



06 . Modello di Dominio

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. Disciplina Modellazione del Business: Modello di Dominio
 2. Creare un modello di dominio
 3. Classi concettuali
 4. Associazioni
 5. Attributi
 6. Ultime verifiche al modello

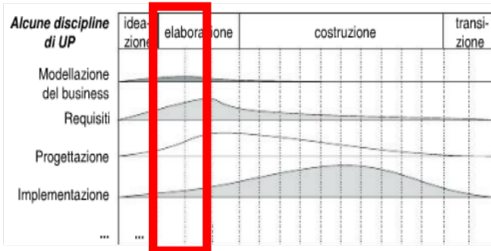


Disciplina Modellazione del Business: Modello di Dominio

UP maps

Tabella 2.1 Scenario di Sviluppo di esempio (i – inizio; r – raffinamento).

Disciplina	Pratica	Elaborato	Ideazione	Elaboraz.	Costr.	Transiz.
Modellazione del business	modellazione agile workshop requisiti	Modello di Dominio		i		
	esercizio sulla visione votazione a punti	Casi d'Uso				
		Visione	i	r		
		Specifica	i	r		
		Supplementare				
		Glossario	i	r		
Progettazione	modellazione agile sviluppo guidato dai test	Modello di Progetto		i	r	
		Documento dell'Architettura		i		
		Software				
		Modello dei Dati		i	r	
Implementazione	sviluppo guidato dai test programmazione a coppie integrazione continua standard di codifica	...				
Gestione del progetto	gestione del progetto agile riunioni Scrum giornaliere	...				
...						

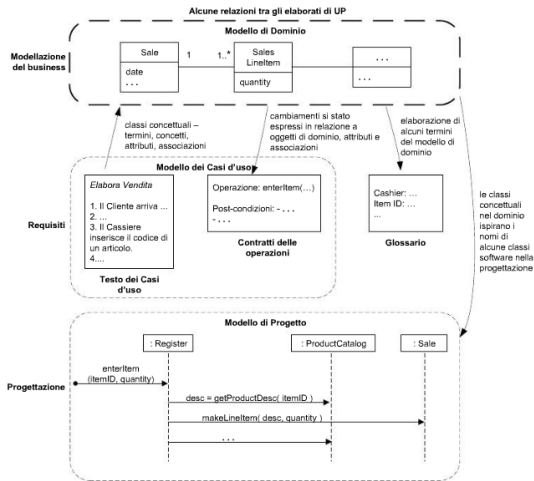


L'impegno relativo nelle discipline cambia a seconda delle fasi.

Questo esempio è solo un suggerimento, non è da prendere alla lettera.

Relazioni tra elaborati di UP

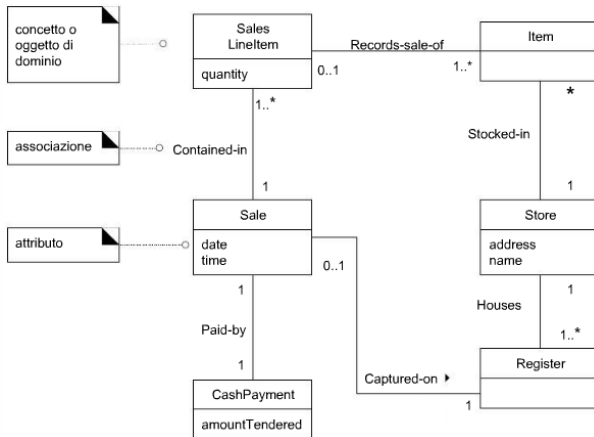
- Il modello di dominio può evolversi in modo da mostrare i concetti significativi relativi ai casi d'uso
- Il modello di dominio a sua volta può influenzare i contratti delle operazioni, il glossario e il Modello di Progetto



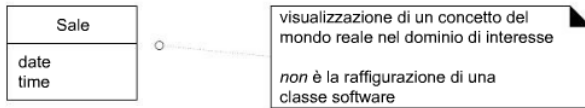
- Rappresentazione **visuale** delle **classi concettuali**, oggetti reali del dominio (non sono oggetti software!)
- Sviluppato nell'ambito della disciplina di *Modellazione del Business*
 - Specializzazione del “Business Object Model”
- Insieme di **diagrammi di classi UML**, includono:
 - **Oggetti** di dominio-classi concettuali
 - **Associazioni** tra classi concettuali
 - **Attributi** di classi concettuali
 - Non appaiono operazioni (*firma di metodi o responsabilità*)
- È un **dizionario visuale**
- Non è un modello dei dati
- Un modello di dominio è un modo particolare di utilizzare un diagramma delle classi di UML, con uno scopo specifico

Esempio di Modello di Dominio di POS NextGEN

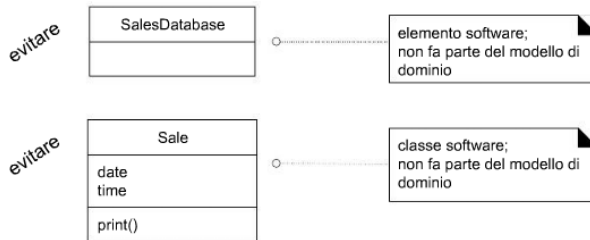
Un modello secondo un punto di vista concettuale. L'identificazione di un ricco insieme di classi concettuali è al centro dell'analisi OO.



