Sistemi Informativi Aziendali

Appunti per il corso - Capitolo 4

Fulvio Corno Marco Torchiano

Politecnico di Torino – Dipartimento di Automatica e Informatica

Versione 0.5.0

14 ottobre 2021

INDICE

Indice **i**

4 Modellazione concettuale **1**
4.1 Linguaggi di modellazione e UML 1
4.2 Livelli di astrazione 2
4.3 Classi 3
4.3.1 Classe 3
4.3.2 Oggetto 4
4.3.3 Attributi 4
4.4 Associazioni 5
4.4.1 Definizione di Associazione 5
4.4.2 Molteplicità delle associazioni 7
4.4.3 Interpretazione operativa 8
4.4.4 Association class 9
4.4.5 Aggregazione 10
4.5 Generalizzazione e specializzazione 11
4.6 Metodo di analisi 13
4.6.1 Procedura 13
4.6.2 Qualità dei modelli concettuali 14
4.6.3 Pattern di modellazione 14
4.7 Esempio di analisi 18

Bibliografia **25**

CAPITOLO 4

MODELLAZIONE CONCETTUALE

This is your last chance. After this, there is no turning back.
You take the blue pill - the story ends, you wake up in your bed
and believe whatever you want to believe. You take the red pill -
you stay in Wonderland and I show you how deep the rabbit-hole goes.
Morpheus - The Matrix

Quando abbiamo a che fare con sistemi informativi che permetto di supportare ed attuare dei processi, dobbiamo prendere in considerazione aspetti diversi: informazioni gestite ed elaborate, flusso del processo, interazioni con gli utenti. L'idea fondamentale è che per ciascun punto di vista dobbiamo essere in grado di descrivere il sistema: questo avviene tramite opportuni modelli astratti e semplificati.

4.1 Linguaggi di modellazione e UML

Un modello è una rappresentazione *astratta* e *semplificata* di un oggetto o sistema, materiale o immateriale, esistente o da costruire.

Si tratta di una visione *semplificata* perchè tipicamente rappresenta soltanto un determinato punto di vista sul sistema. Inoltre è una rappresentazione *astratta* in quanto, rispetto al sistema che descrive, riporta una selezione degli elementi, unicamente quelli ritenuti rilevanti per il punto di vista che si sta analizzando e per gli scopi del modello.

La modellazione viene solitamente svolta per mezzo di un *linguaggio di modellazione*, ovvero un insieme di costrutti e regole a cui è associata una semantica ben definita. I linguaggi possono essere di varia natura, ma solitamente nell'ambito dell'informatica si utilizzano linguaggi di modellazione testuali, grafici, o misti (ovvero con elementi sia grafici che testuali).

Uno dei modelli utilizzati per descrivere un sistema informativo è il *modello informativo concettuale*. Questo modello descrive il punto di vista relativo alle informazioni che vengono acquisite, elaborate e memorizzate dal sistema informativo che si vuole descrivere.

Oltre ai modelli informativi concettuali illustrati in questo capitolo, saranno presi in esame i *modelli di processo* (descritti nel capitolo **??**), i *modelli dei casi d'uso* (descritti nel capitolo **??**) ed i *modelli dell'interazione utente* (trattati nel capitolo **??**).

In questo libro, per le definizione dei modelli, sarà adottatolo Unified Modeling Language (UML) [1]. UML è un linguaggio standard[1] di modellazione, viene detto unificato

[1]UML è mantenuto sviluppato dall'OMG (Object Management Group), inoltre la versione 2.4.1 è definita nello standard ISO/IEC 19505 del 2012.
La documentazione ufficiale è disponibile all'url: `http://www.omg.org/spec/UML/`

perché include al suo interno diversi tipi di modelli ed ha una rappresentazione grafica ben definita. La rappresentazione grafica dei modelli è definita in maniera precisa dallo standard, ed è accompagnata da una semantica ben precisa. Ad ogni modello compreso in UML corrisponde un diagramma; in questo corso saranno trattati:

- diagramma delle classi UML (Class Diagram), per la modellazione concettuale
- diagramma dei casi d'uso UML (USe Case Diagram), per la modellazione funzionale.

La scelta di UML riflette un'esigenza di semplicità e coerenza all'interno del corso. Tuttavia UML non è l'unica possibile scelta. I due linguaggi di modellazione più diffusi per la costruzione di modelli informativi concettuali sono la notazione ER [2] (entità-relazione) e i modelli delle classi UML (Unified Modeling Language). Per quanto riguarda le tecniche di diagramma di processo sono i diagrammi di attività UML e i BPMN [3]. Per quanto riguarda invece i modelli funzionali, oltre ai diagrammi dei casi d'uso, le tecniche utilizzate nella pratica possono essere molto varie ed utilizzare delle notazioni testuali non sempre codificare in un vero e proprio linguaggio formale.

In questo capitolo si affronterà il tema della modellazione concettuale delle informazioni rappresentate da un sistema informativo, e si presenterà la notazione dei diagrammi delle classi UML applicato a tale contesto.

4.2 Livelli di astrazione

Il modello concettuale descrive un SI dal punto di vista dei concetti (o entità) che caratterizzano il dominio del problema a cui esso risponde. La modellazione concettuale consiste nel costruire un modello che fornisca una descrizione ottimale (si veda anche la sezione 4.6.2) delle informazioni che vengono acquisite, manipolate e memorizzate dal sistema informativo.

La modellazione concettuale è uno dei risultati di una delle prime fasi di analisi dei requisiti di un sistema informativo, che deve basarsi sulla raccolta di informazioni e dati sulla base dei quali si definiscono i modelli concettuali. L'obiettivo della modellazione concettuale è di catturare e rappresentare con un modello i principali concetti astratti, le loro caratteristiche e le relazioni che esistono tra essi.

Possiamo identificare due[2] principali livelli di astrazione dei concetti rilevanti per un sistema informativo: il livello astratto e quello concreto (si veda la tabella 41). A livello astratto, parliamo di concetto, entità, classe, categoria, tipo; a livello concreto, utilizziamo i termini istanza, oggetto, esempio, occorrenza.

Tabella 41: Livelli di astrazione

Astratto	Concetto
	Entità
	Classe
	Categoria
	Tipo
Concreto	Istanza
	Elemento
	Oggetto
	Esempio
	Occorrenza

Ad esempio, dato il concetto generale di studente, le istanze o le occorrenze che discendono da esso rappresentano particolari studenti reali (Mario Rossi, Paolo Neri, ...). Gli

[2]In teoria è possibile identificare molti livelli di astrazione, superiori a quello qui chiamato astratto, si parla di meta-modellazione quando si sviluppano descrizioni in tali livelli più alti.

studenti che siedono in aula o leggono questo libro sono delle entità concrete, che esistono, ma per quanto riguarda il sistema informativo, tutti sono sostanzialmente riconducibili al concetto astratto di studente. Ogni studente avrà delle caratteristiche che sarà utile andare ad illustrare nel SI, e altre che invece non saranno utili. L'astrazione consiste proprio nel rimuovere le informazioni non rilevanti. All'idea di studente dal punto di vista astratto si assoceranno matricola, nome, cognome e data di nascita, mentre invece non verranno riportati altri dettagli meno significativi quali, altezza, colore degli occhi, tipo dei vestiti etc. perché al fine del funzionamente del SI queste sono informazioni non rilevanti.

Il modello informativo concettuale descrive gli elementi al livello astratto, mentre il sistema informativo (una volta costruito) durante il suo funzionamento manipolerà i dati delle istanze, ovvero le informazioni al livello concreto.

La modellizzazione avviene in modo astratto, ossia è una definizione astratta di un insieme di entità concrete (istanze di una classe). Le istanze concrete sono i veri dati su cui lavora il sistema informativo.

Il modello concettuale sintetizza gli aspetti astratti e cerca di rispondere a tre domande fondamentali:

- Quali sono i *concetti principali*?
  Assegnando un nome e definendo una *classe*.
- Quali sono le caratteristiche rilevanti per descrivere le istanze del concetto?
  Quali sono le variabili che memorizzo, ovvero gli *attributi*.
- Quali sono le *relazioni* esistenti tra i concetti?
  Come si possono legare e correlare tra loro concetti diversi tramite le *associazioni*.

4.3 Classi

I concetti principali del dominio del problema che si affronta sono rappresentati nel modello UML tramite le classi.

4.3.1 Classe

In un modello concettuale i concetti sono i primi elementi da descrivere, e questi vengono rappresentati dalle **classi**.

La classe è un elemento del modello che rappresenta un insieme di oggetti che hanno caratteristiche in comune, ossia che sono descrivibili attraverso le stesse caratteristiche. Possono essere cose, persone, fatti, ecc. Non ci sono vincoli di alcun tipo sulla natura di concetti rappresentanti tramite le classi.

Quando identifichiamo un concetto, inseriamo nel modello una classe alla quale associamo un nome, per convenzione si utilizza un sostantivo singolare. Tornando all'esempio, il nome di una possibile classe è *Studente*.

La classe ha una rappresentazione grafica (figura 4.1) che consiste in un rettangolo suddiviso in tre comparti; il nome della classe compare nel comparto più in alto.

