
// package foundation creati dal nostro team  
com.mycompany.util
```

©C. Larman. Applicare UML e i Pattern. Pearson, 2016.

Architettura a strati

L'obiettivo dell'architettura a strati è la suddivisione di un sistema complesso in un insieme di elementi software che, per quanto possibile, possano essere sviluppati e modificati ciascuno indipendentemente dagli altri.

Architettura a strati

L'obiettivo dell'architettura a strati è la suddivisione di un sistema complesso in un insieme di elementi software che, per quanto possibile, possano essere sviluppati e modificati ciascuno indipendentemente dagli altri.

Separation of concerns (*separazione degli interessi*) ridurre l'accoppiamento e le dipendenze, *aumenta la possibilità di riuso, facilita la manutenzione e aumenta la chiarezza.*

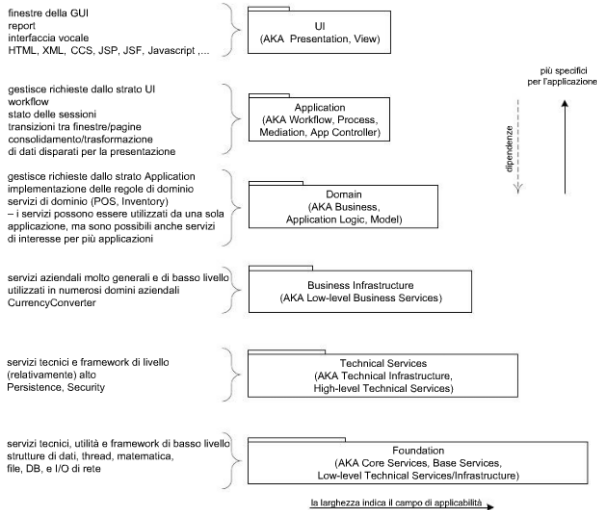
Architettura a strati

L'obiettivo dell'architettura a strati è la suddivisione di un sistema complesso in un insieme di elementi software che, per quanto possibile, possano essere sviluppati e modificati ciascuno indipendentemente dagli altri.

Separation of concerns (*separazione degli interessi*) ridurre l'accoppiamento e le dipendenze, *aumenta la possibilità di riuso, facilita la manutenzione e aumenta la chiarezza.*

Alta coesione, in uno strato le responsabilità degli oggetti devono essere fortemente correlate, l'uno all'altro, e non devono essere mischiate con le responsabilità degli altri strati. Es. gli oggetti dell'interfaccia utente non devono implementare logica applicativa. Aumentare la coesione *aumenta la possibilità di riuso, facilita la manutenzione e aumenta la chiarezza.*

Una tipica architettura a strati





Strato del dominio

Come va progettata la logica applicativa con gli oggetti?

L'approccio consigliato consiste nel creare degli oggetti software con nomi e informazioni simili al dominio del mondo reale, e assegnare a essi responsabilità della logica applicativa.

Come va progettata la logica applicativa con gli oggetti?

L'approccio consigliato consiste nel creare degli oggetti software con nomi e informazioni simili al dominio del mondo reale, e assegnare a essi responsabilità della logica applicativa.

Un oggetto software di questo tipo è chiamato un **oggetto di dominio**, rappresenta una cosa nello spazio di dominio del problema e ha una logica applicativa o di business correlata (es. un oggetto *Sale* è in grado di calcolare il suo totale).

Come va progettata la logica applicativa con gli oggetti?

L'approccio consigliato consiste nel creare degli oggetti software con nomi e informazioni simili al dominio del mondo reale, e assegnare a essi responsabilità della logica applicativa.

Un oggetto software di questo tipo è chiamato un **oggetto di dominio**, rappresenta una cosa nello spazio di dominio del problema e ha una logica applicativa o di business correlata (es. un oggetto *Sale* è in grado di calcolare il suo totale).

Lo strato di dominio fa riferimento al modello di dominio per trarre ispirazione per i nomi delle classi dello strato del dominio.

Come va progettata la logica applicativa con gli oggetti?

L'approccio consigliato consiste nel creare degli oggetti software con nomi e informazioni simili al dominio del mondo reale, e assegnare a essi responsabilità della logica applicativa.

Un oggetto software di questo tipo è chiamato un **oggetto di dominio**, rappresenta una cosa nello spazio di dominio del problema e ha una logica applicativa o di business correlata (es. un oggetto *Sale* è in grado di calcolare il suo totale).

Lo strato di dominio fa riferimento al modello di dominio per trarre ispirazione per i nomi delle classi dello strato del dominio.

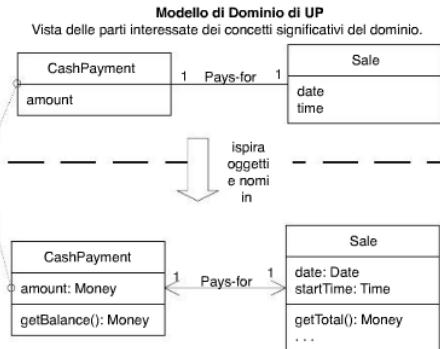
Creando uno strato del dominio ispirandosi al modello di dominio, si ottiene un “*salto rappresentazionale basso*” tra dominio del mondo reale e il progetto software.