Classi concettuali non classi software!



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



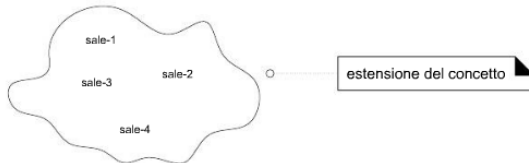
©C. Larman. Applicare UML e i Pattern. Pearson, 2016.

Classi concettuali

- Il **simbolo** è una parola o un'immagine usata per rappresentare la classe concettuale
- L'**intensione** è la definizione (in linguaggio naturale) della classe concettuale
- L'**estensione** è l'insieme degli oggetti della classe concettuale

Una classe concettuale

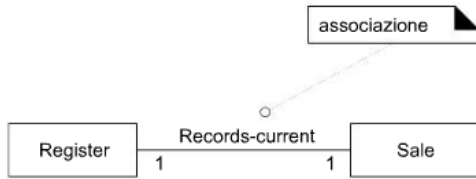
rappresenta un **concetto** del mondo reale o del dominio di interesse di un sistema che si sta modellando (modellazione *ontologica* VS modellazione *concettuale* o di *domino*).



Una associazione

è una **relazione** tra classi (più precisamente, tra le istanze di queste classi) che indica una connessione significativa e interessante.

Se si pensa alle classi in maniera estensionale, un'associazione tra due classi è un insieme di coppie di oggetti dalle due classi.



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

Un attributo

è un **valore logico** (ovvero una dato, una proprietà elementare) degli oggetti di una classe.

Ciascun oggetto della classe ha un proprio valore, separato, per quella proprietà.

Una funzione che associa un valore a ciascun oggetto della classe.



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

Riduzione del “gap di rappresentazione”

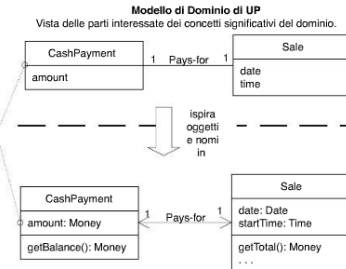
Comprendere i concetti chiave e la terminologia relativa al dominio del discorso del problema, più specificatamente:

- **Comprendere** il dominio del sistema da realizzare e il suo vocabolario
- Definire un **linguaggio comune** che abiliti la comunicazione tra le diverse parti interessate al sistema
- Come **fonte di ispirazione** per la progettazione dello strato del 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.



Creare un modello di dominio



Ci restringiamo ai requisiti scelti per la progettazione nell'iterazione corrente

- Trovare le classi concettuali
- Disegnarle come classi in un diagramma delle classi UML
- Aggiungere le associazioni
- Aggiungere gli attributi



Classi concettuali



Ci restringiamo ai requisiti scelti per la progettazione nell'iterazione corrente

- Riuso-modifica di modelli esistenti (pattern di analisi specifici in un determinato dominio di applicazione)
- Utilizzo di elenchi di categorie (preparando un elenco di classi candidate)
- Analisi linguistica (delle descrizioni testuali di un dominio):
 - Il mapping nome-classe non è automatico
 - Il linguaggio naturale è ambiguo
 - Fonti: casi d'uso descritti in formato dettagliato

Esempio: elenco di categorie

Dal dominio POS, utilizziamo un elenco di categorie che enfatizzano le transazioni commerciali e le loro relazioni con le altre cose.

Categoria di classe concettuale	Esempi
transazioni commerciali <i>Linea guida:</i> sono aspetti critici (riguardano denaro), dunque si inizi con le transazioni.	<i>Sale, Payment (o CashPayment) Reservation</i>
elementi/righe di transazioni <i>Linea guida:</i> le transazioni spesso sono composte da righe per gli articoli correlati, quindi queste vanno considerate subito dopo le transazioni.	<i>SalesLineItem</i>
prodotto o servizio correlato a una transazione o a una riga di transazione per articolo <i>Linea guida:</i> le transazioni sono <i>per</i> qualcosa (un prodotto o un servizio). Vanno considerate subito dopo.	<i>Item Flight, Seat, Meal</i>
dove viene registrata la transazione? <i>Linea guida:</i> importante.	<i>Register, Ledger (o SalesLedger), FlightManifest</i>
ruoli di persone o organizzazioni correlati alle transazioni; attori nei casi d'uso <i>Linea guida:</i> normalmente dobbiamo sapere quali sono le parti coinvolte in una transazione.	<i>Cashier, Customer, Store MonopolyPlayer Passenger, Airline</i>

