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.1**

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 effettuare la fase di Analisi, producendo uno schema concettuale per l'applicazione i cui requisiti sono descritti in calce.

Requisiti

Il sistema *Blackbuster* viene utilizzato (oltre che dal personale della catena) dagli utenti finali attraverso degli opportuni terminali (simili a quelli del Bancomat). I noleggi di DVD vengono effettuati mediante l'utilizzo di tessere prepagate ricaricabili, chiamate *Blackbuster card*.

Le card vengono inizialmente richieste dai clienti al personale addetto (una per cliente): sono quindi nominative, ed il sistema deve poter memorizzare nome, cognome, indirizzo, data di nascita ed indirizzo email (se fornito) del titolare. Le card hanno inoltre una durata temporale di un anno (allo scadere del quale, è prevista una apposita procedura di rinnovo presso il personale addetto), e possono essere ricaricate dagli stessi clienti presso un qualunque terminale.

Delle card è di interesse calcolare il saldo (credito residuo) all'istante corrente. Tuttavia, il sistema deve tener traccia anche delle *diverse singole* ricariche effettutate su una card di modo da consentire al sistema di attivare promozioni di vario genere a beneficio dei clienti.

È richiesto che i clienti della catena *Blackbuster* possano, mediante i terminali, noleggiare film in DVD, utilizzando la loro card personale. Dei film in DVD gestiti dal sistema è di interesse conoscere titolo, regista, genere, anno di uscita nelle sale cinematografiche, e data di uscita dell'edizione DVD. Di ogni film, la catena può possedere più copie (di cui interessa il numero) su altrettanti supporti (i dischi DVD veri e propri), e, come si vedrà in seguito, è di interesse per il sistema mantenere traccia dei noleggi delle *singole* copie di un film. Il numero

di copie di un film non è fisso, in quanto il personale della catena può crearne di nuove per sopperire a eventuali carenze nell'offerta, mentre altre possono essere ritirate, ad es., in caso di deterioramento o di vendita (cf. seguito).

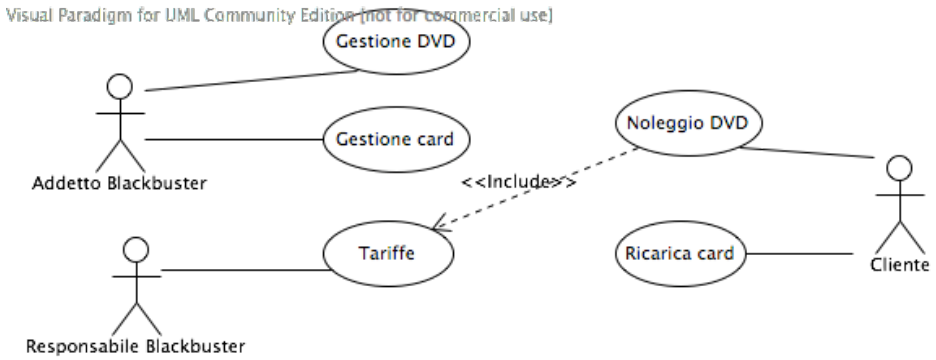
Il noleggio di un film in DVD prevede la verifica preventiva che ci siano copie disponibili, ed in caso positivo viene effettuato attribuendo una qualunque copia disponibile del film alla card del cliente. All'atto della restituzione, il prezzo del noleggio viene detratto dalla card.

Le componenti che concorrono alla determinazione di tale prezzo sono due: la durata del noleggio (in giorni) e l'informazione sul fatto che il film sia di nuova uscita o meno (i film di nuova uscita sono quelli la cui edizione DVD è uscita da meno di 30 giorni). In particolare, detta G la tariffa giornaliera fissata dal gestore del sistema (uguale per tutti i film), il costo di un noleggio di g giorni è dato $g \cdot G$. Se il film è di nuova uscita, tale prezzo viene aumentato del 30%. Tuttavia, va previsto uno sconto finale del 20% sul prezzo applicato ai noleggi da parte dei clienti *PowerRenter*, ovvero di coloro che hanno effettuato ricariche complessive, nel corso degli ultimi 12 mesi, di almeno 100 euro. Per semplicità, si può assumere che per le durate dei noleggi vengano considerati solo giorni interi, e che il valore di G possa essere ricavato da una opportuna operazione `tariffaGiornaliera()` presente in uno use-case `Tariffe`, del quale non è richiesta la progettazione (ma se ne tenga conto in modo opportuno nel diagramma degli use-case).

