

Elementi di UML (6)

Università degli Studi di Bologna
 Facoltà di Scienze MM. FF. NN.
 Corso di Laurea in Informatica
 Anno Accademico 2001-2002

Corso di:
 Laboratorio di Ingegneria del Software

Rocco Moretti moretti@cs.unibo.it



UML

1

Vincoli

n **Vincolo:**

Una restrizione su uno o più valori di un modello o di un sistema orientato agli oggetti (o parte di essi).

n UML definisce tre stereotipi standard per i vincoli:

- n invarianti
- n precondizioni
- n postcondizioni

UML

2

Differenti tipi di vincoli: esempi

n **Invariante** di una classe:

- n un vincolo che deve essere sempre soddisfatto da **TUTTE** le istanze di una classe

n **Precondizione** di un'operazione:

- n un vincolo che deve essere sempre soddisfatto **PRIMA** dell'esecuzione dell'operazione

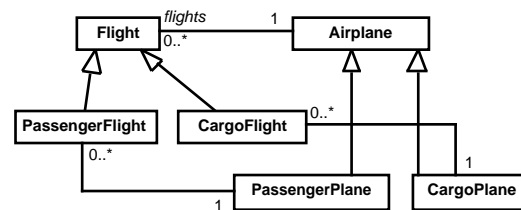
n **Postcondizione** di un'operazione:

- n un vincolo che deve essere sempre soddisfatto **DOPO** l'esecuzione di un'operazione

UML

3

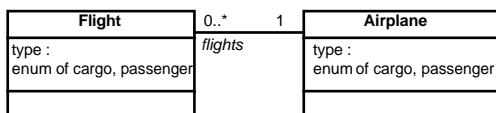
Possiamo essere più precisi? (1/2)



UML

4

Possiamo essere più precisi? (2/2)



{context Flight

inv: type = #cargo implies airplane.type = #cargo

inv: type = #passenger implies airplane.type = #passenger}

UML

5

Che cos'è OCL?

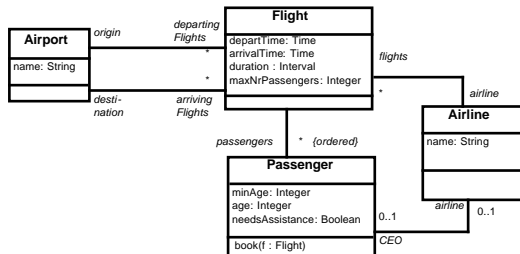
n OCL (Object Constraint Language) è:

- n un linguaggio testuale per descrivere vincoli
- n il linguaggio per la definizione di vincoli utilizzato nei modelli UML
- n un linguaggio formale senza side effect

UML

6

Uno scenario



UML

7

Contesto di un vincolo (1/3)

- Ogni espressione OCL è legata ad uno specifico contesto:
 - il contesto è spesso rappresentato graficamente dall'elemento al quale il vincolo viene collegato
 - il contesto è rappresentato a livello di espressione dall'utilizzo della parola chiave "self" che è implicita in tutte le espressioni OCL

UML

8

Contesto di un vincolo (2/3)

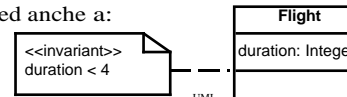
- Sintassi per un'invariante:
 - context** typeName **inv:** constraint
 - context** packageName: typeName **inv:** constraint
- Sintassi per precondizioni e postcondizioni:
 - context** typeName::operationName(param1: Type1, ...): returnType
 - pre:** constraint
 - post:** constraint

UML

9

Contesto di un vincolo (3/3)

- I vincoli possono essere presentati all'interno di un modello UML o in un documento separato
 - L'espressione:
 - context** Flight **inv:** self.duration < 4
 - è equivalente a:
 - context** Flight **inv:** duration < 4
 - ed anche a:



UML

10

Elementi di un'espressione OCL

- In un'espressione OCL possono essere utilizzati i seguenti elementi:
 - Tipi di base: String, Boolean, Integer, Real
 - Classi derivate dal modello UML e loro caratteristiche:
 - attributi
 - operazioni che non hanno side effect
 - Associazioni derivate dal modello UML

UML

11

Attributi: esempio

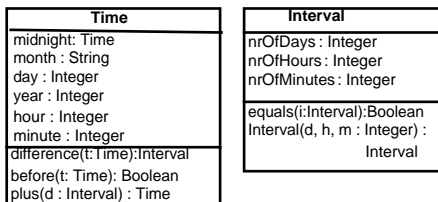
- Attributi:
 - context** Flight **inv:**
 - self.maxNrPassengers <= 1000
 - context** Passenger **inv:**
 - age >= Passenger.minAge

UML

12

Operazioni: esempio

context Flight inv:
 self.departTime.difference(self.arrivalTime)
 .equals(self.duration)



