Appunti di Progettazione del Software

Canale 2 · Prof. Massimo Mecella

Capitolo 11: Analisi e diagramma delle classi

Autore: Tommaso C.

Appunti tratti dalle lezioni in aula e dal materiale del corso

---

Argomenti: La fase di analisi · Perché modellare · Schema concettuale
Il linguaggio UML · Oggetti e classi · Attributi
Relazioni e associazioni

Indice

1 La fase di analisi **1**

2 Perché si costruisce un modello **1**

3 Lo schema concettuale **2**

4 Il linguaggio UML **3**

5 Oggetti e classi **3**

6 Attributi **5**

7 Relazioni e associazioni **6**

8 Esercizi **8**

9 Soluzioni **10**

1 La fase di analisi

Definizione – Analisi

L'**analisi** è la fase del ciclo di vita del software che

- riceve in ingresso i **requisiti** raccolti;
- produce lo **schema concettuale** dell'applicazione, detto anche **modello di analisi**.

Un **requisito** dice che cosa il sistema deve fare, per esempio memorizzare nome, cognome e indirizzo dei clienti, o servire mille utenti insieme. Nel ciclo di vita (Capitolo 1, Sezione 3) l'analisi segue la raccolta dei requisiti e precede il progetto.

Lo schema concettuale deve essere:

- **completo**, **preciso** e **rigoroso**: ha una semantica definita, e un simbolo sbagliato è un errore;
- **leggibile**, anche da chi non è tecnico, come il committente;
- **indipendente dal linguaggio di programmazione**, ma traducibile in un programma.

Lo schema descrive *che cosa* serve all'applicazione, cioè i concetti e le informazioni su cui lavora, non *come* lo si realizza. Per questo si chiama concettuale: lo stesso schema si può realizzare in Java, in C o in un altro linguaggio.

2 Perché si costruisce un modello

Ogni ingegneria costruisce un modello prima del prodotto: un modello in scala, un plastico, un disegno. Il software è intangibile, e il suo modello è un diagramma, con una semantica precisa.

Formalizzare toglie le ambiguità. Il committente descrive i requisiti in linguaggio naturale, che è ambiguo. Modellare obbliga a dare a ogni requisito una forma precisa. Così emergono le incompletezze, le inconsistenze e le ambiguità: un requisito che non si sa rappresentare, o che si può rappresentare in due modi diversi.

Le domande vanno al committente. L'analista non risolve da solo un'ambiguità: la porta al committente, che risponde, e il modello si corregge. L'analisi è un dialogo continuo, e finisce quando il committente riconosce nel modello esattamente ciò che vuole.

Proposizione – A che cosa serve lo schema concettuale

1. Verifica i requisiti con l'utente finale.
2. È il riferimento di tutte le fasi successive: il software è corretto se è coerente con lo schema.
3. Guida le decisioni sulla struttura e sulla modularizzazione del software.
4. È la specifica delle funzionalità da realizzare, per tutto il gruppo di sviluppo.

3 Lo schema concettuale

Definizione – Schema concettuale

Nella metodologia del corso, basata su UML, lo schema concettuale è formato da:

1. il **diagramma delle classi e degli oggetti**: le classi dell'applicazione e le loro proprietà, e gli oggetti particolarmente significativi;
2. il **diagramma delle attività**: le funzionalità che il sistema realizza, come processi;
3. il **diagramma degli stati e delle transizioni**: per le classi significative, il ciclo di vita tipico delle istanze;
4. i **documenti di specifica**: le condizioni precise che i programmi devono soddisfare.

Una classe, qui, non è una classe Java: è un tipo di informazione che l'applicazione deve trattare, come lo studente o l'esame.

I tre diagrammi guardano il sistema da tre punti di vista:

Diagramma	Vista	In un sistema universitario
delle classi	statica: una fotografia delle informazioni	esistono studenti ed esami
delle attività	dinamica, su più oggetti: come si passa da una fotografia alla successiva	l'immatricolazione aggiunge uno studente; prenotazione e svolgimento di un esame
degli stati	dinamica, su un solo oggetto: gli stati che attraversa	uno studente è in immatricolazione, poi immatricolato, iscritto al primo anno, . . . , laureando, laureato

Le attività cambiano lo stato di più oggetti insieme: l'immatricolazione cambia lo stato dello studente, ma anche quello di altri oggetti dell'università. Per capire il sistema servono quindi tutte e tre le viste.