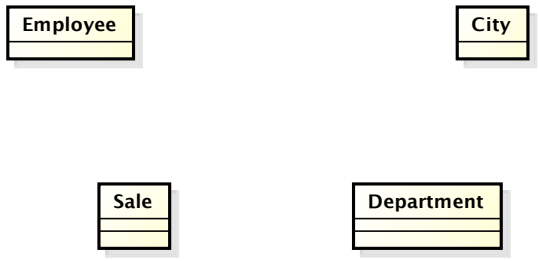


Figura 4.1: Rappresentazione grafica delle classi

4.3.2 Oggetto

Le classi sono costrutti astratti, ad esse corrispondono degli costrutti concreti, o istanze, che vengono rappresentate in UML tramite gli **oggetti**. Un oggetto è un'istanza concreta di una classe, esso rappresenta un'entità fisica o immateriale che viene descritta nel sistema informativo.

La rappresentazione UML degli oggetti è illustrata in figura 4.2 e si basa sulla notazione:
`Nome dell'istanza concreta : Nome della Classe`

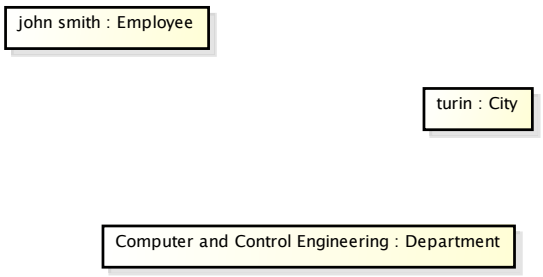


Figura 4.2: Rappresentazione grafica degli oggetti

La presenza di una classe in un modello concettuale implica che il SI dovrà memorizzare le informazioni relative alle istanze di tale classe. Tipicamente questo viene realizzato tramite un database nel quale vengono memorizzate le informazioni.

Ad esempio se il modello concettuale contiene la classe *Studente* questo significa che il SI conterrà un database con una tabella in cui memorizzare le informazioni relative agli studenti. Le informazioni sui singoli studenti (ovvero gli oggetti studente o le istanze della classe Studente) saranno memorizzati nelle righe di tale tabella.

A scanso di equivoci, conviene ricordare che le classi presenti nel modello concettuale rappresentano sempre "informazioni a proposito di ...", e mai gli utenti del sistema. Ad esempio, la classe *Studente* rappresenta le *informazioni sugli studenti* che il sistema deve mantenere, e non i singoli studenti. Occorre tenere presente questa distinzione, soprattutto per evitare frequenti errori di modellazione, dove erroneamente si definiscono delle associazioni che cercano di rappresentare le azioni compiute dagli utenti del sistema. Questo genere di informazioni non è di pertinenza del modello informativo concettuale, bensì del modello di processo.

Richiamando l'interpretazione insiemistica, se le classi corrispondono agli insiemi, gli oggetti corrispondono agli elementi appartenenti a tali insiemi. Perciò si dice che un oggetto che è un'istanza di una classe, appartiene alla classe; facendo riferimento all'appartenenza insiemistica ($\in$).

4.3.3 Attributi

I concetti solitamente sono caratterizzati da alcune informazioni, che permettono di descrivere le loro istanze. Queste informazioni sono definite tramite gli **attributi** (o proprietà) delle classi.

Un attributo è definito tramite un nome ed un tipo. Gli attributi sono molto simili alle variabili dichiarate nei linguaggi di programmazione. Il valore di un attributo rappresenta il tipo di dato che può assumere (stringa, numero intero, numero float, date, ecc.).
Esempi: `Nome:String`, `Anno:int`, `Stipendio:double`

Ciascun oggetto (istanza della classe) avrà propri valori per ciascun attributo. Ad esempio ciascuno studente avrà un proprio nome, cognome, data di nascita, ecc. Spesso tra gli attributi ne è presente uno che identifica in maniera univoca l'oggetto: ad esempio per le

persone, il codice fiscale (di tipo stringa); per gli studenti, la matricola (di tipo numerico intero). Tale attributo viene detto *chiave*[3].

Si può assumere che esista un identificatore implicito `ID` per ciascuna classe. Tale identificatore può essere omesso dalla lista degli attributi. Se la chiave è rilevante e/o viene menzionata esplicitamente nei requisiti, l'attributo può essere aggiunto tra gli attributi. Nel caso si voglia utilizzare una diversa chiave, è necessario definirla esplicitamente tra gli attributi. Si raccomanda in questo caso di aggiungere uno stereotipo «`key`» all'attributo.

Da un punto di vista grafico (figura 4.3) gli attributi vanno elencati nel secondo comparto della classe, quello centrale.

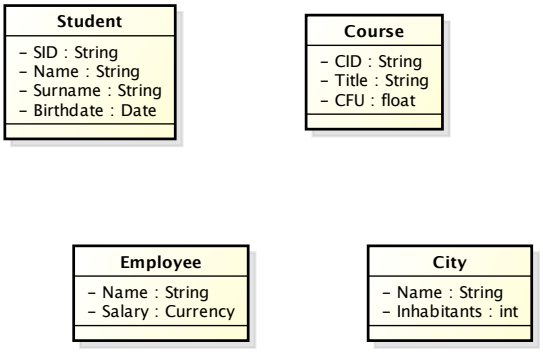


Figura 4.3: Rappresentazione grafica degli attributi

I tipi di attributi adottati in questo libro corrispondono ai principali tipi di variabile definiti nel linguaggio Java e sono illustrati in tabella 42.

Tabella 42: Tipi degli attributi

Tipo	Descrizione
`int`	numero intero
`double`	numero reale (con parte decimale) in doppia precisione
`float`	numero reale (con parte decimale) in singola precisione
`boolean`	valore logico, vero/falso, si/no
`String`	stringa di caratteri, testo
`Date`	data in termini di anno, mese, giorno
`Time`	ora (ore, minuti, secondi, ..)
`Currency`	denaro (valore e valuta)

4.4 Associazioni

Nella pratica gli oggetti sono raramente utili individualmente, le informazioni in un sistema informativo riportano spesso il legame tra più oggetti, ad esempio: uno studente sarà legato al corso seguito.

4.4.1 Definizione di Associazione

Questi legami vengono descritti tramite il concetto di **associazione** tra le classi, che rappresenta un legame logico tra due classi.

[3]Se si pensa alla realizzazione del modello nello schema di un database relazione la chiave assume un significato fondamentale, e più in generale può consistere di uno o più attributi che collettivamente permettono di identificare univocamente le istanze, o righe della tabella.

L'esistenza di un'associazione tra due classi indica la possibilità che esista un legame tra due oggetti delle rispettive classi. Le classi hanno come istanze gli oggetti, mentre le associazioni hanno come istanze i link (detti anche occorrenze dell'associazione) tra coppie di oggetti. In assenza di un'associazione non è possibile definire un link tra gli oggetti.

L'idea di associazione si basa sul concetto di relazione matematica, ovvero di sottoinsieme del prodotto cartesiano tra gli insiemi di oggetti delle classi connesse dall'associazione (figura 4.4). Come conseguenza, data una coppia di oggetti, istanze di classi connesse da un'associazione, essi possono essere collegati al più da una istanza dell'associazione stessa. Come conseguenza, nell'esempio in figura, non è possibile che uno studente frequenti più volte lo stesso corso.

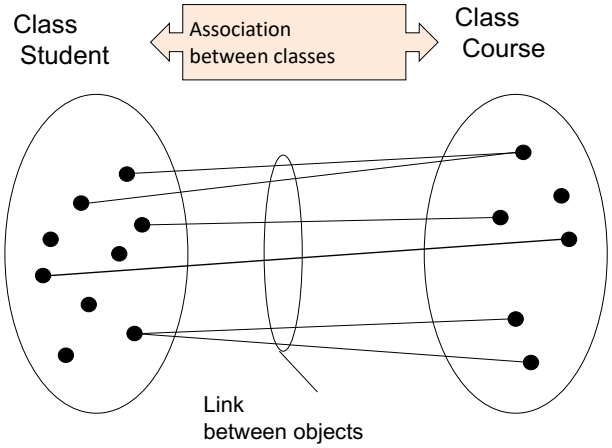


Figura 4.4: Associazioni e link

La rappresentazione grafica di un'associazione consiste in una linea (tipicamente un segmento ma anche una spezzata) che unisce le due classi; vicino alla linea viene riportato il nome dell'associazione (figura 4.5). È possibile affiancare al nome un piccolo triangolino (la testa di un freccia) che indica il verso in cui leggere il nome dell'associazione.