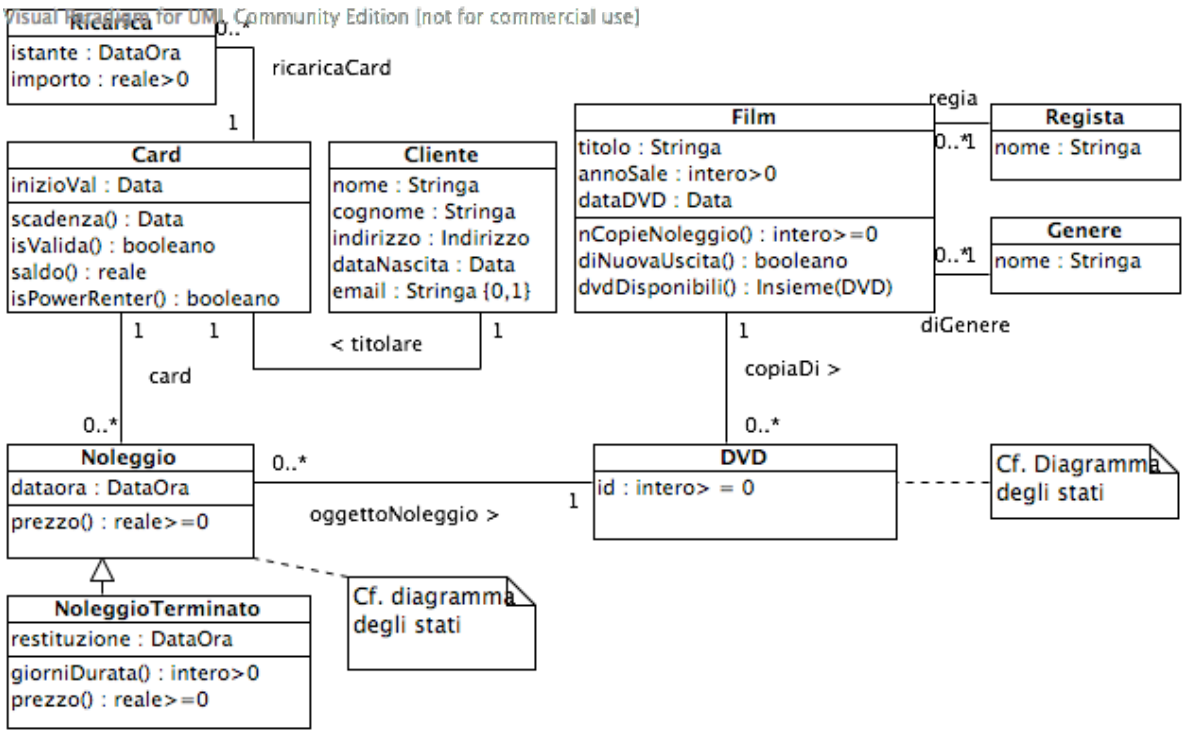
Dato che il prezzo di un noleggio dipende dalla sua durata, non è possibile, in fase di acquisizione di un DVD, verificare che il saldo della card sia sufficiente. In altre parole, è possibile che, al momento della restituzione (e del pagamento), il saldo della card del cliente diventi *negativo*. È richiesto che in questi casi il sistema non permetta al cliente di prendere a noleggio altri DVD (fino a che, procedendo alla ricarica, il saldo non torni positivo).