Le specifiche completano i diagrammi. Non tutto si disegna. Che uno studente diventi immatricolato solo dopo aver pagato le tasse non si esprime nel diagramma degli stati: lo dice la specifica, un testo in italiano scritto secondo regole precise.

4 Il linguaggio UML

Fino agli anni Novanta le metodologie di analisi erano **orientate alle funzioni**, con diagrammi funzionali, di flusso di controllo e di flusso di dati. Quelle attuali sono **orientate agli oggetti**.

Definizione – UML

UML (*Unified Modeling Language*) è un linguaggio grafico per costruire modelli di sistemi software. Nasce nel 1994 dall'unificazione delle metodologie orientate agli oggetti di Booch, Rumbaugh (OMT) e Jacobson (OOSE). La versione 2.0 è del 2005.

UML è un *linguaggio*, non una metodologia. Il linguaggio fissa i simboli e il loro significato; la metodologia dice come usarli per fare l'analisi. UML ha molti tipi di diagrammi; il corso ne usa tre, il diagramma delle classi e degli oggetti, il diagramma delle attività e il diagramma degli stati e delle transizioni, e di questi solo i costrutti che servono alla sua metodologia.

Il diagramma delle classi, in analisi, descrive soprattutto classi: interessa che il sistema possa memorizzare studenti e professori, non quali studenti contenga oggi. Gli oggetti servono per elementi singoli particolarmente significativi.

UML è nato per essere disegnato con carta e matita. Per questo il significato sta nella forma dei segni: conta se un tratto è pieno o tratteggiato, se la punta di una freccia è piena o vuota, non lo spessore né il colore.

5 Oggetti e classi

Definizione – Oggetto

Un **oggetto** UML rappresenta un elemento del dominio di analisi. Un oggetto:

- ha vita propria;
- è identificato in modo univoco dal suo **identificatore**;
- è istanza di una classe, la sua **classe più specifica**.

Si disegna come un rettangolo con `id : Classe` sottolineato.

Con la generalizzazione, cioè l'ereditarietà vista in analisi (Capitolo 1, Sezione 12), un oggetto è istanza anche delle classi più generali: uno studente è anche una persona. Fra queste classi, la più specifica è una sola.

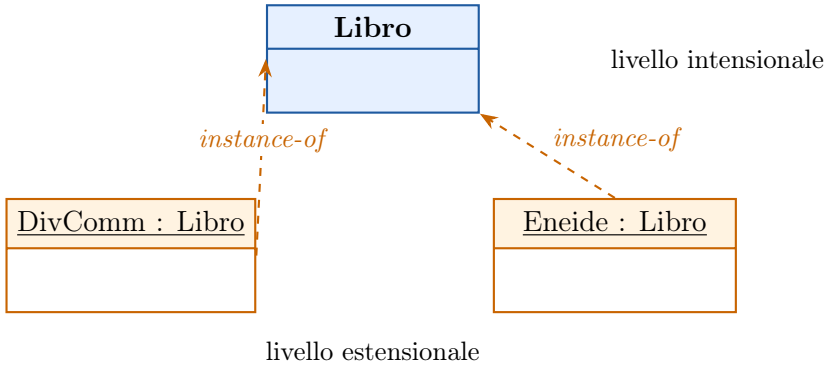
Definizione – Classe

Una **classe** modella un insieme di oggetti omogenei, le sue **istanze**. È descritta da un **nome** e dalle **proprietà** comuni a tutte le istanze. Si disegna come un rettangolo diviso in due: in alto il nome, non sottolineato, in basso le proprietà.

La classe nasce per astrazione. Gli oggetti di un dominio hanno le stesse caratteristiche, con valori diversi: tutti i libri hanno un titolo, ciascuno il suo. La classe tiene le caratteristiche e lascia da parte i valori.

Definizione – Instance-of

Fra un oggetto e la sua classe si traccia l'arco **instance-of**, tratteggiato con la punta piena, dall'oggetto verso la classe.



La classe Libro e due sue istanze. Le classi formano il livello intensionale, gli oggetti il livello estensionale.