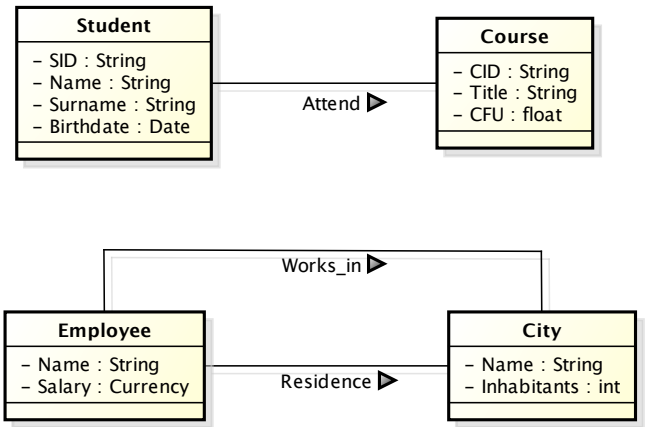


Figura 4.5: Rappresentazione grafica delle associazioni

Esempi: date le classi `Studente` e `Corso`, posso collegarle usando un'associazione che chiamerò `Esame`. Oppure, date le classi `Impiegato` e `Città`, le posso collegare con due associazioni, `Lavora in` e `Risiede in`.

È possibile che un'associazione colleghi una classe a se stessa, in tale caso si parla di associazione *riflessiva* (o ricorsiva). Tali tipi di associazione possono essere di due tipi (figura 4.6):

Simmetriche : dove i due estremi dell'associazione sono semanticamente equivalenti e di conseguenza non esiste un verso di lettura privilegiato. L'associazione ci dice quindi solo che le singole istanze di quella classe possono essere collegate da una relazione. Ad esempio l'associazione `Friend`.

Asimmetriche : dove i due estremi dell'associazione sono semanticamente ben distinti e rappresentano diversi ruoli, i nomi dei ruoli vengono riportati vicino alle rispettive estremità della linea spezzata che rappresenta l'associazione. Ad esempio: l'associazione `Supervise` prevede due ruoli `manager` e `employee`.

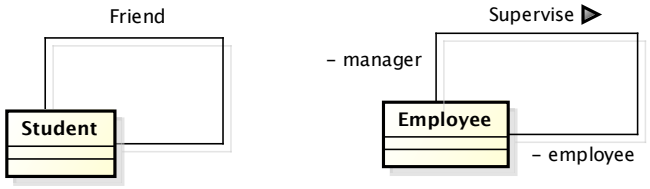


Figura 4.6: Associazioni ricorsive

È possibile utilizzare i ruoli anche per associazioni non riflessive quando sia utile per chiarire il ruolo giocato da diverse classe nella relazione che li lega.

4.4.2 Molteplicità delle associazioni

È sempre utile aggiungere informazioni descrittive al modello, ad esempio dato uno studente a quanti corsi può essere iscritto? Dato un corso, quanti studenti possono iscriversi? Queste informazioni si possono fornire tramite il costrutto della **molteplicità** (detta anche cardinalità). La specifica della cardinalità, infatti, prescrive un numero minimo ed un numero massimo di link ai quali ciascuna istanza di una classe può partecipare, per un determinata relazione.

Il ragionamento da fare per la molteplicità è: "data una determinata istanza, con quante altre istanze dell'altra classe essa si può legare?". Il ragionamento va fatto per entrambe le classi coinvolte nella relazione.

Si specificano un valore di minimo ed uno massimo di occorrenze di quella classe che possono essere legate alle occorrenze dell'altra. Nell'esempio illustato in figura 4.7, un'automobile può montare un minimo di 0 e un massimo di 4 ruote, viceversa una ruota può essere montata su nessuna o al massimo una macchina. In questo modo possiamo avere un'idea più chiara di come sono correlati i concetti all'interno delle classi.

Figura 4.7: Rappresentazione della molteplicità

I valori usati per definire le cardinalità sono normalmente i seguenti:

- Per la cardinalità minima:

  `0` Partecipazione opzionale

1 Partecipazione obbligatoria

- Per la cardinalità massima:

  1 Al più un'istanza (relazione univoca)

  ***** Molte istanze possibili

  n Al più *n* istanze, questa indicazione solitamente non viene usata in quanto è difficile da quantificare, salvo casi specifici, ed inoltre per comprendere la complessità del modello è sufficiente distinguere il caso *uno* dal caso *molti*.

La cardinalità massima è la più importante perché in base ad essa possiamo distinguere tre tipi fondamentali di associazioni: associazione 1 a 1 (un'entità è legata esattamente ad una sola altra entità), associazione 1 a molti (una entità può essere legata a più entità diverse), associazione molti a molti (più entità possono essere legate a più entità). In figura 4.8 è possibile osservare alcuni esempi:

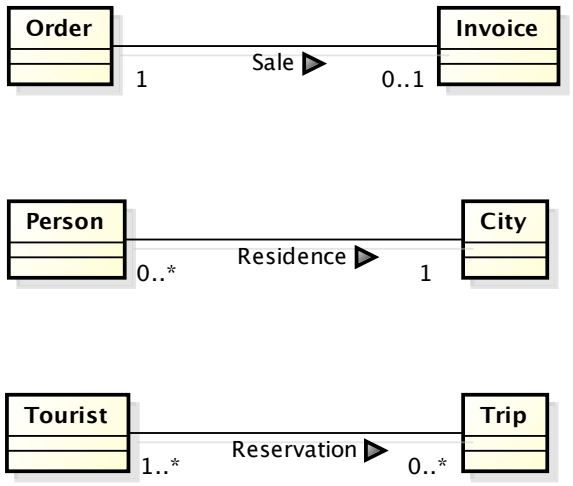


Figura 4.8: Esempi di molteplicità

1 a 1 : un ordine può avere da zero a 1 fatture, una fattura deve per forza avere un ordine.

1 a molti/molti a 1 : molte persone hanno la residenza in una sola città

Molti a molti : un turista può aver prenotato più viaggi o non averne prenotato alcuno, un viaggio può essere prenotato da più turisti e deve per forza essere prenotato da almeno un turista.

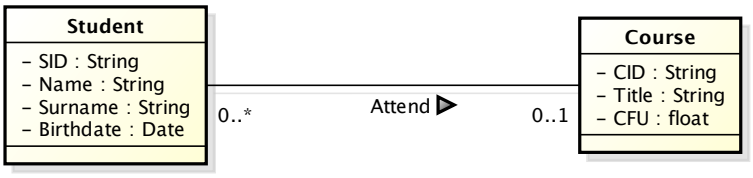
Per determinare correttamente la cardinalità, si usa porsi su una classe e "guardare" l'altra interrogandosi sulla molteplicità. Nel caso di Ruota e Macchina (riportato in figura 4.7) la classe Ruota guarda verso la classe Macchina e ci si chiede: Su quante macchine posso montare una ruota? 0 macchine o al più una sola macchina, e quindi la cardinalità della classe che "guardo" sarà `0..1`. Successivamente, mi pongo sulla classe Macchina e "guardo" la classe Ruota: Quante ruote può montare una macchina? Da 0 a 4, e dunque la cardinalità della classe che "guardo" (in questo caso la classe Ruota) sarà `0..4`.

4.4.3 Interpretazione operativa

Gli elementi fondamentali di un modello informativo concettuale descritto in UML sono: classi, attributi e associazioni. Questi elementi sono sufficienti per descrivere quali informazioni sono immagazzinate e manipolate da un sistema informativo.

Una classe indica quali informazioni saranno memorizzate per ciascun oggetto. Gli oggetti di una classe possono essere rappresentati in una semplice griglia. Consideriamo l'esempio di due classi `Student` e `Course` legate dall'associazione `Attend`, come illustrato nel diagramma in figura 4.9.

Le tabelle nella parte centrale di figura 4.9 riportano tre istanze della classe Student e tre istanze della classe Course. Gli attributi *CID* e *SID* sono chiavi in quanto permettono di identificare univocamente i relativi oggetti.



SID	Name	Surname	Birthdate
S2345	John	Smith	1990/04/12
S1234	Jane	Brown	1991/07/11
S5678	Mario	Rossi	1991/11/05

CID	Title	CFU
C001	Information Systems	8
C002	Advanced Programming	10
C003	Calculus	10

	C001	C002	C003
S2345			X
S1234	X		
S5678	X		

Figura 4.9: Interpretazione Operativa

L'associazione tra due classi può essere vista come un sottoinsieme del prodotto cartesiano tra le due classi[4]. Con riferimento all'esempio precedente, possiamo rappresentare l'associazione `Attend` come una matrice in cui sulle righe abbiamo le matricole degli studenti e sulle colonne i codici dei corsi che frequentano. Un esempio di istanza della relazione è mostrato nella tabella nella parte inferiore di figura 4.9: le 'X' rappresentano le istanze della relazione.

Osservando le cardinalità dell'associazione, notiamo che un uno studente può frequentare `0..1` corsi, perciò in ciascuna riga della matrice potremo trovare al più una 'X'. Invece ogni corso può essere frequentato da zero o più studenti (`0..*`), quindi ciascuna colonna potrà contenere da nessuna fino a molte 'X'.

Osservando questa rappresentazione delle istanze dell'associazione possiamo notare come, data una coppia di oggetti (ovvero una riga ed una colonna), può esistere al più un'istanza dell'associazione. Ovvero non è possibile che una coppia di oggetti sia collegata da più istanze della stessa associazione.