Esempio: elenco di categorie

Dal dominio POS, utilizziamo un elenco di categorie che enfatizzano le transazioni commerciali e le loro relazioni con le altre cose.

luogo della transazione; luogo del servizio	<i>Store</i> <i>Airport, Plane, Seat</i>
eventi significativi, spesso con un'ora o un luogo che è necessario ricordare	<i>Sale, Payment (o CashPayment)</i> <i>MonopolyGame</i> <i>Flight</i>
oggetti fisici <i>Linea guida:</i> questo è particolarmente importante quando si crea software per il controllo di dispositivi, oppure simulazioni.	<i>Item, Register</i> <i>Board, Piece, Die</i> <i>Airplane</i>
descrizioni di oggetti <i>Linea guida:</i> vedere il Paragrafo 12.13 per una discussione.	<i>ProductDescription</i> <i>FlightDescription</i>
cataloghi <i>Linea guida:</i> le descrizioni sono spesso contenute in un catalogo.	<i>ProductCatalog</i> <i>FlightCatalog</i>
contenitori di oggetti (fisici o informazioni)	<i>Store, Bin</i> <i>Board</i> <i>Airplane</i>

Esempio: elenco di categorie

Dal dominio POS, utilizziamo un elenco di categorie che enfatizzano le transazioni commerciali e le loro relazioni con le altre cose.

oggetti in un contenitore	<i>Item</i>
	<i>Square (in un Board)</i>
	<i>Passenger</i>
altri sistemi che collaborano	<i>CreditAuthorizationSystem</i>
	<i>AirTrafficControl</i>
registrazioni di questioni finanziarie, di lavoro, contrattuali e legali	<i>Receipt, Ledger</i>
	<i>MaintenanceLog</i>
strumenti finanziari	<i>Cash, Check, LineOfCredit</i>
	<i>TicketCredit</i>
piani, manuali, documenti cui si fa regolarmente riferimento per eseguire il lavoro	<i>DailyPriceChangeList</i>
	<i>RepairSchedule</i>

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

L'analisi linguistica è un'altra fonte di ispirazione e i casi d'uso formato dettagliato sono un'ottima descrizione a cui ispirarsi per questa analisi.

Scenario principale di successo (o Flusso di base):

1. Il **Cliente** arriva alla **cassa POS** con gli **articoli** e/o i **servizi** da acquistare.
2. Il **Cassiere** inizia una nuova **vendita**.
3. Il **Cassiere** inserisce il **codice identificativo di un articolo**.
4. Il Sistema registra la **riga di vendita per l'articolo** e mostra la **descrizione dell'articolo**, il suo **prezzo**, il **totale** parziale. Il prezzo è calcolato in base a un insieme di regole di prezzo.

Il Cassiere ripete i passi 2-3 fino a che non indica che ha terminato.

L'analisi linguistica è un'altra fonte di ispirazione e i casi d'uso formato dettagliato sono un'ottima descrizione a cui ispirarsi per questa analisi.

5. Il Sistema mostra il totale con le **imposte** calcolate.
6. Il Cassiere riferisce il totale al Cliente e richiede il **pagamento**.
7. Il Cliente paga e il Sistema gestisce il pagamento.
8. Il Sistema registra la vendita completata e invia informazioni sulla **vendita** e sul pagamento ai sistemi esterni di **Contabilità** (per la contabilità e le **commissioni**) e di **Inventario** (per l'aggiornamento dell'inventario).
9. Il Sistema genera la **ricevuta**.
10. Il Cliente va via con la ricevuta e gli articoli acquistati.

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

L'analisi linguistica è un'altra fonte di ispirazione e i casi d'uso formato dettagliato sono un'ottima descrizione a cui ispirarsi per questa analisi.

Estensioni (o Flussi alternativi):

...

7a. Pagamento in contanti:

1. Il Cassiere inserisce l'**importo in contanti** presentato dal Cliente.
2. Il Sistema mostra il **resto dovuto** e apre il cassetto della **cassa**.
3. Il Cassiere deposita il contante presentato e restituisce il resto in contanti al Cliente.
4. Il Sistema registra il **pagamento in contanti**.

Esempio: definizione delle classi

Un modello di dominio conterrà una collezione, in un certo senso arbitraria, di astrazioni e termini del dominio che i modellatori considerano significative.



Si consideri il concetto "*Ricevuta*" :

- Se le informazioni che contiene si possono derivare da altre fonti (altre classi concettuali) si può escludere dal modello di dominio
- Se ha un ruolo in termini di regole di business, business ad esempio conferisce il diritto al possessore di restituire oggetti acquistati acquistati, allora è bene includere il concetto nel modello di dominio

Nota: nell'iterazione corrente di POS la restituzione articoli non è considerata, quindi *Ricevuta* viene esclusa dal modello di dominio in questa iterazione

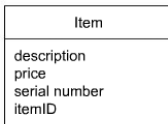
Classi “Descrizione”

Classe Descrizione

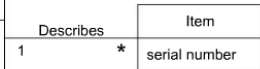
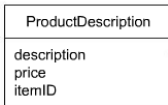
Una classe descrizione contiene informazioni che descrivono qualcos'altro.