Definizione – Livello intensionale ed estensionale

- Il **livello intensionale** è quello delle classi, cioè dei concetti.
- Il **livello estensionale** è quello degli oggetti, cioè delle istanze concrete.

Il livello intensionale determina come è strutturato quello estensionale.

Osservazione – Un paragone con Java

In un programma Java, il livello intensionale è il codice, con le classi scritte nei file; il livello estensionale sono gli oggetti che vivono nello stack e nello heap durante l'esecuzione (Capitolo 2, Sezione 9). Il paragone vale solo come intuizione: UML non dipende da Java, e uno schema si può realizzare anche in C.

6 Attributi

Definizione – Attributo

Un **attributo** modella una proprietà locale di una classe, valida per tutte le istanze e indipendente da altri oggetti. Ha un **nome** e un **tipo**, e si scrive nella parte bassa del rettangolo della classe come

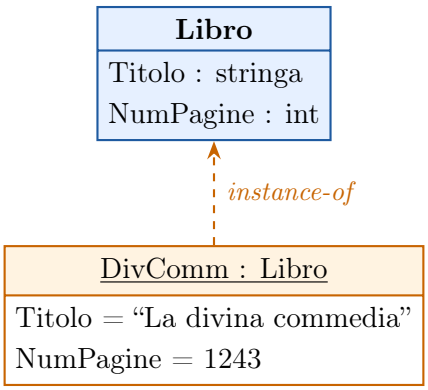
`Nome : tipo`

I tipi sono quelli comuni ai linguaggi di programmazione, come `stringa`, `int`, `boolean`, `float`. Formalmente, un attributo A di tipo T della classe C è una funzione che associa a ogni istanza di C un valore di tipo T.

Proposizione – Attributi degli oggetti

1. Se la classe C ha un attributo A di tipo T, ogni istanza di C ha l'attributo A, con un valore di tipo T.
2. Un oggetto non può avere un valore per un attributo che la sua classe non definisce.

Nell'oggetto, il valore si scrive come `Nome = valore`:



Una classe con due attributi e una sua istanza, con un valore per ciascun attributo.

Proposizione – Identificatori

Due oggetti con identificatori diversi sono distinti, anche se hanno gli stessi valori per tutti gli attributi. Due oggetti diversi devono avere identificatori diversi.

Due oggetti `DivComm : Libro` nello stesso diagramma sono quindi un errore. Chi li disegna, però, forse voleva dire un'altra cosa: che la biblioteca possiede due copie della Divina Commedia. La classe `Libro` non lo può dire, perché rappresenta l'opera, e un'opera è una sola. Le copie fisiche sono un altro concetto, e richiedono un'altra classe, per esempio `CopiaLibro`, con un'istanza per ogni copia.

Il caso mostra a che cosa serve modellare. Una regola del linguaggio vieta il diagramma, e la domanda che ne nasce chiarisce che cosa il sistema deve rappresentare.

Osservazione – Classi, oggetti e valori

- Classi diverse, se non sono legate da una generalizzazione, non hanno istanze comuni: modellano insiemi disgiunti.
- Un oggetto è istanza di una classe e ha vita propria.
- Un valore è un elemento di un tipo, e ha senso solo se è associato a un oggetto tramite un attributo.

Per controllare un diagramma si guarda prima la sintassi: un diagramma scorretto non si può discutere con il committente (Esercizio 1).

7 Relazioni e associazioni

Definizione – Relazione matematica

Dati due insiemi S_1 e S_2, il **prodotto cartesiano** $S_1 \times S_2$ è l'insieme di tutte le coppie $\langle x, y\rangle$ con $x \in S_1$ e $y \in S_2$. Una **relazione** fra S_1 e S_2 è un sottoinsieme del prodotto cartesiano:

$$R \subseteq S_1 \times S_2$$

Per esempio, con $S_1 = \{1, 2, 3\}$ e $S_2 = \{a, b\}$:

$$S_1 \times S_2 = \{\langle 1, a\rangle, \langle 1, b\rangle, \langle 2, a\rangle, \langle 2, b\rangle, \langle 3, a\rangle, \langle 3, b\rangle\},$$

e $R = \{\langle 1, a\rangle, \langle 1, b\rangle, \langle 2, a\rangle\}$ è una relazione. Una relazione seleziona, fra tutte le coppie, quelle che le interessano.