1 Fase di Analisi

1.1 Diagramma degli Use Case



1.2 Diagramma delle classi UML



1.3 Specifica dei tipi di dato

Nessun tipo di dato definito

1.4 Specifica degli use case

```
SpecificaUseCase NoleggioDVD
  noleggio(c:Card, f:Film): DVD
    pre: c.isValida()=true, c.saldo() > 0 e |f.dvdDisponibili()| > 0
    post:
        Sia d un arbitrario elemento di f.dvdDisponibili();

        Viene creato 'n', nuovo oggetto di classe Noleggio, con
        n.dataora = 'adesso' (i.e., l'istanza del tipo DataOra che denota
        l'istante corrente), e vengono creati i seguenti link:
        - <c,n>:card
        - <n,d>:oggettoNoleggio
        (Dal diagramma degli stati della classe Noleggio, segue che
        d si trovera' nello stato 'Attivo')

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

        result = d

  restituzione(c:Card, n:Noleggio)
     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'

              Infine, viene generato l'evento 'restituzione' su n.oggettoNoleggio.DVD
              (il DVD noleggiato, quindi, passa nello stato 'Disponibile')
FineSpecifica
```

```
SpecificaUseCase RicaricaCard
  ricarica(c:Card, importo: reale>0)
    pre: c.isValida() = true
    post: Viene creato un nuovo oggetto r: Ricarica con
              - r.istante = 'adesso'
              - r.importo = importo

            Viene inoltre creato il link <c,r>:ricaricaCard
            (ovvero, r viene aggiunta all'elenco delle ricariche della card c)
FineSpecifica


SpecificaUseCase GestioneCard
  inserisciNuovoCliente(nome:Stringa, ...): Cliente
    pre: ...
    post: result e' un nuovo oggetto di classe Cliente con...

  rilasciaNuovaCard(c:Cliente): Card
    pre: nessuna
    post: result e' un nuovo oggetto di classe Card con
              - result.inizioVal = 'oggi'
             Inoltre, viene creato il link <result, c>:titolare
FineSpecifica

SpecificaUseCase GestioneDVD
  aggiungiDVD(f:Film): DVD
    pre: nessuna
    post: result e' un nuovo oggetto di classe DVD con:
             - result.copiaDi.Film = f
             - result.id = M+1

             dove
```

$$\texttt{M} = \max_{\texttt{d:DVD t.c. } \langle \texttt{d,f} \rangle \texttt{:copiaDi}} (\texttt{d.id}).$$

```
            Viene inoltre creato il link <result, f>:copiaDi.
FineSpecifica
```

1.5 Specifica delle classi e diagrammi degli stati e transizioni

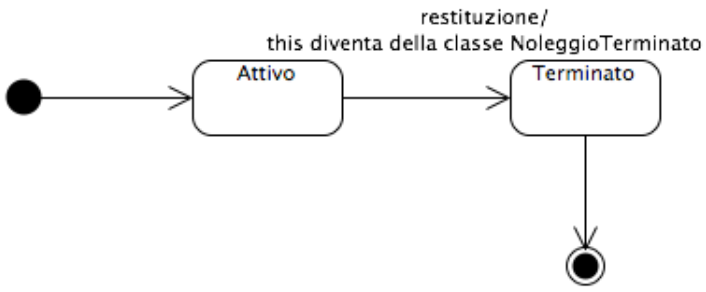
La classe Card

```
SpecificaClasse Card
  scadenza(): Data
    pre: nessuna
    post: result e' pari a this.inizioVal + 1 anno

  isValida():booleano
    pre: nessuna
    post: result = true se adesso <= this.scadenza(), false altrimenti

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

                              ImpRic = Σ_{r:R} r.importo

      il loro importo totale.

result = ImpRic − Σ_{l:this.card t.c. le prec. di l.Noleggio.prezzo() sono verif.} l.Noleggio.prezzo()

  isPowerRenter():booleano
    pre: nessuna
    post:
       Detto R = { r:Ricarica | <this,r>:ricaricaCard e
                                    adesso.differenza(r.istante, ANNI) <= 1}
       l'insieme delle ricariche effettuate sulla card this nell'ultimo anno, sia

                        ricaricatoUltimoAnno = Σ_{r:R} r.importo

        il relativo importo complessivo.
        Se ricaricatoUltimoAnno >= 100, result = true, altrimenti result = false.
FineSpecifica
```

La classe Noleggio

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

FineSpecifica
```

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



La classe Film

```
SpecificaClasse Film
  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: