Progettazione del Software
Specifica di classi e attività

Giuseppe De Giacomo, Massimo Mecella
Dipartimento di Informatica e Sistemistica
SAPIENZA Università di Roma

La specifica

Il diagramma delle classi e il diagramma delle attività viene corredato da:

- una **specifica** per ogni **classe**
- una **specifica** per ogni **attività**

La specifica di una classe ha lo scopo di definire precisamente il comportamento di ogni operazione della classe

La specifica di un'attività ha lo scopo di definire precisamente il comportamento di ogni operazione di cui l'attività è costituito, e dell'attività nel suo complesso (variabili e flusso di controllo)

Specifica di una classe

La specifica di una classe C ha la seguente forma:

InizioSpecificaClasse C

Specifica della operazione 1

…

Specifica della operazione N

FineSpecifica

Specifica di una operazione

La specifica di una operazione di classe ha la seguente forma:

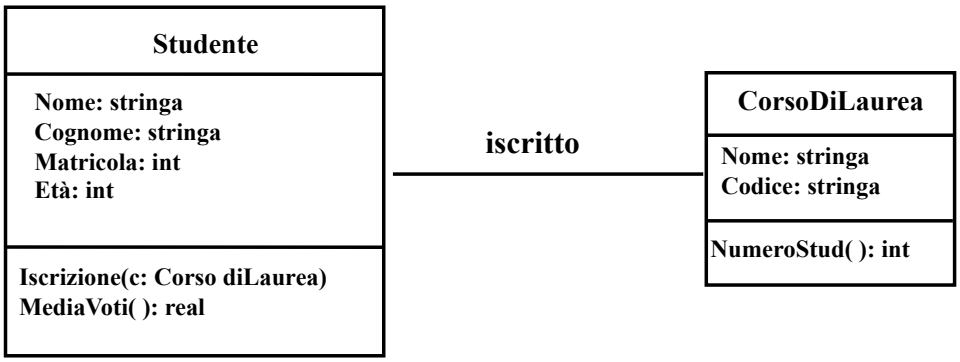
alfa (X1: T1, … , Xn: Tn): T

pre: *condizione*

post: *condizione*

- alfa (X1: T1, … , Xn: Tn): T è la **segnatura** dell'operazione (T può mancare),
- pre rappresenta la **precondizione** dell'operazione, cioè l'insieme delle condizioni (a parte quelle già stabilite dalla segnatura) che devono valere **prima** di ogni esecuzione della operazione
- post rappresenta le **postcondizioni** della operazione, cioè l'insieme delle condizioni che devono valere **alla fine** di ogni esecuzione della operazione

Esempio di specifica di una operazione



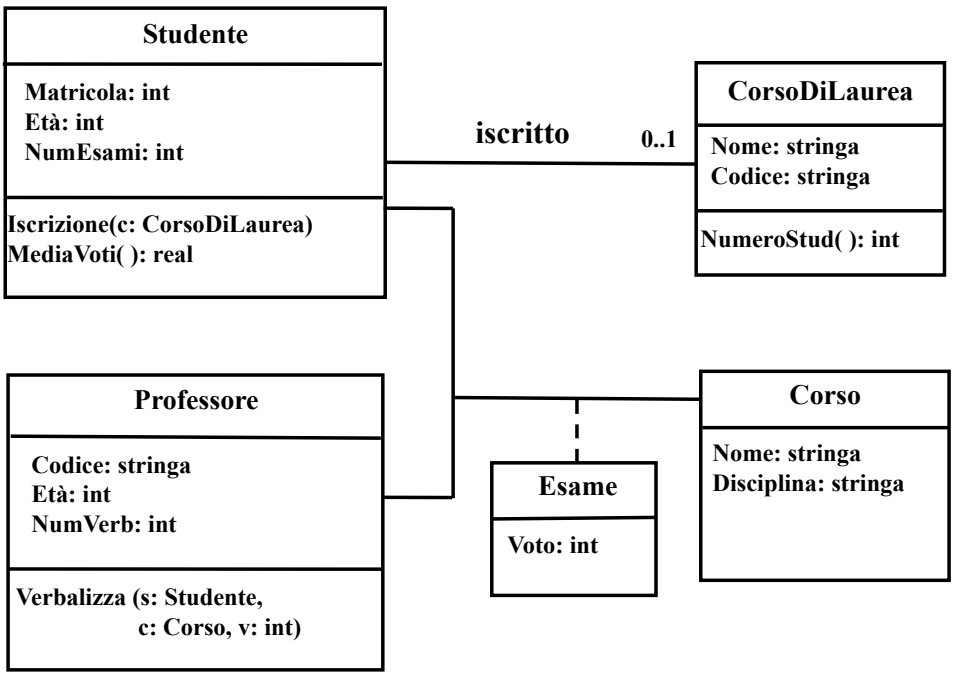
InizioSpecificaClasse CorsoDiLaurea
NumeroStud() : int
pre : nessuna
post : result è uguale al numero di studenti iscritti nel corso di laurea this
FineSpecifica

