Specifica di vincoli in modelli UML con OCL

Lezione 21 di Ingegneria del software

Docente: Giuseppe Scollo

Università di Catania, sede di Comiso (RG)
Facoltà di Scienze Matematiche, Fisiche e Naturali
Corso di Studi in Informatica applicata, AA 2009-10

Indice

1. Specifica di vincoli in modelli UML con OCL
2. OCL: origini, cenni storici, definizioni
3. OCL e MDA nello sviluppo del software
4. livelli di maturità di modellazione
5. UML + OCL → MDA
6. un esempio più convincente
7. UML + OCL → modelli più espressivi
8. UML + OCL → modelli più precisi
9. criteri di progetto di OCL
0. cosa è un modello?
1. come usare OCL

OCL: origini, cenni storici, definizioni

origini ed evoluzione:

- **anni '90:** metodologia Syntropy, IBM, ...
- **1997:** v. 1, incluso da OMG in UML 1.1
- **2003:** v. 2, proposta, in UML 2.0 (Warmer & Kleppe, 2003)
- **2005:** v. 2, specifica, in UML 2.0 (OMG, 2005)
- **2006:** v. 2, versione ufficiale, in UML 2.0 (OMG, 2006)

linguaggio per **espressioni** su modelli UML a oggetti:

vincolo:

- restrizione su uno o più valori di (una parte di) un modello o sistema a oggetti

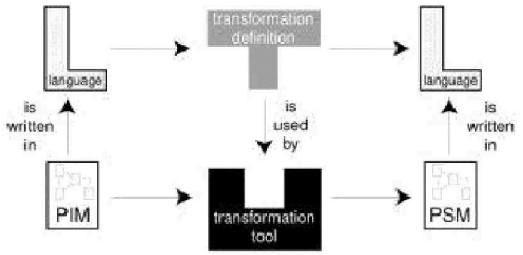
espressioni:

- vincoli, interrogazioni, riferimenti, regole, condizioni, ...

OCL e MDA nello sviluppo del software

OCL è inteso favorire l'uso di modelli per l'automazione dello sviluppo del software, nella concezione della Model Driven Architecture (MDA), ovvero di uno sviluppo basato su modelli e loro trasformazioni

le trasformazioni da PIM a PSM sono intese avvalersi di strumenti basati su definizioni standard delle trasformazioni fra i linguaggi in gioco:



tratta da: (Warmer & Kleppe, 2003), Cap. 1

livelli di maturità di modellazione

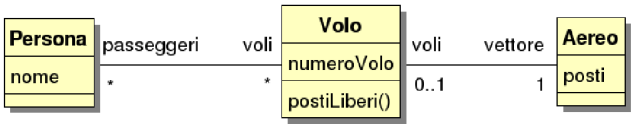
in analogia con i livelli di maturità del processo produttivo (CMM):

0. : **nessuna specifica**
   da dilettanti
   sviluppo di piccole dimensioni e/o di breve durata
1. : **testo naturale**
   problemi: ambiguità, difficile manutenibilità
2. : **testo naturale + diagrammi di alto livello**
   migliore comprensibilità, ma stessi problemi
3. : **modelli + testo naturale**
   migliore corrispondenza con il prodotto, ma ancora sviluppo manuale
4. : **modelli precisi**
   *il **target** di MDA, automazione **significativa** dello sviluppo da modelli*
   *caratteristica del modello: **consistenza***
5. : **solo modelli precisi**
   ***goal** del futuro, automazione **completa** dello sviluppo da modelli*
   *caratteristiche del modello: **consistenza** e **completezza***

UML + OCL → MDA

UML + OCL → modelli precisi → maturità di livello ≥ 4

un esempio dei limiti espressivi dei soli diagrammi UML:



tradotto da: (Warmer & Kleppe, 2003), Cap. 1

problema: voli a numero di passeggeri *illimitato* ?

