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.3**

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.

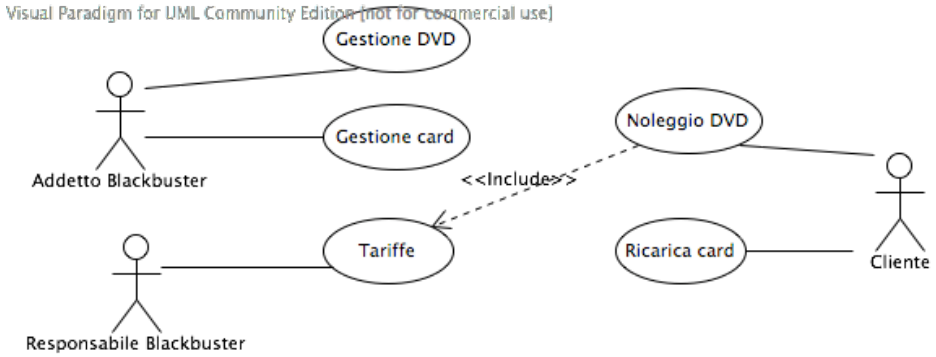
Requisiti

Oltre ai noleggi, il sistema deve permettere ai clienti anche di *acquistare* (alcune delle) copie su DVD dei film offerti dalla catena. Tuttavia, non tutti i film, e non tutte le copie di un film, possono essere acquistati. In particolare, si può acquistare una copia di un film solo se questo è uscito nelle sale cinematografiche da almeno 2 anni, e solo se ne restano a disposizione per il noleggio almeno altre 2 copie. Come per le tariffe di noleggio, il prezzo di vendita di un DVD (uguale per tutti i film) può essere ricavato da una opportuna operazione `prezzoVendita()` presente nello use-case `Tariffe`. Tale prezzo viene immediatamente addebitato sulla card del cliente. Ovviamente, le copie di un film acquistate non potranno essere più oggetto di noleggi.

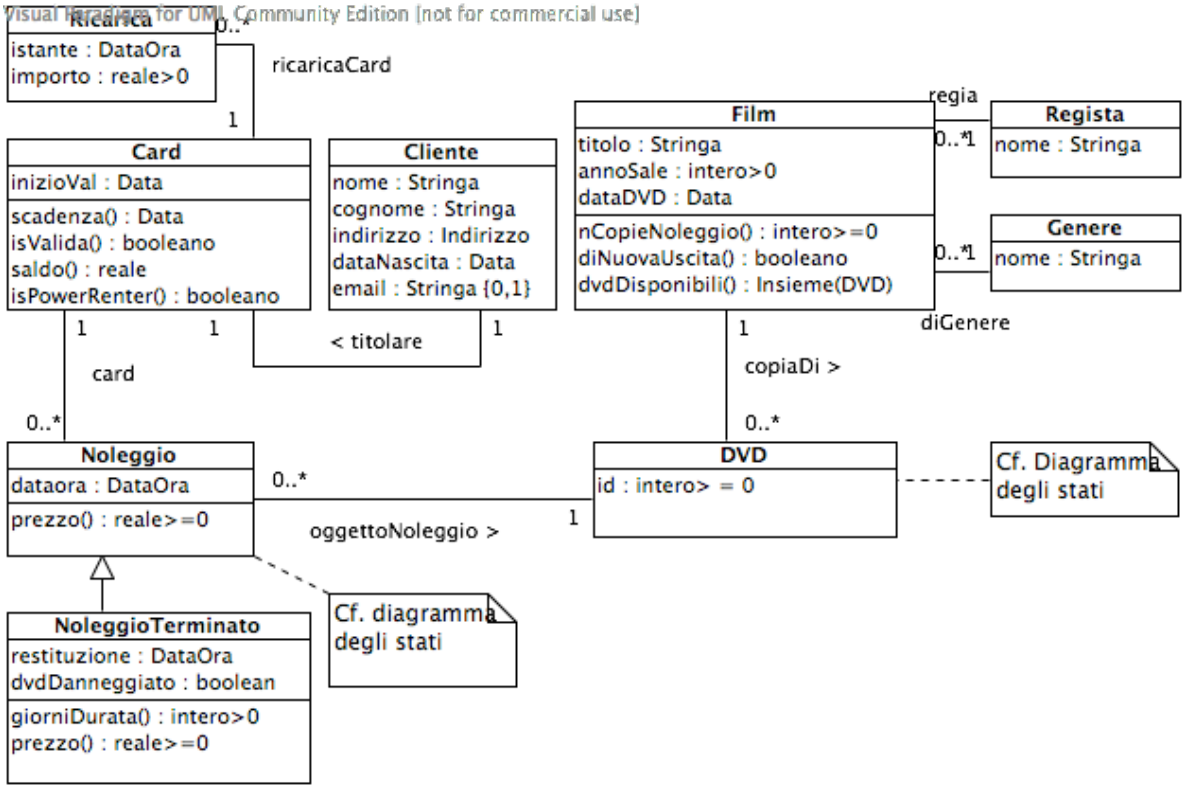
Il calcolo del saldo di una card deve riflettere queste estensioni.