4.4.4 Association class

Talvolta alcune informazioni non trovano posto in una classe, ma risulta naturale attribuirle all'associazione. Questo accade quando si hanno associazioni molti-a-molti. Ad esempio il compenso di un consulente può variare in base alle aziende per cui lavora: in tale caso il compenso non è un attributo del consulente, altrimenti sarebbe lo stesso per tutte le aziende con cui lavora; ma non è neppure un attributo dell'azienda, altrimenti sarebbe lo stesso per tutti i consulenti dell'azienda. Si tratta di un'informazione legata ad una coppia consulente-azienda, ovvero al link che lega un consulente all'azienda.

[4]Più precisamente, tra gli insiemi di oggetti istanze delle due classi

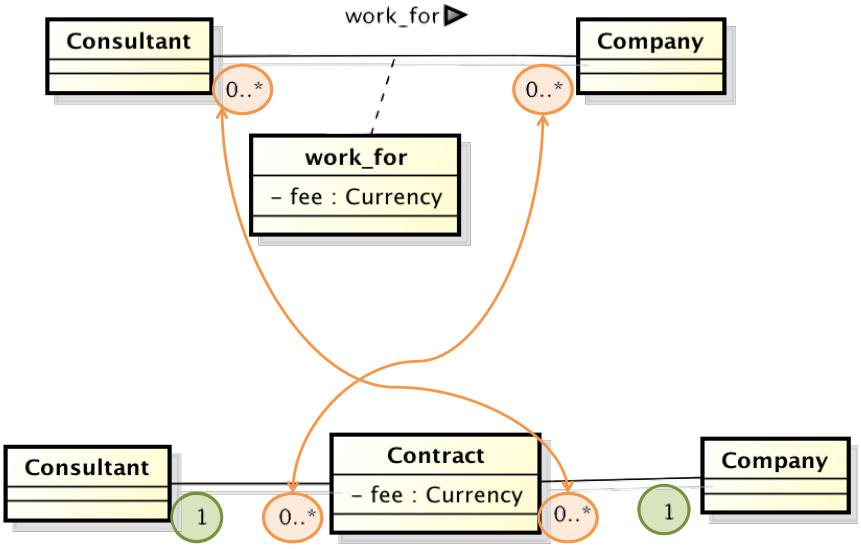


Figura 4.10: Classe di associazione e classe intermedia

Tramite la *classe di associazione* è possibile definire degli attributi che vengono assegnati all'associazione. La rappresentazione grafica è uguale a quella di una classe ma è collegata tramite una linea tratteggiata all'associazione per la quale definisce gli attributi (parte alta della figura 4.10).

Un'alternativa (non del tutto equivalente) all'uso della classe di associazione è quella di inserire una classe vera e propria (detta anche *classe intermedia*) tra le due classi connesse dall'associazione e rimpiazzare l'associazione con due associazioni tra la classe intermedia e le due di partenza come mostrato nell'esempio di figura 4.10. È possibile in maniera praticamente automatica trasformare un modello nell'altro, facendo attenzione a come vengono definite le molteplicità delle associazioni.

Le due soluzioni mostrate in figura 4.10 **non** sono equivalenti: la soluzione con la classe di associazione è in grado di rappresentare solo i casi in cui un consulente lavora al più una volta per una data azienda[5], mentre la soluzione con la classe intermedia non presenta questa limitazione. Inoltre, una classe intermedia è una classe a tutti gli effetti, e pertanto può partecipare anche ad altre associazioni con altre classi. Le classi di associazione, invece, non possono partecipare a nessuna altra associazione se non quella che ne definisce l'esistenza.

L'uso delle classi di associazione dovrebbe essere limitato ai casi strettamente necessari in quanto la loro lettura è meno immediate delle semplici associazioni. In generale dovrebbe essere limitata alle associazioni molti-a-molti. Quando si ha la tentazione di mettere una classe di associazione al posto di una associazione 1-a-molti, generalmente è possibile mettere gli attributi della classe di associazione nella classe dal lato molti, senza perdere informazioni e guadagnando in chiarezza.

4.4.5 Aggregazione

Spesso gli oggetti di una classe sono caratterizzati (e definiti) non soltanto dai propri attributi ma anche da altri oggetti a cui sono collegati. In tal caso gli oggetti collegati idealmente potrebbero essere degli attributi dell'oggetto principale ma praticamente non lo sono perché non hanno un tipo semplice (intero, stringa, etc.)

[5]Si ricordi che l'associazione (semplice o con classe di associazione) è essenzialmente una relazione matematica e quindi non è possibili che esista più di un'istanza tra due oggetti

Per rappresentare questa particolare relazione si utilizza l'associazione di **aggregazione**[6], che è un caso particolare di associazione. L'aggregazione rappresenta una relazione la cui semantica sia del tipo "A è composta da B" o "B sono parti di A".

La rappresentazione grafica prevede una notazione simila a quella di un'associazione con l'aggiunta di un rombo vuoto dal lato della classe principale, come mostrato in figura 4.11).

Si tratta in ogni caso di una relazione tra le classi, ma un po' particolare. Ad esempio se prendessimo come esempio la macchina essa presenta al suo interno diverse parti legate ad essa con una certa molteplicità: 1 motore, 4 copertoni, 1 cd player etc..

Per definizione, la molteplicità di una relazione di aggregazione è sempre del tipo 1-a-molti, pertanto non è necessario specificarla sul diagramma.

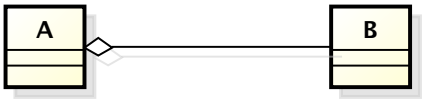


Figura 4.11: Rappresentazione dell'aggregazione

4.5 Generalizzazione e specializzazione

Nella modellazione di realtà complesse può succedere che alcune classi si assomiglino, avendo in comune alcuni attributi. Come regola generale, è bene evitare la ridondanza e la ripetizione di elementi all'interno di un modello. La ragione è duplice: la ripetizione porta ad un modello più complesso per il semplice motivo che conterrà più elementi; inoltre ogni volta che si dovrà modificare il modello, le modifiche agli elementi ripetuti dovranno essere replicate per ciascuna istanza con un aumento del lavoro necessario e della possibilità di errore.

Un caso frequente di somiglianza è quando una classe rappresenta un caso particolare di un'altra. Nei diagrammi delle classi è possibile rappresentare la relazione di **specializzazione** o **generalizzazione** tra le classi. Si dice che una classe `B` specializza una classe `A` se la classe `B` è identica alla classe `A` (ossia contiene gli stessi attributi e partecipa alle stesse associazioni) più alcuni attributi aggiuntivi o relazioni aggiuntive. Tramite la generalizzazione è possibili definire gli attributi comuni una sola volta nella classe più generica ed *ereditarli* nelle classi più specifiche, senza il bisogno di ripeterli esplicitamente.

La rappresentazione grafica della generalizzazione consiste in una freccia con una grande punta triangolare, che va dalla classe più specifica (con maggiori dettagli) a quella più generale (con meno attributi). La figura 4.12 mostra un esempio di generalizzazione.

La generalizzazione definisce due ruoli distinti che possono essere indicati con la seguente terminologia:

1. Classe più generale (dove sta la punta della freccia): classe padre, classe di base, o superclasse[7];
2. Classe più specifica: classe figlia, classe derivata[8].

Nell'esempio di figura 4.12, la classe `Employee` specializza la classe `Person` aggiungendo ai tre attributi presenti nella classe più generale (`First`, `Last`, e `SSN`), un nuovo attributo `Salary`. Analogamente, per la classe `Student`, che aggiunge agli attributi generali la matricola, `ID`.

[6] L'aggregazione è descritta per completezza, solitamente non viene usata per i modelli informativi concettuali

[7] Sebbene i termini siano spesso usati intercambiabilmente, a rigore il termine *parent* (padre) si riferisce ad una classe che è l'immediata generalizzazione, mentre i termini *base* e *super* si applicano anche a classi che sono generalizzazioni indirette (es. generalizzazione di generalizzazione)

[8] Sebbene i due termini siano spesso usati intercambiabilmente, a rigore il termine *child* (figlia) si riferisce ad una classe che è l'immediata specializzazione, mentre il terminie *derived* si applica anche a classi che sono specializzazioni indirette (es. specializzazione di specializzane)

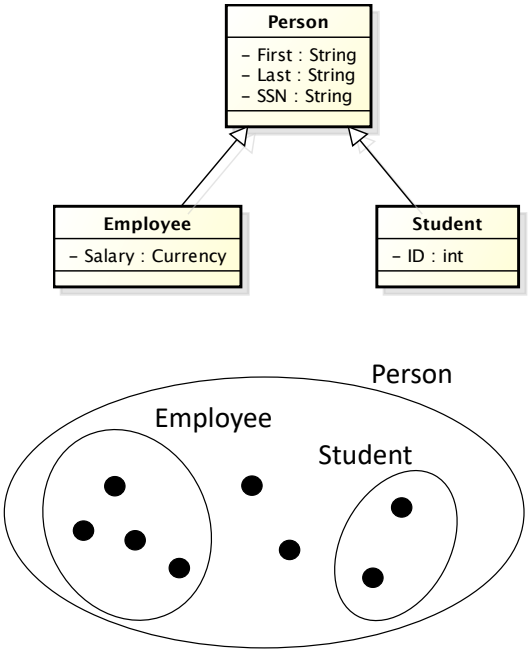


Figura 4.12: Rappresentazione della generalizzazione o specializzazione

La generalizzazione/specializzazione può essere interpretata, a livello insiemistico, come la relazione di sottoinsieme. Con riferimento alle tre classi di figura 4.12, L'insieme degli oggetti di classe `Student` è un sotto-insieme degli oggetti di classe `Person`, ed analogamente per `Employee`. Quindi tutti gli studenti sono *anche* persone, con in più un attributo (`ID`).