È utile avere un'associazione che collega la classe e la sua classe descrizione (*pattern Item-Descriptor*)

- È necessaria una descrizione di un articolo o servizio indipendentemente dall'attuale esistenza di istanze dell'articolo o servizio
- L'eliminazione delle istanze di un articolo (esempio, Item) darebbe luogo ad una perdita di informazioni che è necessario conservare
- Si vogliono ridurre le informazioni ridondanti



Sconsigliato



Migliore



Associazioni

Associazioni

Un'associazione rappresenta una relazione tra due o più classi che indica una connessione significativa tra le istanze di quella classe.

È utile includere:

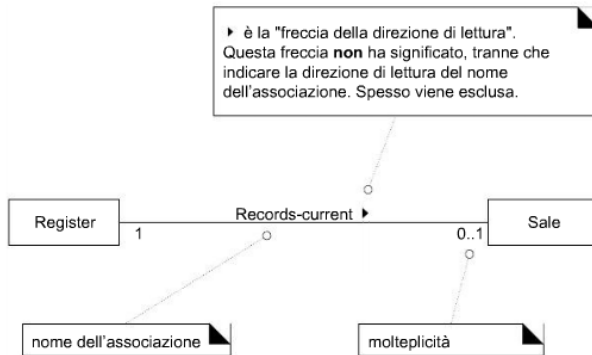
- Associazioni la cui conoscenza della relazione deve essere conservata per una qualche durata (associazioni “*da ricordare*”)
- Associazioni derivate dall'elenco di associazioni comuni (si veda più avanti)

Nota: è utile introdurre solo le associazioni davvero rilevanti al dominio modellato.

Caratterizzare una associazione con

- Nome significativo seguendo la traccia *NomeClasse-FraseVerbale-NomeClasse*
- Molteplicità e direzione di lettura

Nota: un'associazione è per natura **bidirezionale** (non è un'affermazione su collegamenti tra entità software). La direzione di lettura non è una specifica di visibilità.



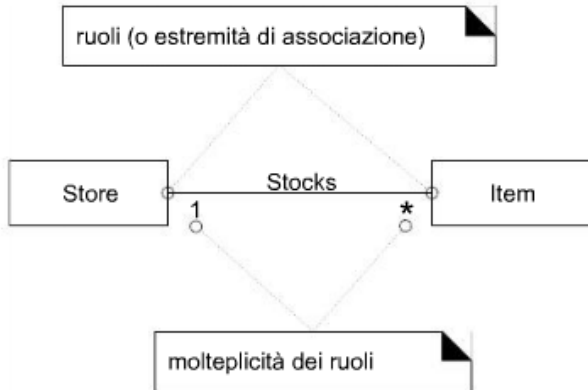
©C. Larman. Applicare UML e i Pattern. Pearson, 2016.

Ruoli nelle associazioni

Ruolo

Ciascuna estremità di una associazione è anche chiamata **ruolo**. I ruoli possono avere, opzionalmente:

- espressione di **molteplicità**
- **nome**
- **navigabilità**



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

Molteplicità delle associazioni

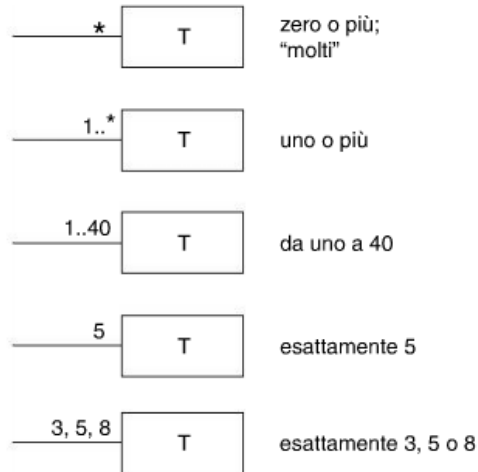
Molteplicità

La **molteplicità** di un ruolo definisce *quante istanze di una classe* possono essere associate ad una istanza di un'altra.

A..B vuol dire che A è la molteplicità minima (spesso 0 o 1) e B è la molteplicità massima (spesso 1 o *).

I casi possibili in base alla molteplicità massima sono:

- *uno-a-molti*
- *molti-a-uno*
- *molti-a-molti*
- *uno-a-uno*



Molteplicità delle associazioni

Molteplicità

Il valore di una molteplicità comunica quante istanze possono essere associate in modo valido a un'altra istanza, in un particolare momento, piuttosto che in un arco di tempo.



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

Attenzione!

