Università degli Studi di Roma "La Sapienza"
Facoltà di Ingegneria – Corso di Laurea in Ingegneria Gestionale
Corso di Progettazione del Software
Proff. Toni Mancini e Monica Scannapieco

Progetto **PI.20070419**, passo **A.2**

versione del 21 aprile 2007

Si vuole progettare e realizzare *Blackbuster*, un sistema informatico che consenta, ad una nuova catena di negozi di noleggio e vendita di DVD, di automatizzare la gestione delle relative procedure. In particolare, il sistema deve permettere ai clienti di noleggiare o acquistare DVD attraverso opportuni terminali, richiedendo ed utilizzando delle tessere prepagate ricaricabili. Inoltre, è richiesto che il sistema abbia delle funzionalità di profiling che tengano conto delle preferenze dei clienti, deducibili dalle precedenti interazioni con il sistema.

Si richiede di proseguire la fase di Analisi, estendendo lo schema concettuale di modo da modellare i requisiti aggiuntivi descritti in calce.

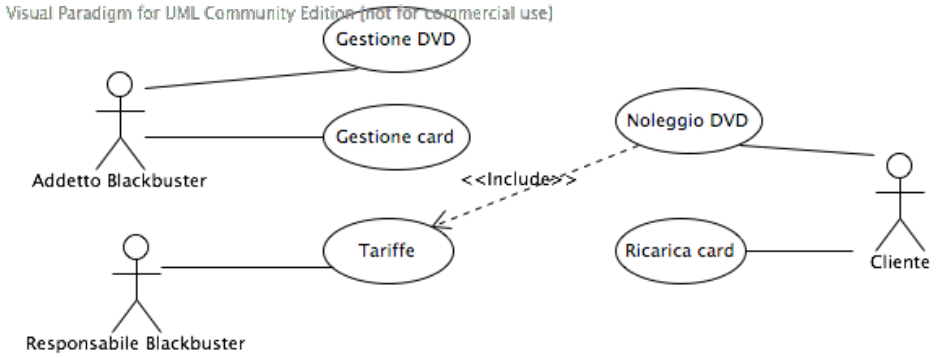
Requisiti

All'atto della restituzione di un DVD, il sistema controlla automaticamente che il supporto rientrato non sia danneggiato. Se dovesse esserlo, la copia deve venire immediatamente contrassegnata come danneggiata e non deve più essere oggetto di noleggi. Per semplicità, si assuma che l'operazione che modella la restituzione di un DVD abbia un argomento aggiuntivo per rappresentare le condizioni (DVD danneggiato o no) della copia ritornata. In caso di restituzione di una copia danneggiata, la card del cliente viene automaticamente addebitata del valore ritornato dall'operazione `penale()` presente nello use-case `Tariffe`.