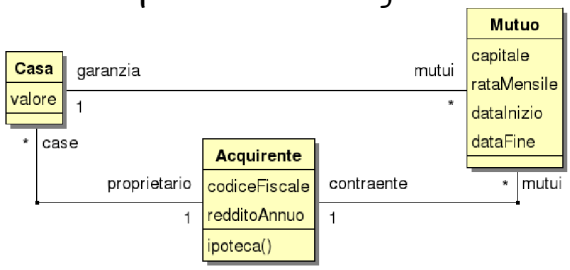
soluzione con OCL:

aggiungere un **vincolo** al modello espresso dal diagramma!

```
context Volo
inv: passeggeri->size() <= vettore.posti
```

un esempio più convincente

un **altro esempio** dei limiti espressivi dei soli diagrammi UML:



tradotto da: (Warmer & Kleppe, 2003), Cap. 1

nel diagramma, pur se si mostrano i tipi degli attributi e i parametri delle operazioni:

```
valore, redditoAnnuo, capitale, rataMensile : Denaro
dataInizio, dataFine : Data
codiceFiscale : string
ipoteca(somma: int, garanzia: Casa) : void
```

molti **vincoli ovvi** restano tuttavia non specificati ...

UML + OCL → modelli più espressivi

per la **precisione**, c'è da aggiungere fra l'altro che:

1. si possono ipotecare solo case di cui si è **proprietari**
2. la data di **inizio** di un mutuo deve essere **anteriore** alla data della sua **fine**
3. il **codice fiscale** di ciascun acquirente deve essere **unico**
4. un mutuo viene concesso solo a chi dispone di un **reddito annuo sufficiente**
5. un mutuo viene concesso solo in presenza di una **garanzia sufficiente**

UML + OCL → modelli più precisi

con **precisione**, in *OCL*:

```
context Mutuo
inv garanzia.proprietario = contraente
```

```
context Mutuo
inv dataInizio < dataFine
```

```
context Acquirente
inv Acquirente::allInstances()->isUnique(codiceFiscale)
```

```
context Acquirente::ipoteca(somma:Denaro, garanzia:Casa)
pre self.mutui.rataMensile->sum() * 12 <= self.redditoAnnuo * 0.30
```

```
context Acquirente::ipoteca(somma:Denaro, garanzia:Casa)
pre garanzia.valore >= garanzia.mutui.capitale->sum()
```

criteri di progetto di OCL

caratteristiche del linguaggio e loro motivazioni:

espressione sia di **vincoli** che di **interrogazioni**

per: MDA, migliore espressività dei modelli

notazione non matematica (ma semantica matematica)

per: ragioni pragmatiche, ad es. facilitarne la diffusione

tipizzazione forte

per: facilitare l'analisi statica di correttezza

linguaggio **dichiarativo**

per: livello di astrazione della specifica di PIM,
indesiderabilità di effetti collaterali ai fini dell'analisi

cosa è un modello?

un insieme *coerente* e *consistente* di **elementi** del modello con **caratteristiche** (***features***) e **vincoli**

- **elementi:** classi, interfacce, stati, componenti, ...
- **caratteristiche:** attributi, operazioni, estremi di associazioni, azioni, ...
- **vincoli:** invarianti, pre- e post-condizioni, guardie, ...

unitarietà del modello:

- un diagramma esprime solo una **vista** del modello
- un ***model repository*** è la rappresentazione dei costituenti di un modello in uno strumento di elaborazione automatica
- un elemento del modello può occorrere in **più diagrammi**, in ciascuno con caratteristiche e vincoli appropriati alla vista espressa, ma rispettando globalmente **coerenza** e **consistenza**

la dichiarazione del **contesto** di un'espressione OCL

- ne indica l'**entità di riferimento** (elemento o caratteristica) nel modello
  - *di solito un **tipo** (contestuale): classe, interfaccia, tipo di dato, componente*
- specifica l'**istanza contestuale** rispetto alla quale si valuta l'espressione:
  - *designata da* `self`*, è la generica istanza del **tipo contestuale**, ovvero del contesto, o del suo contenitore quando il contesto non è un tipo, ad es. è un'operazione*

come usare OCL

alcuni suggerimenti di **metodo:**

cominciare con la stesura di **diagrammi di classi**

- definiscono il **vocabolario del modello**, usato in altri diagrammi, *e.g.* dinamici

data una collezione coerente di diagrammi, individuare le **necessità di ulteriore specifica**, tipicamente:

- specifica parziale di elementi, ad es. **derivati** ma privi della regola di derivazione
  - ⇐ esprimibile con `derive`
- aggiunta di ***business rules***
  - ⇐ espresse da **invarianti**
- specifica di **interfacce**
  - ⇐ mediante **pre-** e **post-condizioni** sulle operazioni
- identificazione di **ambiguità** nel diagramma
  - ⇐ eliminabili con **invarianti**