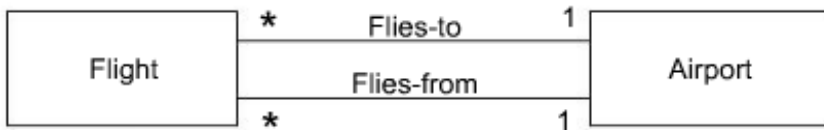
È possibile che uno specifico “*item*” possa essere in più “*store*” nel tempo **MA** in qualsiasi particolare momento l’item è in **un solo** store.

Attenzione!

Il valore di una molteplicità dipende dall’interesse del modellatore e dello sviluppatore del software, perché comunica un vincolo di dominio che si rifletterà o potrebbe riflettersi nel software.

Associazioni multiple tra due classi

È possibile che due classi siano collegate da più di una associazione.



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

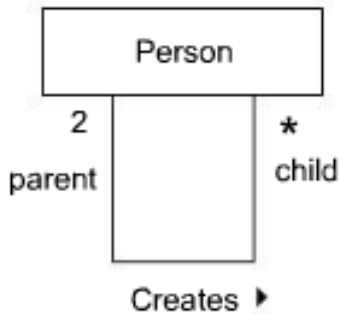
Nomi di ruolo

È possibile indicare il nome di un ruolo.



Associazioni riflesse

Una classe può anche avere un'associazione con se stessa. I nomi di ruolo sono in questo caso utili per indicare la molteplicità.

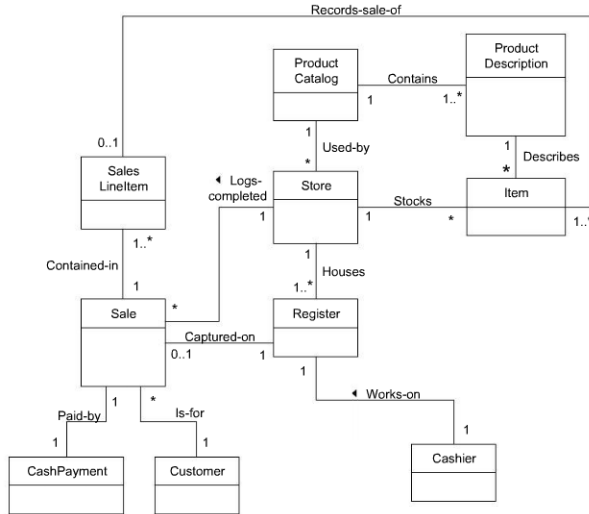


Esempio: modello di dominio (parziale) per POS NextGen

Associazioni derivate dall'elenco di associazioni comuni per un sistema informativo.

Categoria	Esempi
A è una transazione correlata a un'altra transazione B	CashPayment — Sale Cancellation — Reservation
A è un elemento/riga di una transazione B	SalesLineItem — Sale
A è un prodotto o servizio per una transazione (o riga per l'articolo) B	Item — SalesLineItem (o Sale) Flight — Reservation
A è un ruolo relativo a una transazione B	Customer — Payment Passenger — Ticket
A è una parte fisica o logica di B	Drawer — Register Square — Board Seat — Airplane
A è contenuto fisicamente o logicamente in B	Register — Store, Item — Shelf Square — Board Passenger — Airplane
A è una descrizione per B	ProductDescription — Item FlightDescription — Flight
A è noto/registrato/memorizzato/riportato/acquisito in B	Sale — Register Piece — Square Reservation — FlightManifest
A è un membro di B	Cashier — Store Player — MonopolyGame Pilot — Airline
A è una sottounità organizzativa di B	Department — Store Maintenance — Airline
A utilizza o gestisce o possiede B	Cashier — Register Player — Piece Pilot — Airplane
A è vicino/prossimo a B	SalesLineItem — SalesLineItem Square — Square City — City

Esempio: modello di dominio (parziale) per POS NextGen



L'*aggregazione* è, in UML, un tipo di associazione che suggerisce una relazione intero-parte. Un esempio di aggregazione è la relazione fra un'automobile e le sue ruote.

Composizione

La **composizione**, o *aggregazione composta*, è un tipo di **forte di aggregazione intero-parte**:

- ciascuna istanza della parte appartiene ad **una sola** istanza del composto alla volta
- ciascuna parte deve sempre appartenere **a un** composto
- la vita delle parti è limitata da quella del composto: le parti possono essere create dopo il composto (ma non prima) e possono essere distrutte prima del composto (ma non dopo)

L'aggregazione fra automobile e le sue ruote non è una composizione.