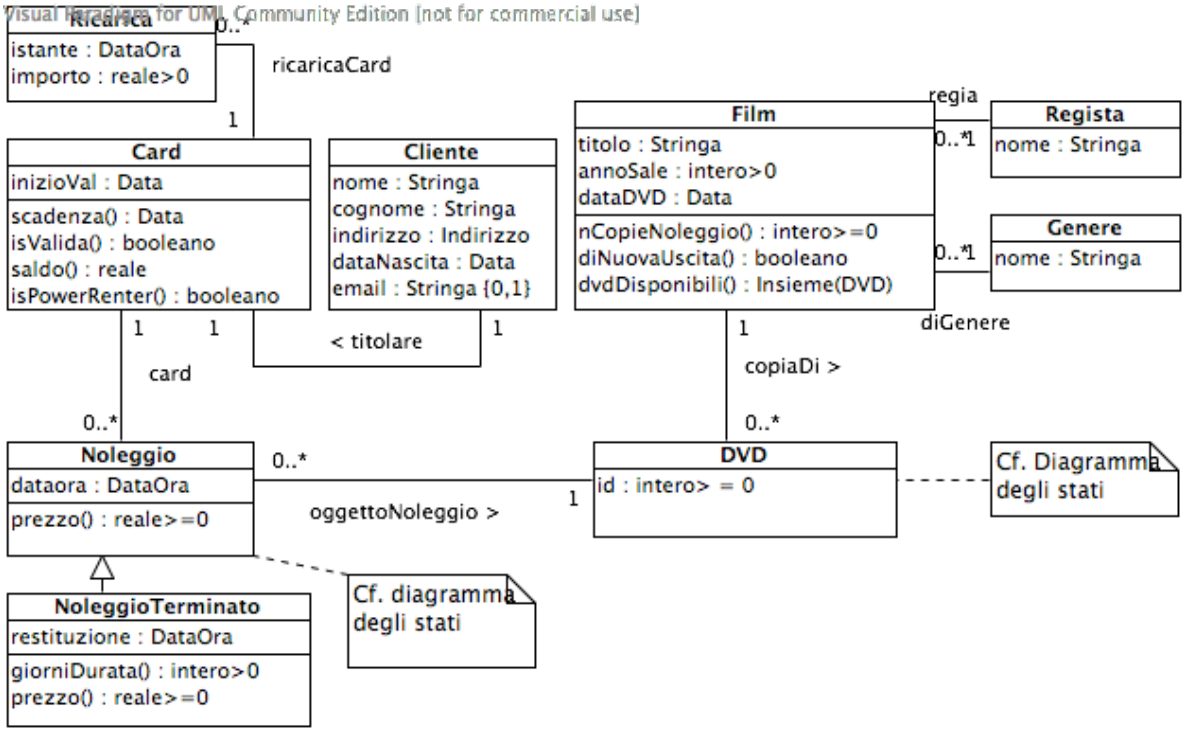
Per favorire i clienti assidui si decide di realizzare un'operazione che suggerisca loro i prossimi film da noleggiare. In particolare, è richiesto che vengano suggeriti i film (mai noleggiati dal cliente in questione) dello stesso genere di quello oggetto del loro ultimo noleggio, di cui ci sono copie disponibili, e che sono stati oggetto di almeno 20 noleggi negli ultimi 30 giorni.

1 Fase di Analisi

1.1 Diagramma degli Use Case del passo A.1



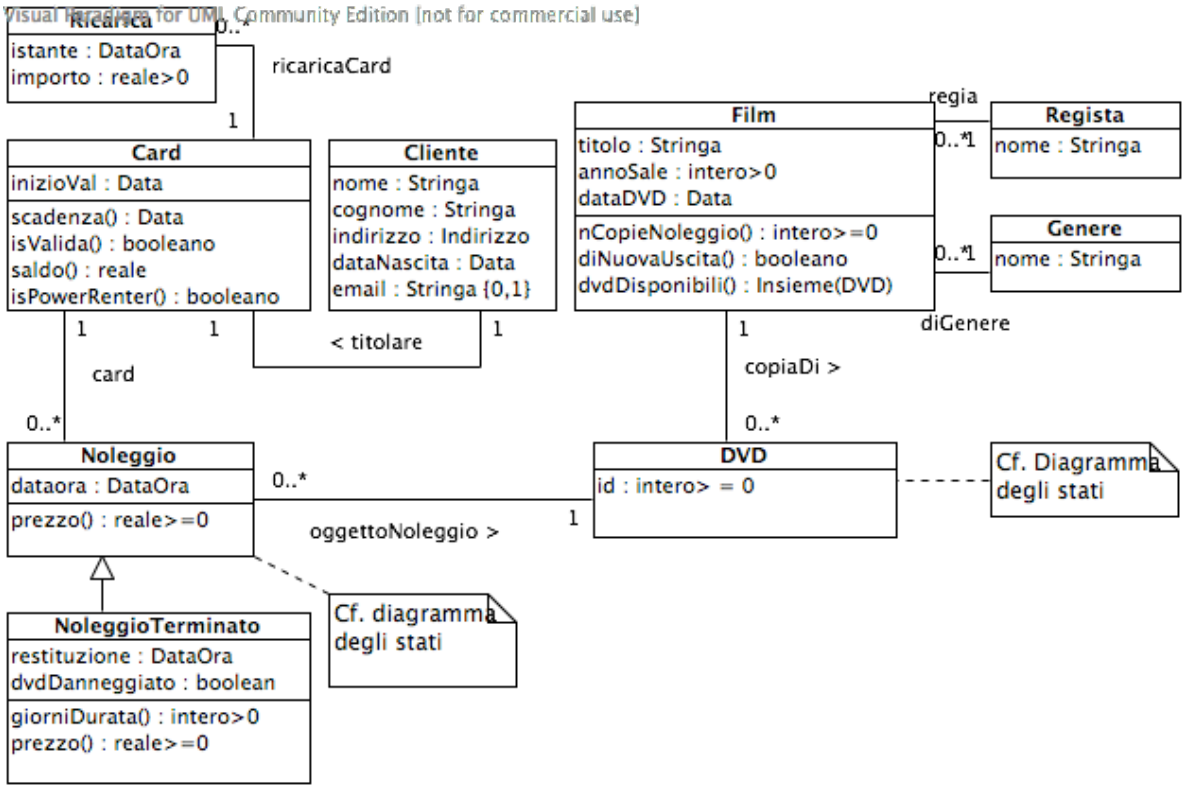
1.2 Diagramma delle classi UML del passo A.1