Precondizioni e postcondizioni

- Nella specifica di una operazione, nella **precondizione** si usa "**this**" per riferirsi all'oggetto di invocazione della operazione

- Nella specifica di una operazione, nella **postcondizione** si usa
  - "**this**" per riferirsi all'**oggetto di invocazione** della operazione nello stato corrispondente **alla fine** della esecuzione della operazione
  - "**result**" per riferirsi al **risultato** restituito dalla esecuzione della operazione
  - **pre(alfa)** per riferirsi al valore della espressione **alfa nello stato corrispondente alla precondizione**

Esempio di diagramma delle classi



Esempio di specifica di classi

InizioSpecificaClasse Professore
Verbalizza(s: Studente, c: Corso, v: int)
pre : s non ha ancora sostenuto l'esame c, e 18 ≤ v ≤ 31
post : this, s e c sono collegati da un link di tipo Esame, con voto v. Inoltre vale che s.NumEsami = pre(s.NumEsami) + 1, e this.NumVerb = pre(this.NumVerb) + 1
FineSpecifica

InizioSpecificaClasse Studente
Iscrizione(c: CorsoDiLaurea)
pre : this non è iscritto ad alcun CorsoDiLaurea
post : this è iscritto al CorsoDiLaurea c,
MediaVoti() : real
pre : this.NumEsami > 0
post : result è la media dei voti degli esami sostenuti da this
FineSpecifica

Specifica mediante una notazione formale

InizioSpecificaClasse Professore
Verbalizza(s: Studente, c: Corso, v: int)
pre : ¬(∃ p . p ∈ Professore∧ <s,p,c> ∈ Esame)
∧ 18 ≤ v ≤ 31

post : Esame = pre(Esame) ∪ {<s,this,c>} ∧
Esame.voto(s,this,c) = v ∧
s.NumEsami = pre(s.NumEsami) + 1 ∧
this.NumVerb = pre(this.NumVerb) + 1
FineSpecifica

Specifica mediante una notazione formale (2)

InizioSpecificaClasse Studente
Iscrizione(c: CorsoDiLaurea)
pre : ¬(∃ c2 ∈ CorsoDiLaurea . <this,c2> ∈ iscritto)
post : iscritto = pre(iscritto) ∪ {<this,c>}

MediaVoti() : real
pre : (∃ c ∈ CorsoDiLaurea . <this,c> ∈ iscritto) ∧
this.NumEsami > 0
post : definiamo Voti come l'insieme
{<c,v > | c ∈ Corso ∧ p ∈ Professore ∧
<this,p,c> ∈ Esame ∧ Esame.voto (this,p,c) = v}

$$\text{result} = \frac{\sum_{<c,v>\in Voti} v}{\text{this.NumEsami}}$$

FineSpecifica

Esercizio

Scrivere la specifica della seguente operazione di classe usando una notazione formale:

- Operazione NumeroStud() della classe CorsoDiLaurea

Soluzione

InizioSpecificaClasse CorsoDiLaurea
NumeroStud():int
pre : true
post : result = card({ s | s ∈ Studente ∧ <s,this> ∈ iscritto})
FineSpecifica

Specifica di attività

Distinguiamo la specifica di

- attività **atomiche**
- attività **complesse**

Per una attività atomica abbiamo necessità di specificare l'operazione che svolge in termini di segnatura, precondizione e postcondizioni.

Per le attività complesse invece una una specifica in termini di precondizioni e postcondizioni non è sufficiente.