Notazione della composizione

La notazione UML per la composizione è un rombo pieno su una linea di associazione, posto all'estremità della linea vicina alla classe per il composto.



composizione



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

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



Attributi

Attributi

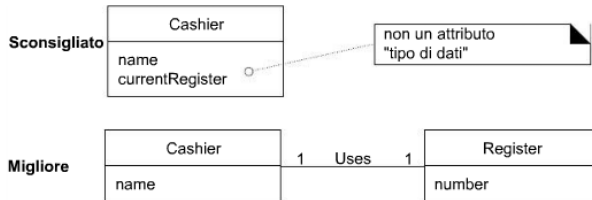
Un **attributo** di una classe rappresenta un *valore* (un dato) degli oggetti di quella classe

Aggiungere attributi che:

- Sono tipi di dati primitivi (boolean, date, number, character, string, ecc.)
- Sono tipi enumerativi (es. tipo servizio: gold, silver)

Caratterizzare un attributo con:

- Origine: derivato/non derivato
- Tipo di dato, ovvero un vincolo sui valori del dominio



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

Gli attributi in un modello di dominio devono preferibilmente essere di **tipo di dato**.

Attributi

Un **attributo** di una classe rappresenta un *valore* (un dato) degli oggetti di quella classe

Aggiungere attributi che:

- Sono tipi di dati primitivi (boolean, date, number, character, string, ecc.)
- Sono tipi enumerativi (es. tipo servizio: gold, silver)

Caratterizzare un attributo con:

- Origine: derivato/non derivato
- Tipo di dato, ovvero un vincolo sui valori del dominio



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

Se nel mondo reale non pensiamo a un concetto X come a un numero, un testo o un valore di un tipo di dato, allora probabilmente X è una classe concettuale, non un attributo.

Attributi

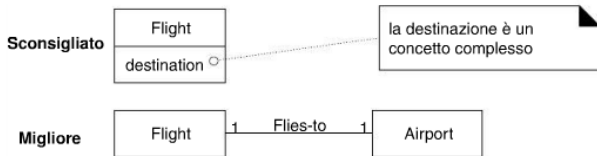
Un **attributo** di una classe rappresenta un *valore* (un dato) degli oggetti di quella classe

Aggiungere attributi che:

- Sono tipi di dati primitivi (boolean, date, number, character, string, ecc.)
- Sono tipi enumerativi (es. tipo servizio: gold, silver)

Caratterizzare un attributo con:

- Origine: derivato/non derivato
- Tipo di dato, ovvero un vincolo sui valori del dominio

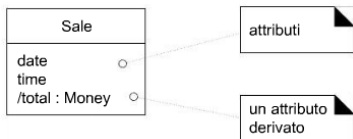


©C. Larman. Applicare UML e i Pattern. Pearson, 2016.

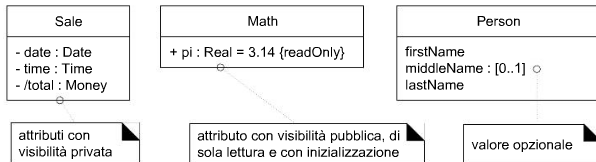
Classi concettuali vanno correlate con un'associazione, non con un attributo.

Notazione per gli attributi

Gli attributi sono mostrati nella seconda sezione del rettangolo per una classe. Opzionalmente è possibile mostrare il loro tipo e altre informazioni, ad esempio la visibilità o il tipo di lettura.



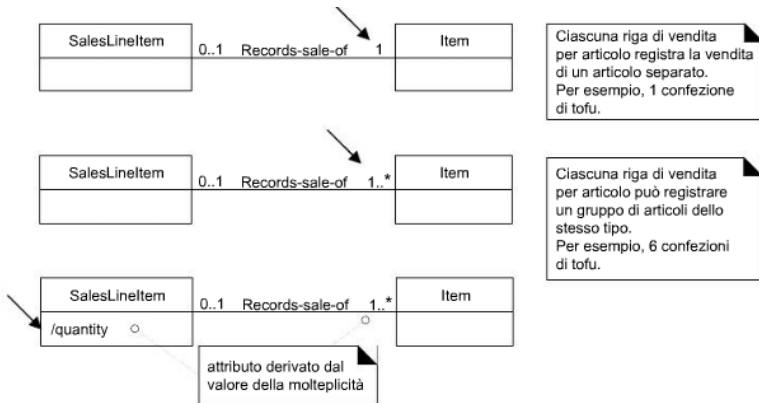
©C. Larman. Applicare UML e i Pattern. Pearson, 2016.



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

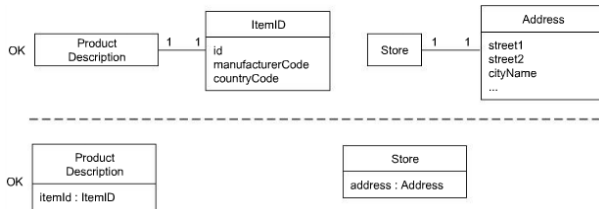
Attributi derivati

Un attributo può essere calcolato o derivato dalle informazioni contenute negli oggetti associati.
Un attributo derivato può essere indicato dal simbolo “/” prima del nome dell'attributo.



Quando introdurle?