Se consideriamo l'interpretazione insiemistica, può essere utile dettagliare due parametri:

Totalità : una relazione di specializzazione si dice **totale**, se gli insiemi identificati dalle sotto-classi definiscono una partizione dell'insieme corrispondente alla classe di base. Ovvero ogni oggetto della classe di base appartiene ad (almeno) una delle classi derivate.

Nelle terminologia ad oggetti, questo significa che la classe di base è **astratta**, ovvero non esiste nessun oggetto che sia istanza di tale classe (e non anche di una classe derivata).

Il contrario di totale è **parziale**.

Esclusività : una specializzazione si dice (mutuamente) **esclusiva** se un oggetto, istanza di una classe derivata non può essere anche istanza di un'altra classe di base.

Nell'approccio ad oggetti, questa è l'interpretazione implicita.

In generale quando viene indicata una specializzazione, si intende che questa sia *totale* e *mutuamente esclusiva*. Se questo non è il caso, occorre indicarlo, ad esempio tramite un commento. In generale conviene esplicitare la tipologia della specializzazione ogni volta che possa essere interpretata in maniera ambigua

La situazione tipica è proprio quella in cui si ha una classe base, e più classi che derivate, come nell'esempio. In casi come questi, avere due classi distinte comporterebbe la duplicazione degli attributi comuni, mentre avere un'unica classe comporterebbe l'avere degli attributi che non sono utilizzati: uno studente non avrà mai un salario ed un impiegato non avrà mai una matricola. La generalizzazione ha lo scopo di evitare di costruire due

classi con n attributi in comune (quelli generici). Con questo costrutto si possono inserire solo una volta nella classe generica.

Sebbene sia teoricamente possibile, ciò che solitamente si cerca di evitare è la generalizzazione multipla ossia che una classe derivata che abbia due classi di base.

La generalizzazione comporta anche che se la classe di base partecipa ad un'associazione, allora anche le classi derivata "ereditano" la partecipazione. Perciò la generalizzazione è utile anche quando diverse classi partecipano alla stessa relazione, se esse sono generalizzate in una classe di base comune è possibile rappresentare un'unica associazione, rendendo il modello più semplice e leggibile.

4.6 Metodo di analisi

4.6.1 Procedura

La costruzione di un modello informativo concettuale solitamente parte da un testo che descrive il dominio del problema. Nei casi reali può trattarsi di documenti lunghi diverse pagine, per gli esercizi considerati in questo libro si considereranno enunciati lunghi circa mezza pagina.

Prima di analizzare il testo e costruire il modello è utile leggere il testo ed eventualmente manipolarlo per semplificare la fase di modellazione:

- Identificare il livello di astrazione giusto, deve essere utile per il nostro SI e deve permettere di catturare la complessità del sistema.
- Costruire un glossario di termini che verranno utilizzati nel modello identificando eventuali sinonimi.
- Identificare e rendere espliciti i riferimenti interni al testo.
- Semplificare la struttura delle frasi in modo da utilizzare una forma il più possibile standardizzata.

Successivamente a questa fase preliminare si può procedere all'analisi dettagliata del testo per costruire il modello concettuale

1. Identificare i concetti principali che costituiranno le classi del modello.

   Se un concetto ha delle proprietà significative o descrive dei tipi di oggetti con un'esistenza autonoma può essere rappresentato da una classe (es: Studente)

   I termini principali del glossario sono dei buoni candidati a diventare delle classi.

2. Identificare le informazioni caratterizzanti per i concetti principali che diventeranno gli attributi delle classi.

   Talvolta può sorgere il dubbio se un elemento abbia la dignità di classe oppure sia un semplice attributo. Una possibile discriminante è se tale elemento abbia una struttura semplice e non abbia proprietà rilevanti associate ad esso, in tal caso si tratta probabilmente di un attributo.

   In caso di dubbio è meglio, inizialmente, modellare l'elemento come una classe ed eventualmente in una fase successiva trasformarlo in un attributo.

3. Identificare le similarità ed i casi particolari che possono essere rappresentanti tramite generalizzazioni.

   Se uno o più concetti rappresentano casi particolari di un altro concetto, è conveniente rappresentarli con metodi di generalizzazione.

4. Scoprire i legami logici tra due o più concetti che saranno rappresentati tramite le associazioni tra classi ed assegnare un nome il più possibile significativo all'associazione.

   Quando un attributo risulta essere un elenco, molto probabilmente occorre rappresentarlo come una classe a se stante ed inserire un'associazione.

5. Definire le molteplicità delle associazioni per catturare i vincoli impliciti ed espliciti descritti nel modello.

I passi sopra delineati non si esauriscono in una semplice passata. Molto spesso devono essere ripetuti, raffinando e modificando il modello finché non si raggiunge un modello completo e corretto.

Ad esempio quando si trova una classe, tipicamente molto semplice con un unico attributo, che non altri legami se non un'associazione 1-1 obbligatorio con un'altra classe, è possibile fondere tale classe con quella collegata.

4.6.2 Qualità dei modelli concettuali

Le caratteristiche essenziali di un buon modello informativo concettuale sono:

Correttezza Nessun concetto è rappresentato in maniera errata.

Completezza Tutti i concetti e le informazioni rilevanti sono rappresentati.

Leggibilità È facilmente leggibile e comprensibile.

Minimalità Non sono presenti elementi (classi, attributi, associazioni) evitabili.

Un possibile quadro per la valutazione della qualità di un modello concettuale [4, 5] prevede tre tipi di qualità:

Qualità sintattica riguarda la correttezza sintattica.

Qualità semantica riguarda la validità e completezza del modello.

Qualità pragmatica riguarda la leggibilità e la minimalità del modello.

Queste categorie possono essere adottate per valutare modelli di tipo testuale, grafico o misto, quindi non solo per i modelli delle classi UML.

La correttezza sintattica richiede che il modello sia sintatticamente corretto, questo implica che tutti i costrutti siano allineati alla sintassi del linguaggio ed alle sue regole di uso. Minori sono gli errori e le deviazione dalle regole, maggiore è la correttezza sintattica.

La qualità semantica si focalizza su cosa manca nel modello che invece è presente nel dominio del problema che si intende modellare, come pure cosa è presente nel modello che non compare nel dominio. Si tratta di verificare la validità (tutti costrutti presenti nel modello sono corretti e rilevanti per il problema) e la completezza (tutti gli aspetti rilevanti del problema sono riportati nel modello). Non sempre è possibile raggiungere contemporaneamente la completa validità e completezza.

La qualità pragmatica riguarda la comprensione del modello dal punto di vista degli *stakholder*. In pratica si deve valutare se tra i vari modi alternativi di rappresentare gli aspetti del problema è stato scelto quello migliore, al fine di massimizzare la comprensione. La comprensione deve essere possibile dai diversi lettori del modello.

4.6.3 Pattern di modellazione

Nella modellazione concettuale è bene tener presente alcune tipiche soluzioni a problemi ricorrenti, che chiamiamo pattern.

Classe contenitore

Molto spesso nell'analisi è possibile identificare una cosiddetta *classe contenitore* che corrisponde al contesto operativo in cui opera il sistema informativo (e.g. l'azienda) se non addirittura al sistema informativo stesso.

Tale classe potrebbe contenere attributi che memorizzano informazioni che hanno una validità complessiva e trasversale a tutto il SI, quindi ragionevolmente non concentrabili in una classe. Per assurdo, se tale classe fosse realmente implementata nel modello, ad essa corrisponderebbe, per definizione, un unico oggetto. In aggiunta tale classe dovrebbe, logicamente, essere connessa a tutte le altre classi tramite un'aggregazione rendendo il modello inutilmente complicato e sostanzialmente illeggibile.

Inoltre è bene ricordare che il modello informativo concettuale è pensato per modellare le informazioni manipolate dal sistema informativo e non il sistema informativo stesso.

Tipicamente, per le ragioni sopra esposte tale classe contenitore **non viene** solitamente inclusa nel modello informativo concettuale.

L'eccezione si ha nel caso in cui il sistema informativo sia "multi-tenant," ossia debba poter supportare le attività di più azienda (e.g. nel caso di un ASP che fornisce stesso servizio per più aziende tipicamente in cloud).

Liste ed elenchi

Durante la modellazione, sovente si identificano degli attributi che sarebbero destinati a contenere una lista o un elenco (es. elenco dei numeri di telefono).

Gli attributi sono pensati per memorizzare dei dati elementari e non delle informazioni complesse. I tipi di dato tipicamente usati (elencati in tabella 42) indicano chiaramente questa natura atomica degli attributi.

Quando si riscontra un caso di questo tipo è opportuno modellarlo con una classe aggiuntiva che rappresenta il dato elementare e collegarlo tramite un'associazione molti-a-1 alla classe principale che contiene l'elenco.