Specifica di attività atomiche

Per quanto riguarda le attività atomiche la specifica è del tutto analoga alla specifica delle classi: Una attività atomica è costituita da una singola operazione.

```
InizioSpecificaAttivitàAtomica A
...
FineSpecifica
```

Specifica di una attività atomica

La specifica della operazione di un'attività atomica ha la seguente forma (analoga alla specifica di una operazione di classe):

alfa (X1: T1, … , Xn: Tn): (Y1:T1, …, Yn:Tn)
pre: *condizione*
post: *condizione*

- alfa (X1: T1, … , Xn: Tn): (Y1: T1, …, Yn: Tn) è la **segnatura** dell'operazione:
  - alfa è il nome dell'operazione (tipicamente quello dell'attività atomica)
  - X1: T1, … , Xn: Tn sono i parametri dell'operazione
  - Y1: T1, …, Yn: Tn sono i risultati dell'operazione (questi possono essere 0, 1, o più di uno – *se il risultato è uno solo allora si può omettere il nome denotandolo convenzionalmente con ``result''*).
- pre rappresenta la **precondizione** dell'operazione, cioè l'insieme delle condizioni (a parte quelle già stabilite dalla segnatura) che devono valere **prima** di ogni esecuzione della operazione
- post rappresenta le **postcondizioni** della operazione, cioè l'insieme delle condizioni che devono valere **alla fine** di ogni esecuzione della operazione

Precondizioni e postcondizioni

- Nella specifica delle precondizione e postcondizioni di una attività atomica **non** si può usare "**this**" non essendoci l'oggetto di invocazione.

- Si usano invece (analogamente al caso delle operazione di classe):
  - "**Y1, …, Yn**" per riferirsi ai **risultati** restituiti dalla esecuzione della operazione
  - **pre(alfa)** per riferirsi al valore della espressione **alfa nello stato corrispondente alla precondizione**

Esempio

Scrivere la specifica delle seguenti attività atomica (usando una notazione formale):

Specifica dell'esempio

InizioSpecificaAttivitàAtomica MediaEsami

MediaEsami(c: CorsoDiLaurea): (res: real)

pre : (∃ s ∈ Studente . <s,c> ∈ iscritto)

post : Definiamo EsamiDelCorso come l'insieme

{<s,p,co> | s ∈ Studente ∧ p ∈ Professore ∧ co ∈ Corso ∧ <s,p,co> ∈ Esame ∧ <s,c> ∈ iscritto}

$$res = \frac{card(EsamiDelCorso)}{c.NumeroStud()}$$

FineSpecifica

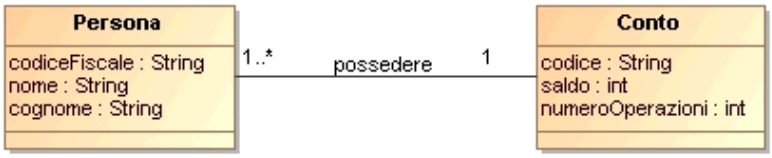
Specifica di attività complesse

Per quanto riguarda le attività complesse la specifica ha una natura profondamente diversa.

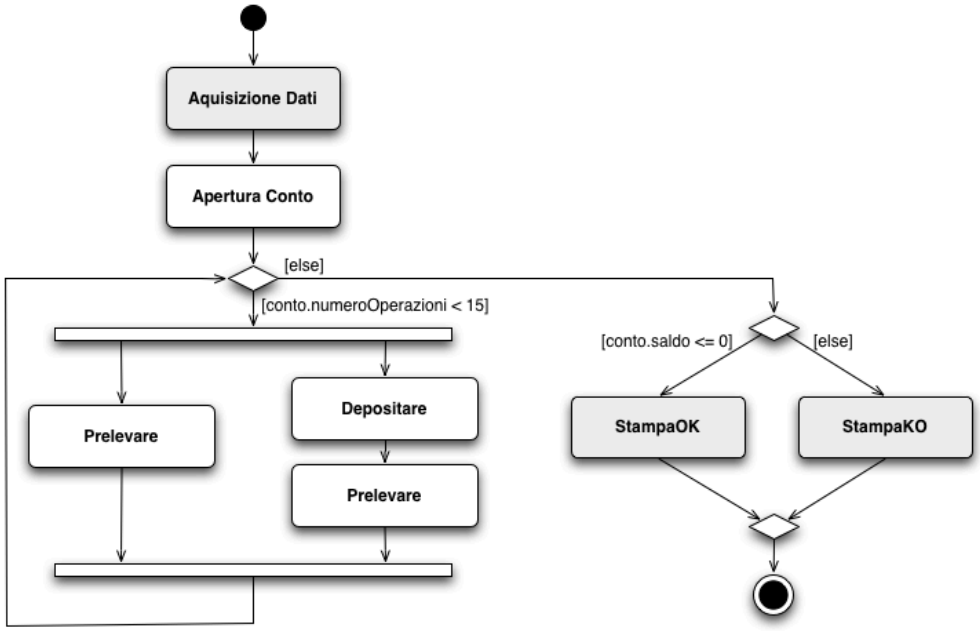
Infatti in tali attività il flusso stesso di controllo è di interesse.