- Dati composti da sezioni separate (es., *nome, numero di telefono*)
- Quando ci sono operazioni associate ai dati, come la validazione o il parsing (es., *codice fiscale*)
- Quantità con unità di misura (es. totale da pagare è caratterizzato da una valuta)



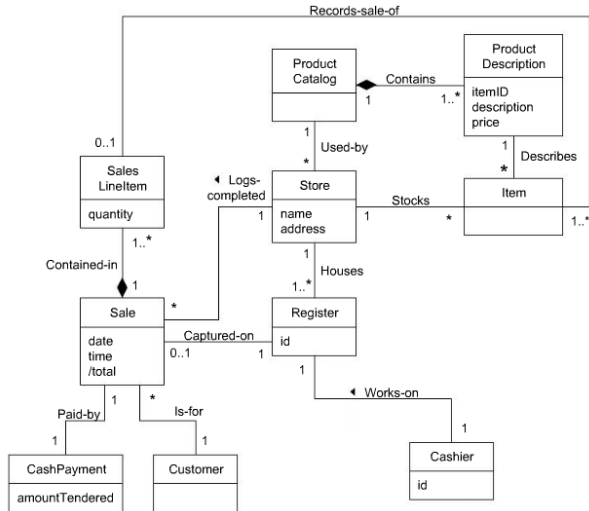
©C. Larman. Applicare UML e i Pattern. Pearson, 2016.



Ultime verifiche al modello

- Verificare le classi concettuali introdotte:
 - Alternativa classe classe-attributo attributo
 - Classi descrizione
- Verificare le associazioni:
 - Indipendenza delle associazioni diverse che sono relative alle stesse classi
- Verificare gli attributi:
 - Non introdurre attributi per riferirsi ad altre classi concettuali (chiavi esterne). Usare le associazioni in questo caso

Esempio: modello di dominio (parziale) per POS NextGen



Modello di dominio in UP

Il modello di dominio in UP viene creato in primo luogo durante le iterazioni dell'elaborazione, quando vi è la massima necessità di capire i concetti significativi e di rappresentarne alcuni come classi software durante il lavoro di progettazione.

Disciplina	Elaborato Iterazione →	Ideaz. I1	Elab. E1..En	Costr. C1..Cn	Trans. T1..T2
Modellazione del business	Modello di Dominio		i		
Requisiti	Modello dei Casi d'Uso (SSD) Visione Specifiche Supplementari Glossario	i i i i	r r r r		
Progettazione	Modello di progetto Documento dell'Architettura Software Modello dei Dati		i i i	r r	

La **generalizzazione** è un'astrazione basata sull'identificazione di *caratteristiche comuni tra concetti*, che porta a definire una relazione tra un concetto più generale (*superclasse*) e un concetto più specializzato o specifico (*sottoclasse*). Vale il principio di **sostituibilità**.

...
Estensioni:

7b. Pagamento con carta di credito:

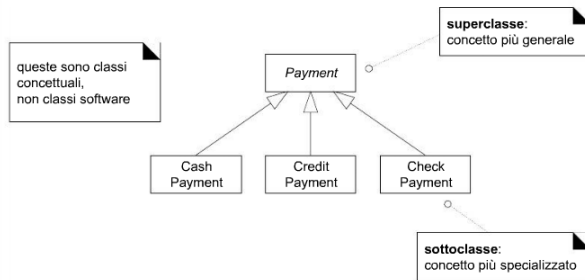
1. Il Cliente inserisce le informazioni sulla propria **carta di credito**.
2. Il Sistema invia una **richiesta di autorizzazione al pagamento** a un sistema esterno di **Servizio di Autorizzazione al Pagamento** per richiedere l'**approvazione del pagamento**.
- 2a. Il Sistema rileva un problema nella comunicazione con il sistema esterno:
 1. Il Sistema segnala l'errore al Cassiere.
 2. Il Cassiere chiede al Cliente un metodo di pagamento alternativo.
3. Il Sistema riceve l'**approvazione del pagamento** e segnala l'approvazione al Cassiere.
- 3a. Il Sistema riceve un **rifiuto per il pagamento**:
 1. Il Sistema segnala il rifiuto al Cassiere.
 2. Il Cassiere chiede al Cliente un metodo di pagamento alternativo.
4. Il Sistema registra il **pagamento con carta di credito**, che comprende l'approvazione del pagamento.
5. Il Sistema presenta il meccanismo di inserimento della firma per il pagamento.
6. Il Cassiere chiede al Cliente la firma per la ricevuta del pagamento con carta di credito. Il Cliente inserisce la firma.

7c. Pagamento con assegno:

1. Il Cliente compila un **assegno** e lo consegna, insieme alla **carta d'identità**, al Cassiere.
2. Il Cassiere scrive il numero della carta d'identità, lo inserisce e richiede un'**autorizzazione al pagamento con assegno**.
3. Il Sistema genera una **richiesta di pagamento con assegno** e la invia a un **Servizio di Autorizzazione Assegni esterno**.
4. Il Sistema riceve un'approvazione del pagamento con assegno e segnala l'approvazione al Cassiere.
5. Il Sistema registra il **pagamento con assegno**, che comprende l'approvazione del pagamento.