1.3 Diagramma degli Use Case

Invariato (l'operazione di use-case `suggerisciFilm()` viene definita nello use-case `Noleggio`

1.4 Diagramma delle classi UML



1.5 Specifica dei tipi di dato

Nessun tipo di dato definito

1.6 Specifica degli use case

```
SpecificaUseCase NoleggioDVD
  noleggio(c:Card, f:Film): DVD
    ... (cf. passo A.1)
```

```
  restituzione(c:Card, n:Noleggio, danni:booleano)
     pre:
       - c.isValida() = true
       - n si trova nello stato 'Attivo'
       - esiste il link <c,n>:card
     post: n passa nello stato 'Terminato', diventando di classe
              NoleggioTerminato (cf. diagramma degli stati della classe Noleggio).

              Inoltre, n.restituzione = 'adesso', e n.dvdDanneggiato = danni

              Infine, viene generato l'evento 'restituzione' su n.oggettoNoleggio.DVD
              (il DVD noleggiato, quindi, passa nello stato 'Ritirato' o 'Disponibile'
              a seconda che sia stato danneggiato o meno)

   suggerisciFilm(c:Card): Insieme(Film)
     pre: |c.card| > 0 (il cliente della card c ha effettuato altri noleggi)
     post:
         Sia U l'ultimo noleggio effettuato con la card c:

                  U = argmax_{n.dataora} {n:Noleggio|⟨c,n⟩ : card}

          Detto g = U.oggettoNoleggio.DVD.copiaDi.Film.diGenere.Genere il genere del
          film U, sia F l'insieme di tutti i film di genere g:

          F = { f:Film | f.diGenere.Genere = g }

          result = F - {f:F | esiste n:Noleggio per cui
                                       n.oggettoNoleggio.DVD.copiaDi.Film in F e
                              <c,n>:card
                        }
                      - { f:F |  |f.dvdDisponibili()|=0 }
                      - { f:F |  |quantiNoleggi(f, oggi-30giorni, oggi)| < 20 }

  quantiNoleggi(f:Film, da:Data, a:Data): intero>=0  (operaz. ausiliaria)
    pre: da <= a
    post:
       Detto N = {n:Noleggio | n.oggettoNoleggio.DVD.copiaDi.Film = f  e
                                          da <= n.dataora.data <= a }
       result = |N|
FineSpecifica
```

```
SpecificaUseCase RicaricaCard
  ... (cf. passo A.1)

SpecificaUseCase GestioneCard
  ... (cf. passo A.1)

SpecificaUseCase GestioneDVD
  ... (cf. passo A.1)
```