Strato del dominio e modello di dominio

CashPayment nel Modello di Dominio è un concetto, ma CashPayment nel Modello di Progetto è una classe software. Non sono la stessa cosa, ma il primo ha *ispirato* il nome e la definizione del secondo.

Questa scelta riduce il salto rappresentazionale.

Questa è una delle grandi idee della tecnologia a oggetti.



Modello di Progetto di UP

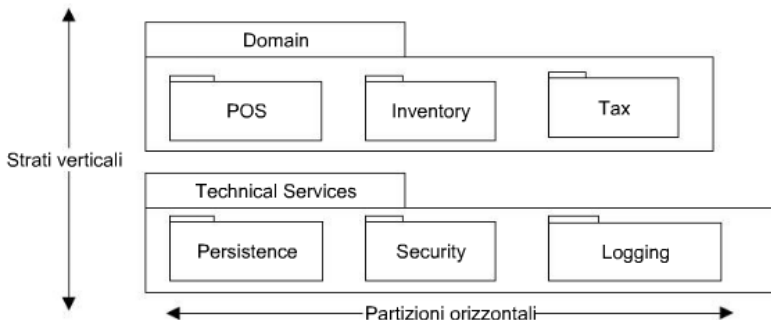
Lo sviluppatore orientato agli oggetti ha tratto ispirazione dal dominio del mondo reale per la creazione di classi software.

Pertanto, il salto rappresentazionale tra il modo in cui le parti interessate concepiscono il dominio e la sua rappresentazione nel software è stato tenuto basso.

Livelli, strati e partizioni

La nozione originaria di livello è di strato logico.

Si dice che gli **strati** di un'architettura rappresentano le *sezioni verticali*, mentre le **partizioni** rappresentano una *divisione orizzontale* di sottosistemi relativamente paralleli di uno strato.



©C. Larman. Applicare UML e i Pattern. Pearson, 2016.



Principio di separazione Modello-Vista

Principio di separazione Modello-Vista

1. **Non relazionare o accoppiare oggetti non UI con oggetti UI:** gli oggetti non UI non devono essere connessi o accoppiati direttamente agli oggetti UI (es. non si deve permettere a un oggetto software di “dominio”, non UI, di avere un riferimento a un oggetto UI).
2. **Non incapsulare la logica dell'applicazione in metodi di UI:** non mettere logica applicativa (es. calcolo delle imposte) nei metodi di un oggetto dell'interfaccia utente. Gli oggetti UI dovrebbero solo inizializzare gli elementi dell'interfaccia utente, ricevere eventi UI e delegare le richieste di logica applicativa agli oggetti non 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
- Gli oggetti UI inizializzano elementi UI, ricevono eventi UI e delegano le richieste della logica dell'applicazione agli oggetti non UI (oggetti di dominio)

Principio di separazione Modello-Vista

Modello = strato di dominio, Vista = strato UI

Modello è sinonimo per lo strato degli oggetti del dominio. **Vista** è un sinonimo per gli oggetti dell'interfaccia utente, come finestre, pagine web, applet e report.

Principio di separazione Modello-Vista

Modello = strato di dominio, Vista = strato UI

Modello è sinonimo per lo strato degli oggetti del dominio. **Vista** è un sinonimo per gli oggetti dell'interfaccia utente, come finestre, pagine web, applet e report.

Principio di separazione Modello-Vista

Gli oggetti del **modello** (**dominio**) non devono avere una conoscenza diretta degli oggetti della **vista** (*UI*), almeno in quanto oggetti della vista.

È un principio fondamentale del pattern **Model-View-Controller** (*MVC*). MVC era un pattern di *Smalltalk-80* relativo a oggetti dati (*modelli*), elementi della GUI (*vista*) e gestori degli eventi del mouse e della tastiera (*controller*).



- Le classi di dominio **incapsulano** le informazioni e il comportamento relativi alla logica applicativa
- Le classi della vista sono relativamente leggere, esse sono **responsabili** dell'input e dell'output e di catturare gli eventi della GUI ma **non mantengono** i dati dell'applicazione né forniscono direttamente la logica applicativa

Vantaggi del principio di separazione Modello-Vista

- Favorire la definizione coesa dei modelli
- Consentire lo sviluppo separato degli strati del modello e dell'interfaccia utente
- Minimizzare l'impatto sullo strato del dominio dei cambiamenti dei requisiti relativi all'interfaccia
- Consentire di connettere facilmente nuove viste a uno strato del dominio esistente
- Consentire viste multiple, simultanee sugli stessi oggetti modello
- Consentire l'esecuzione dello strato di modello indipendente da quella dello strato dell'interfaccia utente (*batch* o *a messaggi*)
- Consentire un porting facile dello strato di modello ad un altro framework per l'interfaccia utente

SSD, operazioni di sistema e strati

Gli SSD mostrano le operazioni di sistema ma nascondono gli oggetti specifici della UI. Gli oggetti dello strato UI inoltreranno (o delegheranno) le richieste da parte dello strato UI allo strato del dominio.

I messaggi inviati dallo strato UI allo strato del domino saranno i messaggi mostrati negli SSD.