UML

13

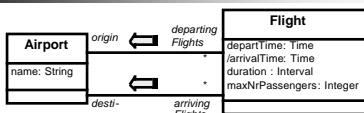
Associazione e navigazione

- Ogni associazione nel modello sottointende un determinato percorso di navigazione
- Il contesto dell'espressione indica il punto di partenza

UML

14

Navigazione: esempio



context Flight
inv: origin <> destination
inv: origin.name = 'Amsterdam'

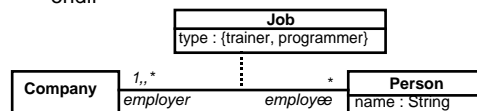
context Flight
inv: destination.name = 'Paris Only'

UML

15

Associazione: esempio

context Person inv:
 if employee.name = 'Mario Rossi' then
 job.type = #trainer
 else
 job.type = #programmer
 endif

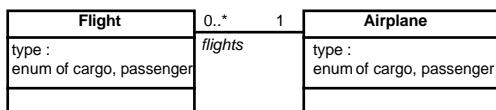


UML

16

Collection in OCL

- Molte navigazioni dei diagrammi rappresentano un insieme di elementi piuttosto che singoli elementi



UML

17

Collection: le tre tipologie

- **Set:**
 - Ad es., *arrivingFlights* (dal contesto *Airport*)
 - Non ordinato, elementi non ripetuti
- **Bag:**
 - Ad es., *arrivingFlights.duration* (dal contesto *Airport*)
 - Non ordinato, elementi ripetuti
- **Sequence:**
 - Ad es., *passengers* (dal contesto *Flight*)
 - Ordinato (vedi fig. pag. 7), elementi ripetuti

UML

18

Collection: operazioni

- n OCL prevede un grande numero di operazioni predefinite sulle collection
- n Sintassi:
 - n collection->operation

Uso dell'operatore ">"
invece dell'operatore "."

UML

19

L'operazione collect

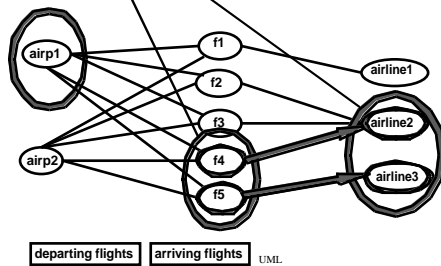
- n Sintassi:
 - collection->collect(elem : Type | expr)
 - collection->collect(elem | expr)
 - collection->collect(expr)
- n L'operazione *collect* crea una nuova collezione applicando l'espressione *expr* per ciascun elemento della collezione *collection*

UML

20

Operazione collect: esempio

context Airport inv:
self.arrivingFlights->collect(airLine)->notEmpty



departing flights arriving flights

UML

21

L'operazione select

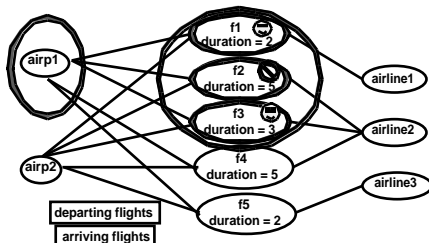
- n Sintassi:
 - collection->select(elem : Type | expr)
 - collection->select(elem | expr)
 - collection->select(expr)
- n L'operazione *select* raccoglie in una nuova collezione il sottoinsieme degli elementi per i quali l'espressione *expr* è valutata *true*

UML

22

Operazione select: esempio

context Airport inv:
self.departingFlights->select(duration<4)->notEmpty



departing flights
arriving flights

UML

23

L'operazione forAll

- n Sintassi:
 - collection->forAll(elem : Type | expr)
 - collection->forAll(elem | expr)
 - collection->forAll(expr)
- n L'operazione *forAll* restituisce *true* se l'espressione *expr* è valutata *true* per ogni elemento della collection, *false* altrimenti

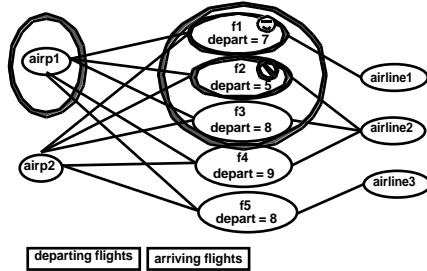
UML

24

Operazione forAll: esempio

context Airport **inv**:

self.departingFlights->forAll(departTime.hour>6)



UML

25

L'operazione exists

n Sintassi:

collection->exists(elem : Type | expr)

collection->exists(elem | expr)

collection->exists(expr)

n L'operazione *exists* restituisce *true* se c'è almeno un elemento nella collection rispetto al quale l'espressione *expr* viene valutata *true*, *false* altrimenti

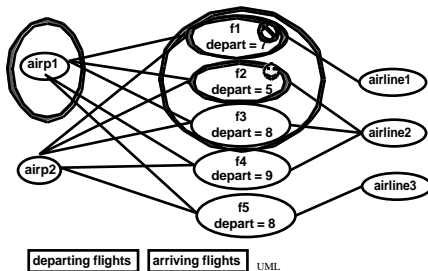
UML

26

Operazione exists: esempio

context Airport **inv**:

self.departingFlights->exists(departTime.hour<6)