1 Fase di Analisi

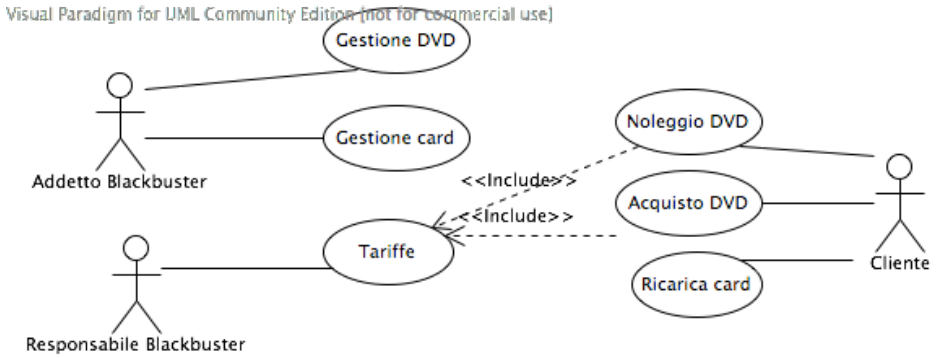
1.1 Diagramma degli Use Case del passo A.2



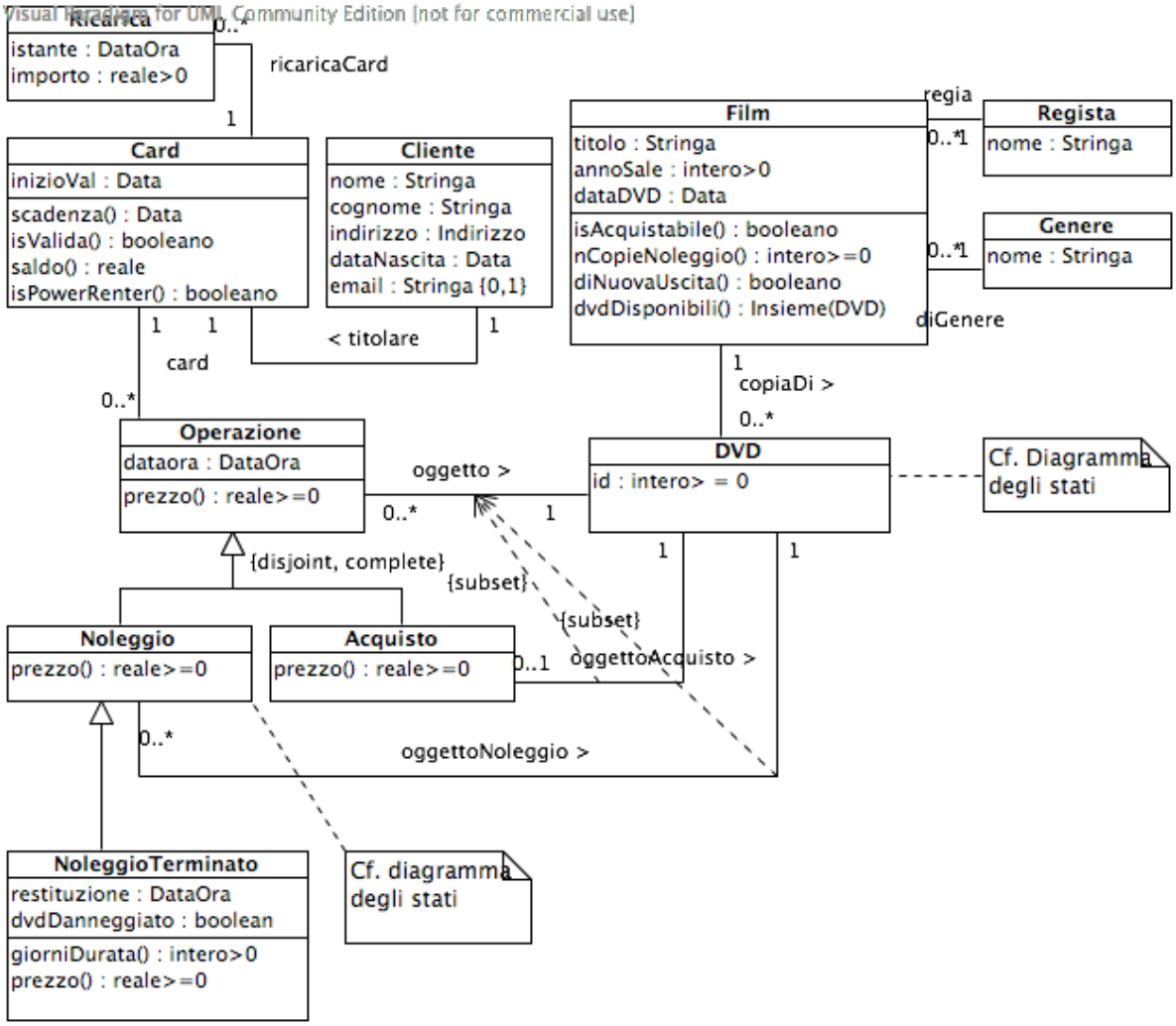
1.2 Diagramma delle classi UML del passo A.2