Naturalmente il processo descritto da tali attività insiste sulla estensione del diagramma delle classi ed infatti le attività atomiche accedono/modificano tale diagramma.

Esempio: diagramma delle classi



Esempio: diagramma delle attività



Analisi del diagramma delle attività

Un diagramma delle attività si compone di

- Attività atomiche che lavorano sulle classi, che chiamiamo **task**: di queste daremo la specifica come visto tenendo presente il diagramma delle classi
- Attività atomiche che si occupano del **I/O** (interfacciamento della applicazione con i clienti): di queste non daremo una specifica in questo corso.
- Strutture di controllo

Classificazione attività atomiche

Task:

- Apertura Conto
- Prelevare
- Depositare

I/O:

- Acquisizione Dati
- StampaOK
- StampaKO

Esempio di specifica attività atomiche

```
InizioSpecificaAttivitàAtomica AcquisizioneDati

AcquisizioneDati():
            (per1: Persona, per2: Persona, conto: Conto)

   pre:     --
   post:    gli oggetti per1, per2, conto vengono creati
           con i valore letti da input

FineSpecifica
```

Esempio di specifica attività atomiche

InizioSpecificaAttivitàAtomica Apertura Conto

Apertura Conto(p1: Persona, p2: Persona,c: Conto)

pre: not (<p1,c> in possedere and <p2,c> in possedere)

post: possedere = pre(possedere) union {<p1,c>,<p2,c>}

FineSpecifica

Esempio di specifica attività atomiche

InizioSpecificaAttivitàAtomica Depositare

Depositare(p: Persona, c: Conto)

pre: <c,p> in possedere
post: c.saldo = pre(c.saldo) + random
//si deposita somma a caso

FineSpecifica

Esempio di specifica attività atomiche

```
InizioSpecificaAttività Prelevare

   Prelevare(p: Persona, c: Conto)
   pre:    <p,c> in possedere
   post:   c.saldo = pre(c.saldo) - random
                        //si preleva somma a caso
FineSpecifica
```


Esempio di specifica attività atomiche

```
InizioSpecificaAttivitàAtomica StampaKO

StampaKO()
   pre:    --
   post:   manda in output messaggio "KO"

FineSpecifica


InizioSpecificaAttivitàAtomica StampaOK

StampaOK()
   pre:    --
   post:   manda in output messaggio "OK"

FineSpecifica
```

Specifica di attività complesse

Oltre a definire le attività atomiche, l'attività complessa si deve occupare di legare gli input e output delle varie operazioni delle stesse tra loro, oltre che del flusso di controllo già specificato nel diagramma delle attività.

Per fare ciò, la specifica delle attività complesse è costituita da:

- **Segnatura** che indica eventuali **parametri di ingresso** e **parametri di uscita** (**risultati**)
- **Variabili di processo** che in aggiunta ai parametri di ingresso e uscita possono essere usate per memorizzare e passare informazioni da una attività atomica ad un'altra
- **Specifica del processo** vero e proprio in termini di qualche formalismo (noi useremo pseudo codice

Specifica di una attività complessa

La specifica di una attività complessa è costituita da:

- Segnatura: questa è del tutto analoga alla segnatura delle attività atomiche e che quindi descrive i parametri di input e di output della attività stessa.
- Variabili di processo: queste formano una sorta di memoria locale della attività e sono principalmente usate per legare i parametri delle varie sottoattività (atomiche o no) tra loro.
- Processo: questo è la descrizione procedurale del processo messo in atto dalla attività, secondo il diagramma delle attività ma con in più l'esatta descrizione di quali dati sono scambiati tra le sottoattività. Noi descriveremo quest'ultimo in pseudocodice Java (vedi dopo).