UML

27

Altre operazioni sulle collection

n *isEmpty*: *true* se la collection non ha elementi

n *notEmpty*: *true* se la collection ha almeno un elemento

n *size*: il numero degli elementi nella collection

n *count(elem)*: il numero delle occorrenze di *elem* nella collection (se è ammessa la ripetizione degli elementi)

n *includes(elem)*: *true* se *elem* è nella collection

n *excludes(elem)*: *true* se *elem* non è nella collection

n *includesAll(coll)*: *true* se tutti gli elementi di *coll* sono nella collection

UML

28

L'operazione oclInState

n Sintassi:

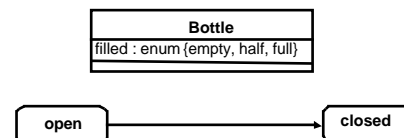
obj.ocInState(state)

n L'operazione *oclInState* restituisce *true* se l'oggetto *obj* è nello stato *state*, *false* altrimenti

UML

29

Operazione oclInState: esempio



context Bottle **inv**:

self.ocInState(closed) implies filled = #full

UML

30

Variabili locali: il costrutto Let

- Il costrutto *Let* consente la definizione di variabili locali ad un vincolo
- Sintassi:
Let var : Type = <expr1> in <expr2>

UML

31

Costrutto let: esempio

context Airport **inv:**

Let supportedAirlines : Set (Airline) =
self.arrivingFlights -> collect(airLine) in
(supportedAirlines ->notEmpty) and
(supportedAirlines ->size < 500)

UML

32

L'operazione iterate

- L'operazione *iterate* su una collection consiste nell'applicazione di una determinata funzione su tutti gli elementi inclusi nella collection, e restituendo la nuova collection così determinata
- Sintassi:
collection->iterate(elem : Type;
answer : Type = <value> |
<expr con elem e answer>)

UML

33

Operazione iterate: esempio

- L'operazione:

context Airline **inv:**

flights->select(maxNrPassengers > 150)->notEmpty

- è equivalente a:

context Airline **inv:**

```
flights->iterate (f : Flight;
  answer : Set(Flight) = Set{ } |
  if f.maxNrPassengers > 150 then
    answer->including(f)
  else
    answer endif )->notEmpty
```

UML

34

Ereditarietà dei vincoli (1/2)

- Principio guida:
Principio di sostituzione di Liskov
(Liskov's Substitution Principle: **LSP**)
 - Quando ci si aspetta un'istanza di una classe, la si può sempre sostituire con un'istanza di una sua qualsiasi sottoclasse

UML

35

Ereditarietà dei vincoli (2/2)

- Conseguenze di LSP per le invarianti:
 - Un'invariante è sempre ereditata da ogni sottoclasse
 - Le sottoclassi possono **rafforzare** l'invariante
- Conseguenze di LSP per le precondizioni e le postcondizioni:
 - Una precondizione può essere **indebolita**
 - Una postcondizione può essere **rafforzata**

UML

36

Conclusioni

- n Le invarianti OCL permettono di:
 - n modellare in modo più preciso
 - n essere indipendenti dall'implementazione
- n Le precondizioni e le postcondizioni permettono di:
 - n specificare contratti (design by contract)
 - n specificare interfacce di componenti in modo preciso
- n Utilizzo di OCL:
 - n Mantenere i vincoli semplici
 - n Combina sempre il linguaggio naturale con OCL
 - n Usa dei tool per il controllo dell'OCL impiegato

UML

37

Tool per OCL

- n Cybernetics
 - n www.cybernetic.org
- n Università di Dresda
 - n www-st.inf.tu-dresden.de/ocl/
- n Boldsoft
 - n www.boldsoft.com
- n ICON computing
 - n www.iconcomp.com

UML

38

Riferimenti

- n [UML 1.3] *OMG UML Specification v. 1.3*,
OMG doc# ad/06-08-99
- n [UML 1.4] *OMG UML Specification v. 1.4*, UML
Revision Task Force recommended final draft,
OMG doc# ad/01-02-13.

UML

39

Per ulteriori informazioni

- n Web:
 - n UML 1.4 RTF: www.celigent.com/omg/umlrtf
 - n OMG UML Tutorials:
www.celigent.com/omg/umlrtf/tutorials.htm
 - n UML 2.0 Working Group:
www.celigent.com/omg/adptf/wgs/uml2wg.htm
 - n OMG UML Resources: www.omg.org/uml/
- n Email
 - n uml-rtf@omg.org
- n Conferences & workshops
 - n UML World 2001, New York, June 11-14, 2001
 - n UML 2001, Toronto, Canada, Oct. 1-5, 2001
 - n OMG UML Workshop 2001, San Francisco, Dec. 3-6, 2001

UML

40