...

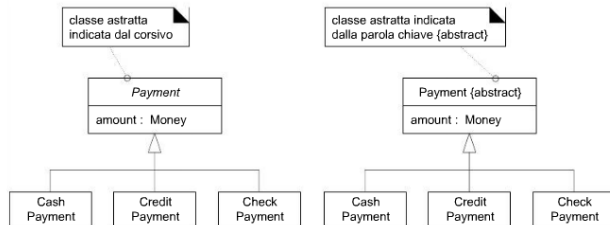
La **generalizzazione** è un'astrazione basata sull'identificazione di *caratteristiche comuni tra concetti*, che porta a definire una relazione tra un concetto più generale (*superclasse*) e un concetto più specializzato o specifico (*sottoclasse*). Vale il principio di **sostituibilità**.



Generalizzazione

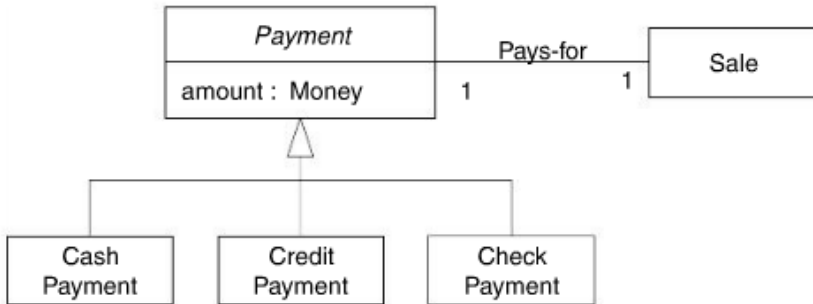
La **generalizzazione** è un'astrazione basata sull'identificazione di *caratteristiche comuni tra concetti*, che porta a definire una relazione tra un concetto più generale (*superclasse*) e un concetto più specializzato o specifico (*sottoclasse*). Vale il principio di **sostituibilità**.

Una classe concettuale è **astratta** se ciascun suo elemento è anche elemento di una delle sue sottoclassi.



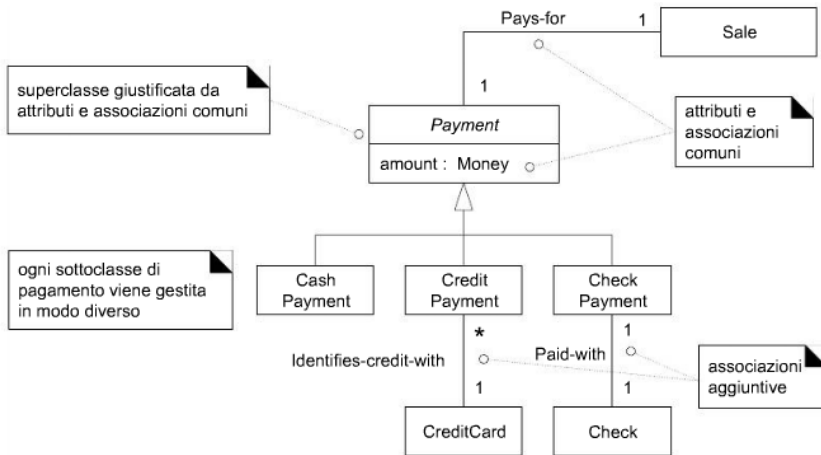
©C. Larman. Applicare UML e i Pattern. Pearson, 2016.

Conformità di una sottoclasse concettuale



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

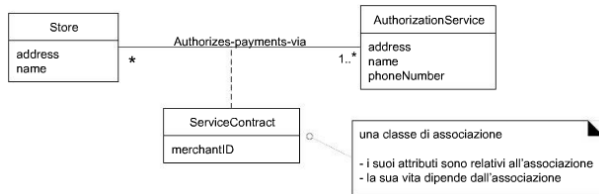
Associazioni aggiuntive in una sottoclasse concettuale



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

Classi di associazioni

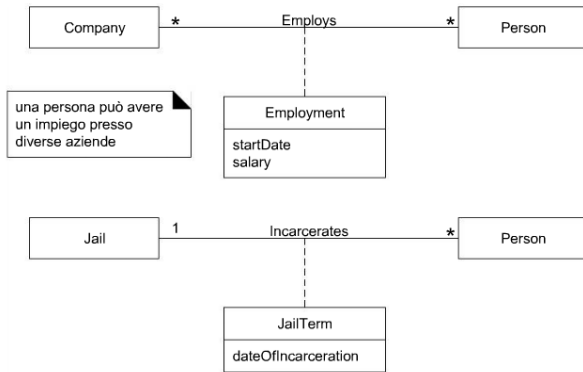
Permettono di aggiungere attributi e altre caratteristiche alle associazioni.



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

Classi di associazioni

Permettono di aggiungere attributi e altre caratteristiche alle associazioni.



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