Esempio – Esami sostenuti

Con gli insiemi $\mathit{studente} = \{\text{Paolo}, \text{Anna}, \text{Lucia}\}$ e $\mathit{esame} = \{\text{Analisi}, \text{Geometria}\}$, la relazione

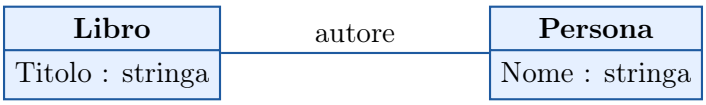
$$\mathit{sostenuto} = \{\langle \text{Paolo}, \text{Analisi}\rangle, \langle \text{Anna}, \text{Analisi}\rangle, \langle \text{Anna}, \text{Geometria}\rangle\}$$

contiene le coppie $\langle x, y\rangle$ in cui lo studente x ha sostenuto l'esame y. Lucia non compare: non ha sostenuto esami.

Una relazione si può dare in due modi. Per **estensione**, si elencano le coppie. Con una **regola generativa**, si dice quando una coppia appartiene alla relazione: per *sostenuto*, quando nel mondo reale lo studente ha superato l'esame. Il nome della relazione suggerisce la regola.

Definizione – Associazione

Un'**associazione** fra le classi C_1 e C_2 modella una relazione matematica fra l'insieme delle istanze di C_1 e quello delle istanze di C_2. Si disegna come una linea a tratto pieno fra le due classi, con un nome, che dà la regola generativa delle coppie.



L'associazione autore*: contiene la coppia (Divina Commedia, Dante) perché Dante è l'autore della Divina Commedia.*

Osservazione – Proprietà di due classi

Gli attributi modellano proprietà locali di una classe. Un'associazione modella una proprietà che coinvolge due classi, ed è una proprietà di entrambe: *autore* dice qualcosa di ogni libro e di ogni persona.

Ogni relazione che interessa al sistema è un'associazione a sé. Se il sistema deve sapere anche chi ha letto quale libro, serve una seconda associazione fra `Libro` e `Persona`, per esempio *lettore*, con la sua regola.

L'analisi decide che cosa il software saprà. Disegnare un'associazione vuol dire chiedere al programma di memorizzare quei collegamenti, con le strutture dati adatte. Il programmatore realizza ciò che lo schema prevede, e niente di più: un'informazione che l'analisi non ha previsto, il sistema non la avrà. Per questo l'analista, mentre disegna, chiede al committente se servono altre associazioni.