1.7 Specifica delle classi e diagrammi degli stati e transizioni

La classe Card

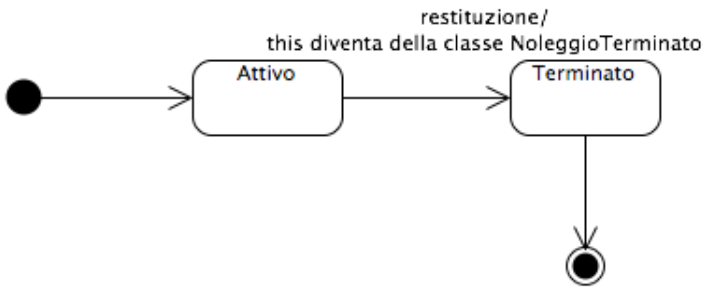
```
SpecificaClasse Card
  ... (cf. passo A.1)
```

La classe Noleggio

```
SpecificaClasse Noleggio
  prezzo(): reale >= 0
    pre: this e' nello stato 'Terminato' (e quindi e' della classe NoleggioTerminato)
    post: cf. specifica della sottoclasse NoleggioTerminato
FineSpecifica

SpecificaClasse NoleggioTerminato
  giorniDurata(): intero>0
    pre: nessuna
    post: result = parteInteraSup(restituzione.differenza(this.dataora))

  prezzo(): reale >= 0
    pre: this e' nello stato 'Terminato' (sempre verificata)
    post:
      Sia p1 = this.giorniDurata()*Tariffe.tariffaGiornaliera().
      Sia p2 = p1 * 1.3 se this.oggettoNoleggio.DVD.copiaDi.Film.isNuovaUscita() = true
                   p1, altrimenti

      Sia p3 = p2 / 1.2 se this.card.Card.isPowerRenter() = true
```

```
                    p2, altrimenti

      result = p3 + Tariffe.penale() se this.dvdDanneggiato = true
                   p3, altrimenti.
FineSpecifica
```

Gli oggetti di questa classe evolvono secondo le regole imposte dal seguente diagramma degli stati e transizioni:



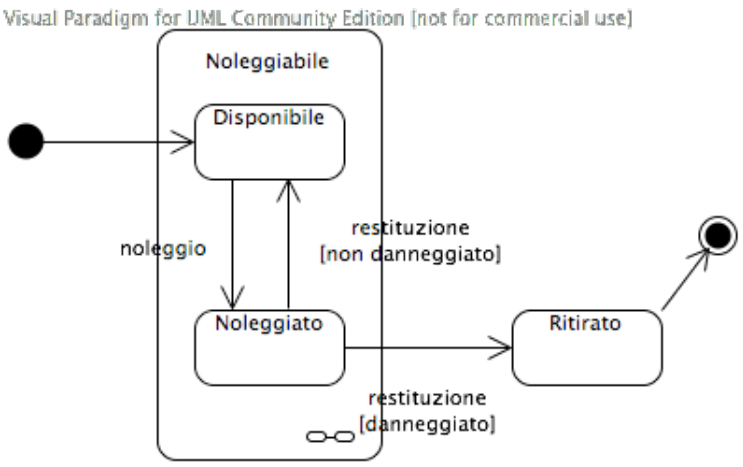
La classe Film

```
SpecificaClasse Film
  nCopieNoleggio() : intero >= 0
    pre: nessuna
    post: result = | { d:DVD | <d,this>:copiaDi e d e' nel macro-stato 'Noleggiabile' } |

  diNuovaUscita(): booleano
    ... (cf. passo A.1)

  dvdDisponibili(): Insieme(DVD)
    ... (cf. passo A.1)
FineSpecifica
```

La classe DVD

Specifica non necessaria (nessuna operazione)

Gli oggetti di questa classe evolvono secondo le regole imposte dal seguente diagramma degli stati e transizioni:


Visual Paradigm for UML Community Edition [not for commercial use]
Noleggiabile
Disponibile
noleggio
restituzione
[non danneggiato]
Noleggiato
Ritirato
restituzione
[danneggiato]