1.3 Diagramma degli Use Case



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`

```
    ... (cf. passo A.2)
FineSpecifica


SpecificaUseCase AcquistoDVD
  acquisto(c:Card, f:Film): DVD
    pre:
      - c.isValida() = true
      - f.isAcquistabile() = true
      - |f.dvdDisponibili()| > 0
      - c.saldo() >= Tariffe.prezzoVendita()
    post:
        Sia d un arbitrario elemento di f.dvdDisponibili();

        Viene creato 'a', nuovo oggetto di classe Acquisto, con
        n.dataora = 'adesso' (i.e., l'istanza del tipo DataOra che denota
        l'istante corrente), e vengono creati i seguenti link:
        - <c,a>:card
        - <a,d>:oggettoAcquisto

        Viene generato l'evento 'acquisto' su d
        (d passa quindi nello stato 'Acquistato', cf. diagramma degli stati
        della classe DVD)

        result = d
FineSpecifica

SpecificaUseCase RicaricaCard
    ... (cf. passo A.2)
FineSpecifica


SpecificaUseCase GestioneCard
    ... (cf. passo A.2)
FineSpecifica

SpecificaUseCase GestioneDVD
    ... (cf. passo A.2)
FineSpecifica
```

1.7 Specifica delle classi e diagrammi degli stati e transizioni

La classe Card

```
SpecificaClasse Card
  scadenza(): Data
    ... (cf. passo A.2)

  isValida():booleano
    ... (cf. passo A.2)

  saldo(): reale
    pre: nessuna
    post:
      Detto R = { r:Ricarica | <this,r>:ricaricaCard } l'insieme delle ricariche
      effettuate sulla card this, sia
```

$$\texttt{ImpRic} = \Sigma_{\texttt{r:R}}\texttt{r.importo}$$

```
      il loro importo totale.
```

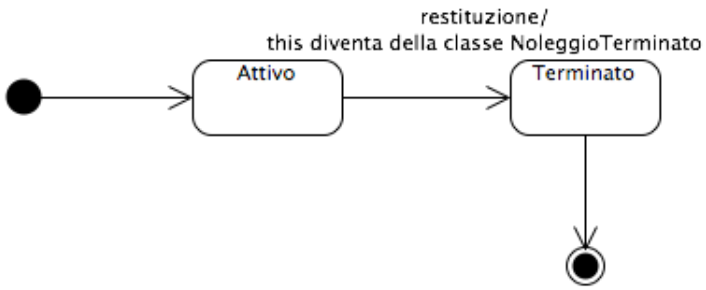
$$\texttt{result} = \texttt{ImpRic} - \Sigma_{\texttt{l:this.card t.c. le prec. di l.Operazione.prezzo() sono verif.}}\texttt{l.Operazione.prezzo()}$$

```
  isPowerRenter():booleano
    ... (cf. passo A.2)
FineSpecifica
```

La classe Noleggio

```
SpecificaClasse Noleggio
    ... (cf. passo A.2)
FineSpecifica

SpecificaClasse NoleggioTerminato
    ... (cf. passo A.2)
FineSpecifica
```

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



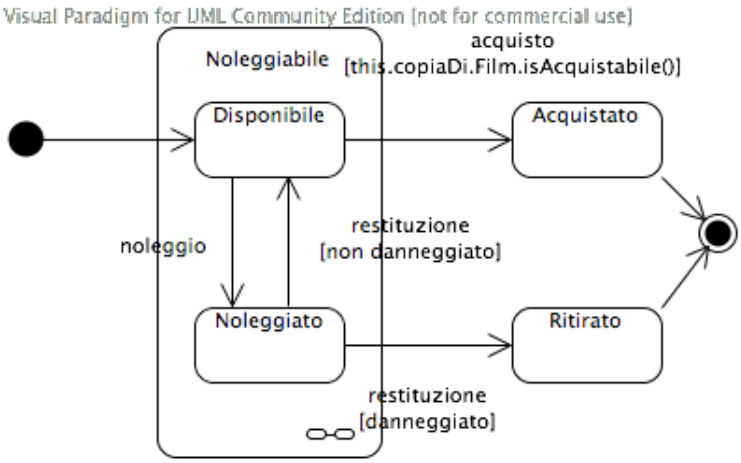
La classe Film

```
SpecificaClasse Film
  isAcquistabile(): booleano
    pre: nessuna
    post: result e' pari a true se:
             - oggi.anno - this.annoSale >= 2, e
             - this.nCopieNoleggio() > 2

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

  diNuovaUscita(): booleano
    pre: nessuna
    post: result = true se oggi.differenza(this.dataDVD, GIORNI) < 30, false altrimenti

  dvdDisponibili(): Insieme(DVD)
    pre: nessuna
    post: result = { d:DVD | <d, this>:copiaDi e d e' nello stato 'Disponibile' }
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
acquisto
[this.copiaDi.Film.isAcquistabile()]
Disponibile
Acquistato
noleggio
restituzione
[non danneggiato]
Noleggiato
Ritirato
restituzione
[danneggiato]