Specifica di una attività complessa

La Specifica di una attività complessa ha la seguente forma:

```
InizioSpecificaAttivita NomeAttivita

Segnatura Attività Complessa
  con nome parametri di input e parametri di output

VariabiliProcesso

Definizione delle variabili di processo
 con nome e tipo

InizioProcesso

Descrizione procedurale del processo stesso
    in pseudocodice

FineProcesso

FineSpecifica
```


Esempio di specifica di attività complessa

```
InizioSpecificaAttivita Principale

Principale()

VariabiliProcesso

    marito: Persona
    moglie: Persona
    conto:  Conto

InizioProcesso
   AquisizioneDati():(marito,moglie,conto);
   AperturaConto(marito,moglie,conto);
   while (conto.numeroOperazioni < 15) {
     fork {
       thread t1: {
          Prelevare(marito,conto);
       }
       thread t2: {
          Deposita(moglie,conto);
         Prelevare(moglie,conto);
       }
     }
     join t1, t2;
   }
   if (conto.saldo <= 0)
      StampaKO();
   else
      StampaOK();

FineProcesso

FineSpecifica
```

Pseudo codice per rappresentare processi

Come pseudocodice usiamo un formalismo simile a Java.

- Per rappresentare cicli e decisioni usiamo le strutture di controllo Java (**`if-else`**, **`while`**, **`do-while`**).
  Esempio:

```
while (conto.numeroOperazioni < 15) { … }

if (conto.saldo <= 0) StampaKO(); else  StampaOK();
```

- Per invocare sottoattività (atomiche e non) usiamo la sua segnatura come fosse un metodo statico, passandogli in input e in output i parametri attuali
  Esempio:

```
Prelevare(marito,conto);
```

- Per quanto riguarda il passaggio dei parametri in input, si assume il solito passaggio di parametri per valore di Java: la variabile parametro attuale viene valutata e il riferimento corrispondente diviene il valore iniziale del parametro attuale di input dell'attività.

Pseudo codice per rappresentare processi

- Per invocare sottoattività (atomiche e non) usiamo la sua segnatura come fosse un metodo statico, passandogli in input e in output i parametri attuali
  Per quanto riguarda il passaggio dei parametri in output delle sottoattività (atomiche e non) , si assume un passaggio di parametri per risultato: la variabile parametro attuale viene assegnata con il nuovo valore al termine dell'operazione, durante l'operazione stessa tale variabile non può essere ne letta ne modificata.
  Esempio:

```
      AquisizioneDati():(moglie,marito,conto);
//NB la segnatura è AquisizioneDati():(pe1: Persona, pe2: Persona, co: Conto);
```

- In pratica per quanto riguarda il passaggio dei parametri di output possiamo pensare l'istruzione: `AquisizioneDati():(moglie,marito,conto);` come una **abbreviazione** per le seguenti istruzioni:

```
AquisizioneDati();
marito = AcquisizioneDati.pe1;
moglie = AcquisizioneDati.pe2;
conto =  AcquisizioneDati.co;
```

dove, attraverso la notazione basata sul punto, usiamo i parametri di uscita come fossero campi dati di una classe.

NB: queste istruzioni sono più simili a istruzioni Java: infatti Java prevede passaggio di parametri solo per valore e non per risultato.

Pseudo codice per rappresentare processi

- Per introdurre ed eliminare thread di controllo concorrenti facciamo uso di nuovi costrutti speciali **fork** e **join**
- A ciascun thread generato associamo un identificatore che può poi essere usato nei join

  Esempio fork:

```
fork {
   thread t1: {
     …
   }
   thread t2: {
     …
   }
}
```

  Esempio join:

```
join t1, t2;
```

- E' naturalmente possibile fare uso di sottoattività complesse nel definire un processo di una attività complessa, ed è anche possibile fare uso della ricorsione (ne vedremo degli esempi in seguito).
- Inoltre se necessario, si può usare liberamente l'assegnazione sulle le variabili di processo