Il vantaggio di tale soluzione è che non si snatura la caratteristica atomica degli attributi e si esplicita la molteplicità del dato tramite l'associazione. Lo svantaggio è che il modello risulta più complesso perchè comprenderà una classe ed un'associazione in più.

Associazione istanza

Può accadere che l'analisi di un caso riveli come importanti dei concetti correlati tra loro, ad esempio un'idea astratta e la sua concretizzazione. Spesso si tratta dello stesso termine usato con due significati oppure due termini che possono essere sinonimi. Ad esempio:

- il prodotto, inteso come la descrizione generale (e.g. Razzo ACME)
- il prodotto, inteso come lo specifico esemplare di razzo utilizzato da Wile E. Coyote in un dato episodio.

In tali casi è importante distinguere chiaramente i due concetti; la differenza emerge facilmente se ragionando sulle numerosità degli oggetti di ciascuna di queste classi. Occorre poi capire se è necessario rappresentare entrambe le classi oppure solo una delle due.

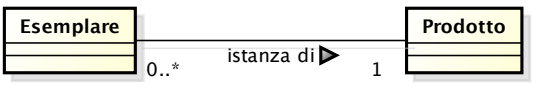


Figura 4.13: Istanza di

Se entrambe le classi sono rappresentate nel modello esse saranno legate da un'associazione tipo `esemplare_di` o `istanza_di`.

In tal caso è bene rimarcare che il legame tra le due classi non può essere rappresentanto dalla generalizzazione; basta osservare che l'idea di sottoinsieme non ha alcun senso.

Classificazione

Quando durante l'analisi emerge la presenza di una classificazione, con categorie e sottocategorie, non sempre il modello più appropriato è quello che usa la generalizzazione.

Si usa la generalizzazione se le classi derivate hanno delle caratteristiche distintive (es. attributi particolari o associazioni in cui sono coinvolte) altrimenti (come indicato nell'esempio di figura 4.14) è sufficiente usare un attributo (es. di tipo String con alcuni valori possibili).

Un unico attributo è meglio di più attributi booleani (uno per ogni categoria) perché in questo modo si evita di dover aggiungere esplicitamente un insieme di vincoli di mutua esclusione.

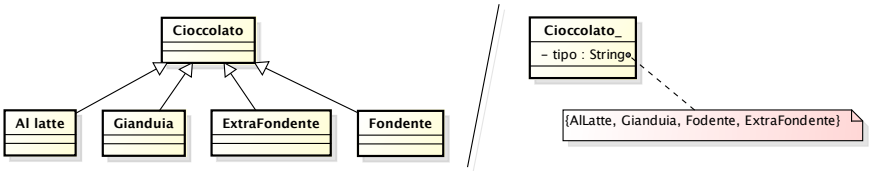


Figura 4.14: Classificazione e generalizzazione

In presenza di una classificazione descritta esplicitamente tramite una generalizzazione, è importante ricorda che gli oggetti (istanze) delle classi sono associati ad una classe e non possono cambiare la classe. Ad esempio (figura 4.15) nel modellare una squadra sportiva composta da giocatori, uno dei quali è il capitano che può variare nel corso della partita o della stagione, non posso utilizzare una genralizzazione (soluzione a sinistra nella figura). Questo perchè un oggetto inizialmente della classe capitano dovrebbe poter diventare un semplice giocatore, e viceversa: questo è impossibile. Una possibile soluzione è quella di utilizzare un attributo che indichi se il giocatore è il capitano (soluzione a destra).

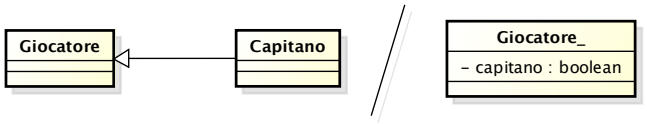


Figura 4.15: Persistenza della generalizzazione

Obbligatorietà delle associazioni

Nella definizione della molteplicità delle associazioni, è solitamente facile identificare quelle massime, mentre possono esistere dubbi su quelle minime.

La molteplicità minima di una classe (la cifra prima dei `..` nell'etichetta vicino alla classe) può avere due valori:

0 (zero) partecipazione opzionale, significa che gli oggetti della classe opposta possono non essere legata ad alcuna istanza della classe data, ovvero gli oggetti della classe opposta possono esistere autonomamente, hanno un significato chiaro anche senza essere legati ad un oggetto delle classe data. La classe opposta è una classe autonoma o principale. Ad esempio in figura 4.16, `Fornitore` è una classe autonoma rispetto alla classe `Prodotto`.

1 (uno) partecipazione obbligatoria, significa che gli oggetti della classe opposta devono essere legati ad un oggetto della classe data, ovvero gli oggetti della classe opposta non possono esistere autonomamente, non hanno un significato compiuto se non legati ad un oggetto della classe data. La classe opposta è una classe dipendente, debole o secondaria. Ad esempio in figura 4.16, `Ordine` è una classe secondaria rispetto alla classe `Prodotto`.

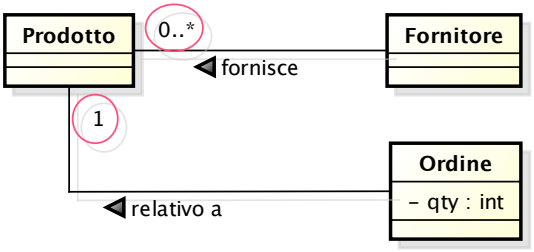


Figura 4.16: Obbligatorietà delle associazioni

Riuso delle classi deboli

Le classi deboli, come la classi come `PriceTag` nell'esempio mostrato figura 4.16, sono classi che rappresentano dei concetti non centrali nel contesto del problema e le cui occorrenze non avrebbero ragione di esistere senza un'occorrenza della classe forte corrispondente. Ad esempio: l'etichetta del prezzo non avrebbe ragione di esistere senza il prodotto corrispondente.

Sebbene in alcune occasioni sia sensato pensare di collegare tali classi deboli a delle classi forti con una molteplicità `1..*` questo se da un lato comporta una minore duplicazione dei dati, può comportare una maggiore complessità nell'aggiornamento.

Nell'esempio in figura invece di avere una molteplicità `1` (evidenziata dal cerchio rosso) si potrebbe avere la molteplicità `1..*`. In tale contesto, consideriamo la situazione in cui dopo aver assegnato lo stesso prezzo ad un insieme di prodotti si decide di correggerlo solo per alcuni tra essi, cosa succede nei due casi:

- molteplicità `1`: si modifica il valore dell'attributo prezzo nelle etichette relativi ai prodotti considerati;
- molteplicità `1..*`: si deve scollegare l'etichetta condivisa dai prodotti considerati, si devono creare una o più etichette con in nuovi prezzi e collegarle ai prodotti.

È evidente da questo semplice esempio che la complessità dell'aggiornamento è signficativamente maggiore se la molteplicità è `1..*` rispetto al caso più semplice di molteplicità `1`. Per tale ragione, solitamente la raccomandazione è di collegare una classe debole ad una classe forte con molteplicità `1`.

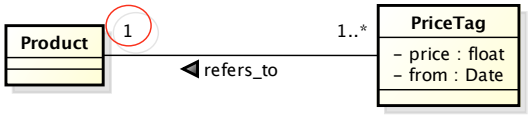


Figura 4.17: Molteplicità classi deboli

Storico di un'associazione

Spesso capita in un sistema informativo di dover memorizzare la storia di una associazione, ad esempio lo storico dell'associazione `lavora` tra una `Persona` e una `Azienda`.

In tale caso, non è sufficiente una semplice associazione, perchè occorre riportare almeno anche una data di inizio. Perciò è opportuno trasformare l'associazione semplice in una classe di associazione.

Questa trasformazione però non è sufficiente: se consideriamo una persona ed un'azienda (ovvero un oggetto persona ed un'oggetto azienda) è possibile che esse siano legate da una legame di lavoro più volte nello storico (ovvero che i due oggetti siano legati da più collegamenti, istanze della stessa associazione). Questa configurazione non corrisponde a un'associazione o ad una classe di associazione, si pensi all'interpretazione insiemistica dell'associazione come sottoinsieme del prodotto cartesiano: un collegamento tra due oggetti o c'è o non c'è, non è contemplata la presenza multipla.

Per rappresentare adeguatamente lo storico di un'associazione è necessario introdurre una classe intermedia che prenda il posto dell'associazione stessa (o della classe di associazione) e che sia legata alle due classi di partenza (come mostrato in figura 4.10). Tale classe avrà come attributi le informazioni necessarie a tenere traccia della storia (es. la data di inizio).

Associazione ricorsiva e gerarchia

Quando l'analisi indica che gli oggetti di una classe possono essere legati ad altri oggetti, formando una struttura gerarchica ad albero, i collegamenti sono istanze di un'associazione ricorsiva.

Ad esempio, la struttura gearchica dei folder in un file system può essere modellata come in figura 4.18. Occorre prestare attenzione alle molteplicità

Osservando le molteplicità massime: un figlio può avere al più un padre ed un padre può avere più figli.