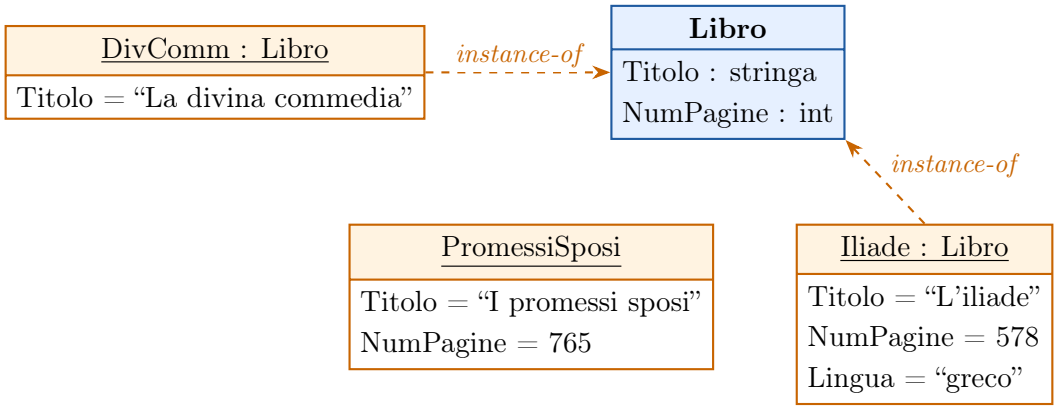
Riepilogo

- **Analisi**: dai requisiti allo schema concettuale; che cosa, non come.
- **Schema concettuale**: completo, preciso, rigoroso, leggibile, indipendente dal linguaggio; riferimento di tutte le fasi successive.
- **Componenti**: diagramma delle classi, delle attività, degli stati, documenti di specifica.
- **UML**: linguaggio grafico, non metodologia; conta il tipo di tratto.
- **Oggetto**: `id : Classe`, vita propria, identificatore unico, classe più specifica; arco instance-of tratteggiato.
- **Livelli**: classi intensionale, oggetti estensionale.
- **Attributo**: `Nome : tipo`; ogni istanza ha un valore, e solo per gli attributi della sua classe.
- **Associazione**: relazione $R \subseteq S_1 \times S_2$ fra le istanze di due classi; linea piena con il nome.

8 Esercizi

Esercizio 1 – Il diagramma è corretto?

Il diagramma rappresenta tre libri. È corretto? Se no, quali sono gli errori?



Soluzione a pagina 10 →

Esercizio 2 – Autori e lettori

Il sistema di una biblioteca ha le classi `Libro`, con `Titolo : stringa`, e `Persona`, con `Nome : stringa`, legate dall'associazione *autore*. Il committente vuole sapere anche quali persone hanno letto ciascun libro.

1. La coppia (Divina Commedia, Mario Rossi), dove Mario Rossi ha letto il libro ma non lo ha scritto, appartiene ad *autore*?
2. Come si modifica il diagramma?

Soluzione a pagina 10 →

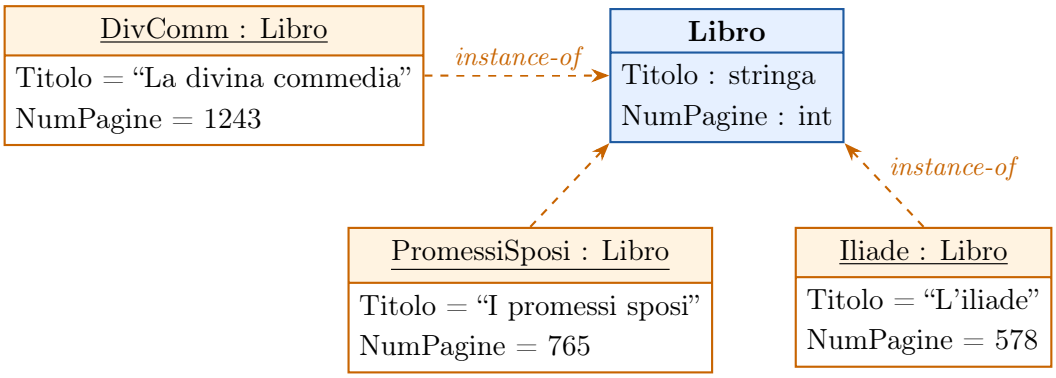
9 Soluzioni

Soluzione dell'Esercizio 1 – Il diagramma è corretto?

Il diagramma ha tre errori:

1. `DivComm : Libro` non ha un valore per `NumPagine`. Ogni istanza di `Libro` deve avere un valore per ogni attributo della classe.
2. `Iliade : Libro` ha un valore per `Lingua`, che non è un attributo di `Libro`. Un oggetto non può avere valori per attributi che la sua classe non definisce: o si toglie `Lingua` dall'oggetto, o lo si aggiunge alla classe, e allora tutti i libri devono averlo.
3. `PromessiSposi` non dice di quale classe è istanza, e manca l'arco instance-of.

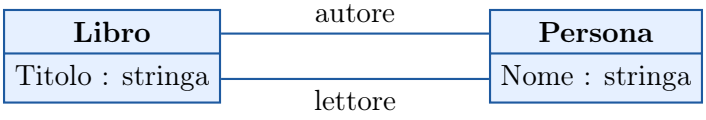
Il diagramma corretto, togliendo `Lingua`:



← *Torna all'Esercizio 1, pagina 8*

Soluzione dell'Esercizio 2 – Autori e lettori

1. No. *autore* contiene una coppia solo se la persona ha scritto il libro: è la sua regola generativa, data dal nome.
2. Leggere è un'altra relazione fra le istanze delle stesse classi, quindi serve una seconda associazione, *lettore*. La coppia (Divina Commedia, Mario Rossi) appartiene a *lettore*.



← *Torna all'Esercizio 2, pagina 9*