Osservando le molteplicità minime: un figlio può avere o meno un padre (il folder radice, non avrà alcun padre mentre gli altri avranno sempre un padre) ed un padre può avere eventualmente nessun figlio (è il caso dei folder foglie, che non contengono altri folder).

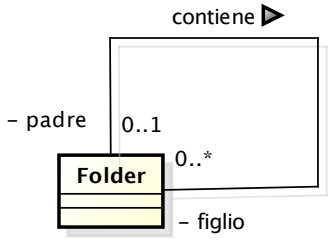


Figura 4.18: Gearchia tramite associazione ricorsiva

4.7 Esempio di analisi

Vogliamo sviluppare un sistema informativo per un'azienda che organizza corsi di formazione. Per fare questo dobbiamo gestire i dati relativi ai partecipanti ed ai docenti.

Per ciascun partecipante ai corsi (circa 5000), identificato da un codice, vogliamo memorizzare il codice fiscale, il cognome, l'età, il genere, il luogo di nascita, il nome del datore di lavoro, l'indirizzo, il numero di telefono, i datori di lavoro precedenti (ed il periodo di lavoro), i corsi frequentati (ci sono circa 200 corsi in catalogo) e la valutazione finale per ciascun corso.

Vogliamo anche rappresentare i seminari che ciascuno studente frequenta al momento e, per ogni giorno, le aule e gli orari delle lezioni. Ogni corso ha un codice ed un titolo e può essere erogato un numero indefinito di volte. Ogni volta che il corso viene erogato lo chiameremo 'edizione' del corso. Per ogni edizione, riportiamo la data di inizio, quella di fine ed il numero di partecipanti.

Se un partecipante è un professionista autonomo, vogliamo sapere il suo settore di competenza, e il suo titolo. Per chi è impiegato da un'azienda memorizziamo il livello e la posizione assunta.

Per ciascun istruttore (circa 300) mostriamo il cognome, l'età, il luogo di nascita, l'edizione dei corsi tenuta, quelle tenute in passato ed i corsi per cui è qualificato all'insegnamento. Anche tutti i numeri di telefono degli istruttori sono memorizzati. Gli istruttori possono essere dipendenti oppure lavoratori autonomi.

Una volta identificati i concetti occorre definire le classi, in questo caso ne vengono identificate 4: Corso, Istruttore, Allievo, Datore di lavoro, definite come specificato in tabella 43.

Tabella 43: Glossario

Termine	Descrizione	Sinonimi	Legami
Corso	Corsi offerti a catalogo. Può avere diverse edizioni.	Seminario	Istruttore, Studente
Studente	Partecipante a un corso. Può essere dipendente o autonomo.	Participante	Corso, Azienda
Istruttore	Docente del corso. Può essere dipendente o autonomo.	Docente	Course
Azienda	Aziende in cui lo studente è o era impiegato.	Datore	Studente

In prima approssimazione possiamo considerare i termini principali che abbiamo identificato nel glossario come le classi della prima bozza di modello concettuale:

A questo punto occorre analizzare il testo frase per frase evidenziando gli attributi da inserire in ogni classe. Le seguenti considerazioni portano al risultato mostrato in figura 4.19:

Corso il testo "*Ogni corso ha un codice ed un titolo*" è chiaro relativamente a quali sono gli attributi essenziali. Per quanto riguarda il tipo; il `codice` potrebbe essere numerico o testuale, in caso di ambiguità si sceglie il tipo più generale, ovvero `String` in questo caso; invece non esiste ambiguità per il tipo di `titolo`.

Un altro frammento di descrizione del corso ne testo riporta "Vogliamo anche rappresentare i seminari che ciascuno studente frequenta al momento e, per ogni giorno, le aule e gli orari delle lezioni."

Per rappresentare questo aspetto dei corsi, è possibile introdurre un'altra classe, `Lezione` che avrà come attributi `giorno`, `ora` e `aula`. Un'altra possibile soluzione, più fedele alla lettera ma più complessa, consiste nel definire una classe `Giorno` collegata alla classe `Edizione` tramite una classe di associazione che includa gli attributi `ora` e `aula`.

Se esaminiamo il resto della descrizione del corso troviamo la seguente frase "*Ogni volta che il corso viene erogato lo chiameremo 'edizione' del corso. Per ogni edizione,*

riportiamo la data di inizio, quella di fine ed il numero di partecipanti." che può indicare a prima vista due modelli diversi (ma uno sono sarà quello corretto):

1. occorre aggiungere gli attributi `edizione`, `dataInizio` e `dataFine` alla classe `Corso`.

   Tale soluzione comporta una significativa duplicazione, infatti in tutti gli oggetti (righe della tabella nel db) che rappresentano edizioni diverse dello stesso corso riporteranno tutti, ripetute, le stesse informazioni comuni sul corso (codice e titolo).

   Inoltre per rappresentare un corso a catalogo per il quale non è ancora state erogate alcuna edizione non saprei quali valori assegnare ai tre nuovi attributi, dovrei probabilmente usare dei valori convenzionali.

2. occorre introdurre una nuova classe `Edizione` con due attributi `dataInizio` e `dataFine`

   Questa soluzione elimina la duplicazione di quella precedente. In particolare esisterà un oggetto (riga) per ciascun corso indipendentemente dal numero di edizioni.

Un'ulteriore considerazione riguarda quello che è un potenziale candidato a diventare un altro attributo, ovvero il *numero di partecipanti*. Questo attributo corrisponde al numero di oggetti collegato ad un determinato corso. Tale informazione non deve essere indicata esplicitamente perché viene memorizzata in maniera implicita dal database che ospita i dati del sistema informativo.

Inserire esplicitamente questi attributi di *cardinalità* oltre ad essere inutile comporta una complicazione del sistema informativo: infatti se memorizzassimo il numero di partecipanti in un attributo, ogni volta che aggiungessimo un partecipante ad un'edizione dovremmo anche aggiornare il conteggio.

Studente : il testo suggerisce alcuni attributi: "*identificato da un codice, vogliamo memorizzare il codice fiscale, il cognome, l'età, il genere, il luogo di nascita, il nome del datore di lavoro, l'indirizzo, il numero di telefono*"

Una considerazione a parte merita l'attributo `età`. Questo è un esempio di attributo che varia col tempo: se inserisco l'età oggi, tra un anno il valore non sarà più valido ma dovrà essere aggiornato, in realtà dovrebbe essere aggiornato il giorno del compleanno dello studente. Per questo motivo è molto meglio inserire come attributo `dataNascita` che non varia nel tempo e dalla quale è facile calcolare l'età, conoscendo la data odierna.

Un altro potenziale attributo è il `nome del datore di lavoro`: avendo scelto di rappresentare il datore di lavoro tramite la classe `Azienda`, questa informazione relativa allo studente verrà rappresentata tramite un'associazione tra lo `Studente` e `Azienda`. Inserire il nome come attributo stringa dello studente sarebbe un errore grave per due motivi: (1) introduce una duplicazione perciò per mantenere le informazioni consistenti tra loro il sistema informativo dovrà aver cura di mantenere i nomi dei datori (definiti per gli studenti) allineati ai nomi delle aziende; (2) osservando il modello non è possibile cogliere il legame tra `Studente` e `Azienda` perchè risulta implicito (tramite l'attributo) e non esplicito (tramite un'associazione).

Altre informazioni sullo studente partecipante ai corsi arrivano da una frase successiva: " *Se un partecipante è un professionista autonomo, vogliamo sapere il suo settore di competenza, e il suo titolo. Per chi è impiegato da un'azienda memorizziamo il livello e la posizione assunta.* ".

Gli attributi da aggiungere dipendono dal tipo di partecipante (lavoratore autonomo o dipendente). In casi come questi la rappresentazione ideale è quella che utilizza la generalizzazione: una classe di base `Studente` e due classi derivate, `Autonomo` e `Dipendente`. Ciascuna classe avrà gli attributi di pertinenza.

Istruttore le informazioni rilevanti sono riportate nell'ultima frase del testo "Per ciascun istruttore (circa 300) mostriamo il cognome, l'età, il luogo di nascita, l'edizione dei corsi tenuta, quelle tenute in passato ed i corsi per cui è qualificato all'insegnamento. Anche tutti i numeri di telefono degli istruttori sono memorizzati.

Relativamente all'età vale lo stesso discorso fatto sopra per gli studenti.

La frase relativi ai numeri di telefono è potenzialmente ambigua: potrebbe significare che i numeri di telefono di tutti gli istruttori sono rilevanti oppure che per ogni istruttore sono rilevanti tutti i numeri di telefono; quest'ultima interpretazione implica che i numeri di telefono per ogni istruttore possano essere molti. Per semplicità assumiamo che la prima interpretazione sia quella corretta. Se dovessimo modellare la seconda opzione, non potendo inserire in un unico attributo più valori, dovremmo introdurre una nuova classe `Telefono` legata a `Istruttore`.

Gli aspetti sottolineati nella frase riportata qui sopra, fanno riferimento ad altre classi e quindi sono candidati ad essere modellati con delle associazioni piuttosto che con degli attributi.

Un altro aspetto da considerare è quello riportato nell'ultimo periodo "Gli istruttori possono essere dipendenti oppure lavoratori autonomi.". Questa caratteristica degli istruttori può essere modellata in due modi:

1. tramite una generalizzazione, introducendo due classi derivate `Dipendente` e `Autonomo`, analogamente a quanto fatto per gli studenti. Tuttavia in questo caso le due classi derivate non avrebbero alcun attributo proprio.

   Si tratta di un uso della generalizzazione con uno scopo puramente tassonomico. Sebbene non errato, quest'uso porta a modelli più complessi (più classi e più relazioni) e quindi meno comprensibili.

2. tramite un attributo `posizione` che indichi la tipologia di posizione dell'istruttore.

   Questa soluzione è decisamente più semplice, permette di modellare le stesse informazioni, e comporta un modello più semplice.

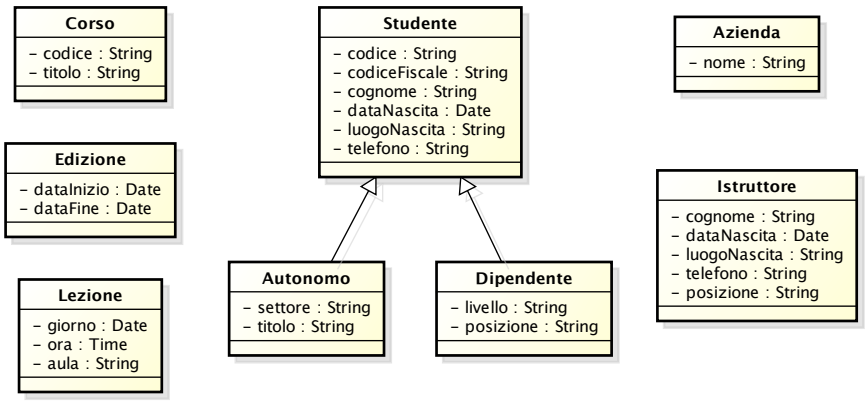


Figura 4.19: Secondo raffinamento del modello

Il passo successivo è quello di identificare il legami logici tra le classe e rappresentarli con delle associazioni. Le seguenti considerazioni portano al risultato mostrato in figura 4.20:

Edizione-Corso l'associazione `riferita_a` lega l'`Edizione` al `Corso`. L'edizione è sempre riferita ad un corso quindi la molteplicità dal lato del corso sarà `1..1` o in maniera più sintetica, solamente `1`. Viceversa un corso potrà avere molte edizioni, ma al limite

potrebbe essere appena stato messo a catalogo senza ancora alcuna edizione, quindi la molteplicità sarà `0..*`. Si tratta di una relazione 1-a-molti con la partecipazione obbligatoria da parte dell'edizione.

Istruttore-Edizione e **Istruttore-Corso** sono rispettivamente delle associazioni 1-a-molti e molti-a-molti che chiamiamo `insegna` e `qualificato_per`.

Anche se non è scritto nel testo, è ragionevole aspettarsi che un istruttore possa insegnare solo in edizioni di corsi per cui è qualificato. Questo vincolo non è esprimibile direttamente con un costrutto del diagramma delle classi UML. Se tale vincolo è importante, una possibilità è quella di inserire un commento nel modello che lo riporti.

Studente-Edizione nel testo troviamo queste informazioni: "*Per ciascun partecipante [...] vogliamo memorizzare [...] i corsi frequentati [...] e la valutazione finale per ciascun corso*".

Avendo introdotto in un passo di analisi precedente la classe `Edizione` per rappresentare più accuratamente le informazioni di `Corso`, è bene capire il termine *corso* nel frammento precedente a quale delle due classi si riferisca.

È facile capire che il termine si riferisce alla classe `Edizione`, perciò sarà necessario introdurre una associazione `frequenta`.

Occorre, poi, decidere come rappresentare la valutazione finale. Essa non può essere un attributo nè di `Studente`, nè di `Edizione`. Una possibilità è quella di definire una classe di associazione per l'associazione `frequenta` che includa l'attributo `valutazione`.

Un altro aspetto da considerare è che si vuole "*rappresentare i seminari che ciascuno studente frequenta al momento*". Questa informazione può essere desunta a partire dalle edizioni frequentate dallo studente, verificando se la data odierna si trova tra la data di inizio e quella di fine dell'edizione.

Studente-Azienda il testo ci dice che "*Per ciascun partecipante [...] vogliamo memorizzare [...] il nome del datore di lavoro [...], i datori di lavoro precedenti (ed il periodo di lavoro)*".

Per memorizzare il datore di lavoro corrente è sufficiente un'associazione molti-a-uno: uno studente lavora presso un'azienda e un'azienda può ospitare molti studenti.

Tuttavia, per memorizzare i datori precedenti, l'associazione non è sufficiente: è possibile che uno stesso studente abbia lavorato, in periodi distinti, presso la stessa azienda: in questo caso avremmo più istanze della stessa associazione tra la stessa coppia di oggetti

Perciò è necessario introdurre una nuova classe intermedia, `Lavoro` che permetta di tracciare i periodi di lavoro dello studente presso le aziende. La classe avrà come attributi la `dataInizio` e la `dataFine`.

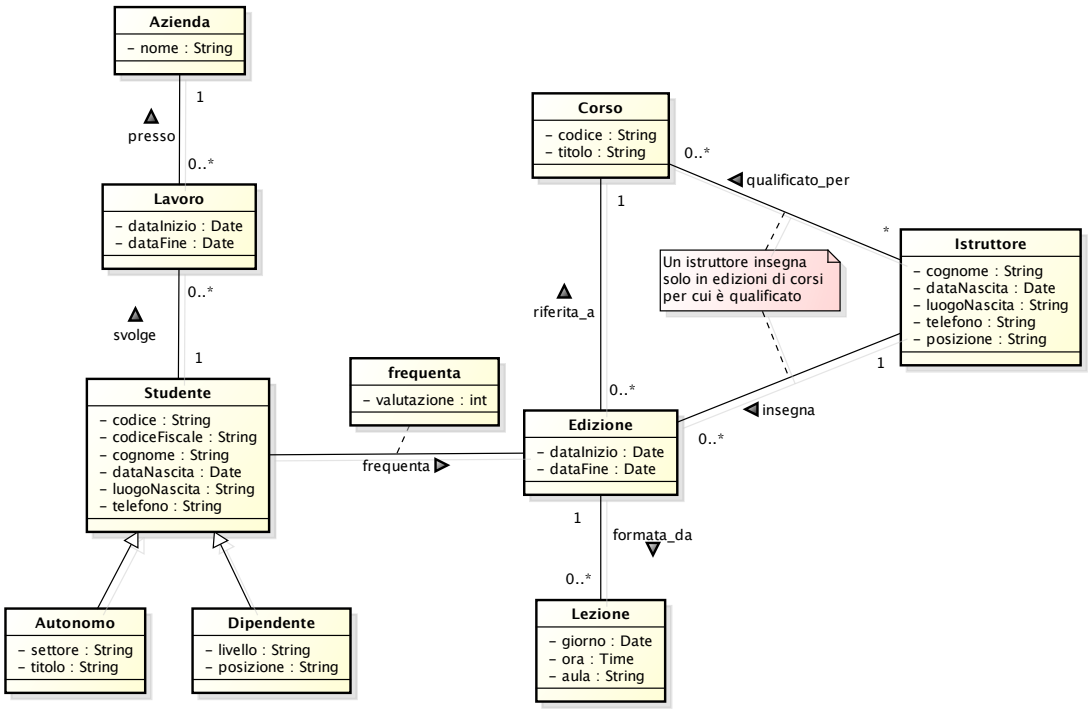


Figura 4.20: Terzo raffinamento del modello

BIBLIOGRAFIA

[1] M. Fowler, *UML Distilled: A Brief Guide to the Standard Object Modeling Language, 3rd edition*. Addison-Wesley Professional, 2003.

[2] P. Chen, "The entity-relationship model: toward a unified view of data," *ACM Transactions on Database Systems*, vol. 1, pp. 9–36, 1976.

[3] OMG, *Business Process Modeling Notation (BPMN) Version 1.0. OMG Final Adopted Specification.* Object Management Group, 2006.

[4] O. Lindland, G. Sindre, and A. Solvberg, "Understanding quality in conceptual modeling," *IEEE Software*, vol. 11, no. 2, pp. 42–49, 1994.

[5] N. Bolloju and F. Leung, "Assisting novice analysts in developing quality conceptual models with uml," *Communications of the ACM*, vol. 49, p. 108, 112 2006.

Licenza e Colophon

Questo volume è stato redatto con il sistema di composizione LaTeX[9] utilizzando il modello di stile `memoir`[10].

Il contenuto del testo è rilasciato con la licenza Creative Commons Attribuzione - Non commerciale - Condividi allo stesso modo 2.5 Italia (CC BY-NC-SA 2.5)[11].

[9] `http://www.latex-project.org/`
[10] `http://www.ctan.org/tex-archive/macros/latex/contrib/memoir/`
[11] `http://creativecommons.org/licenses/by-nc-sa/2.5/it/`